European Journal of Operational Research 309 (2023) 910–924

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Innovative Applications of O.R.

Models and algorithms for a yard crane scheduling problem in container ports

Eva Vallada[a,*], Jose Manuel Belenguer[b], Fulgencia Villa[a], Ramon Alvarez-Valdes[b]

[a] *Grupo de Sistemas de Optimización Aplicada, Instituto Tecnológico de Informática, Ciudad Politécnica de la Innovación, Edifico 8G, Acc. B. Universitat Politècnica de València, Camino de Vera s/n, València, 46021, Spain*

[b] *University of Valencia, Department of Statistics and Operations Research, Doctor Moliner 50, Burjassot, 46100, València, Spain*

ARTICLE INFO

Article history:
Received 1 April 2022
Accepted 25 January 2023
Available online 30 January 2023

Keywords:
Heuristics
Container terminal
Yard crane scheduling
Integer models
Local search

ABSTRACT

This paper addresses the scheduling of a yard crane in an automated container block, considering its relationships with the other terminal subsystems. Containers coming into the block to be stored have release times, indicating the moment at which they arrive from ships or trucks. Containers being retrieved have due times, indicating when they will be required by other subsystems. The problem can be seen as a pickup routing problem and also as a one-machine scheduling problem. As a starting point, integer linear models were developed for both approaches. Then, to obtain good solutions for large instances, several heuristic algorithms, coupled with a local search, were proposed. An extensive computational study, based on a newly generated benchmark, shows the limits of the exact methods and the quality of the solutions obtained by the heuristics.

1. Introduction

Containerized maritime transportation is an essential part of world trade. According to UNCTAD (2021), the total number of containers moved in 2020 reached 149 million Twenty-foot Equivalent Units, just a 1.1% decrease compared to 2019, due to the pandemic, and this number is already increasing in 2021 and is expected to continue to grow in the coming years. These containers pass through maritime container terminals all over the world. Therefore, terminals must be able to handle large and increasing amounts of containers on a daily basis.

To provide the most efficient service to calling vessels, container terminals try to reduce their turnaround time, the time it takes for the berthed vessel to load and unload containers. Quay cranes perform these unloading and loading operations on the vessel. Multiple quay cranes are assigned to a vessel, depending on its size. Their performance has increased significantly in recent years. However, to obtain maximum efficiency from them, all other terminal resources have to be synchronized to ensure that the quay cranes are continuously operating.

Fig. 1 shows a simplified view of an automated terminal, identifying its three subsystems: seaside, storage yard, and landside. On the seaside, vessels are berthed at the quay, some quay cranes are assigned to each of them and some vehicles, usually Automated Guided Vehicles (AGV), transfer containers to and from the storage area. The storage yard is divided into blocks, consisting of multiple bays, rows, and tiers, and each yard crane (YC) is assigned to a single block. The dashed rectangles on the right show the structure of bays and rows. The input/output (I/O) points, located on the seaside and landside of a block, are the only transfer points between AGVs and YCs on the seaside and between YCs and trucks or trains on the landside. The YCs move containers from the block locations to and from I/O points. For storage requests, the yard crane picks up the container from its I/O point and stores it at a block location. For retrieval requests, the YC picks it up at its location in the block and moves it to the assigned I/O point.

This paper addresses the problem of scheduling the tasks of a single yard crane serving a block. At any given moment, the crane has a number of storage and retrieval requests, coming from the seaside and from the landside. These requests have to be sequenced taking into account the arrival time of the containers to be dropped off at the I/O points, defining the release times for these tasks, and, most importantly, the time at which the containers to be retrieved must be on the seaside to comply with the order in which they must be loaded on vessels or at the landside to be picked up by trucks or trains, defining due times. The objective is to minimize the weighted sum of the costs (earliness, tardiness, waiting costs) involved in the process.

As can be seen in Fig. 1, the number of I/O points is usually larger on the seaside, where AGVs pick up and drop off containers, than on the landside, where wider trucks are used to move the containers. In any case, the number of I/O points is limited and if at any given time there are many containers on one side of the

* Corresponding author.
E-mail address: evallada@eio.upv.es (E. Vallada).

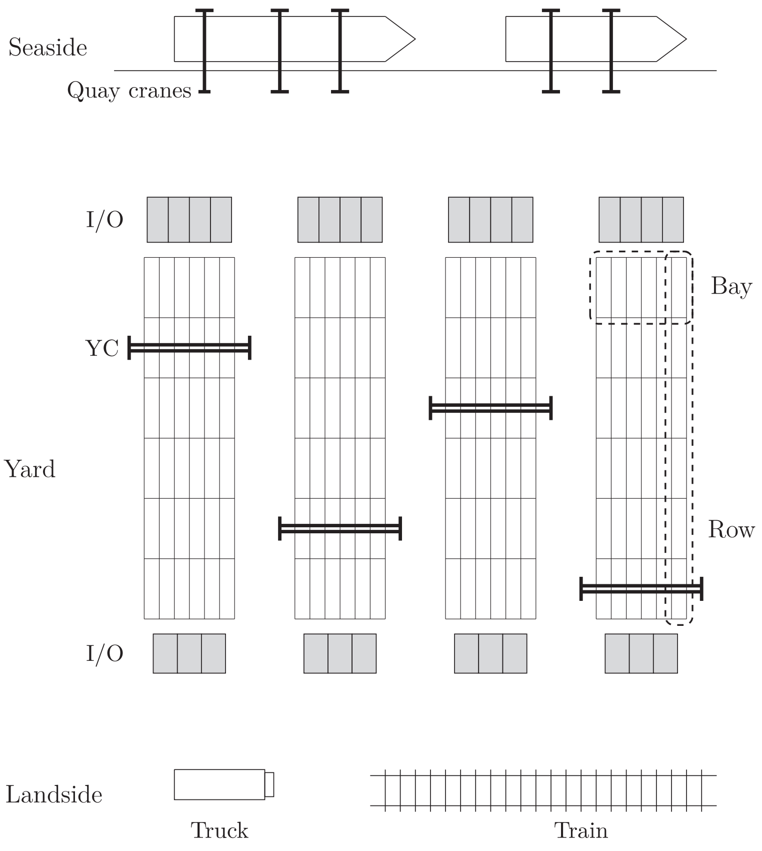


Fig. 1. Top view of an automated terminal.

block, congestion problems may arise, forcing a loaded vehicle to wait before dropping off its container, or an empty vehicle to wait before it can pick up its assigned container. Therefore, assigning one I/O point to each request is part of the YC planning, in addition to solving potential congestion problems.

The problem of scheduling the movements of a crane can be seen as a special type of pickup and delivery problem in which a vehicle of capacity one has to fulfil a set of requests. Instead of time windows, some requests have a release time before which they cannot be served and others have a due time, with associated penalties. As the origin or destination of each request is an I/O point that has not been previously fixed, the travel time between two requests remains unknown as well.

The problem can also be viewed as a one-machine scheduling problem in which the jobs are the movements of the crane transporting a container and the moves of the crane from the destination of one request to the origin of another are considered setup times. Again, neither the processing times nor the setup times are fixed, because they depend on the I/O points assigned to the requests.

Seen from each of the two perspectives, the problem studied here is a especial case of a very hard problem. However, it is necessary to obtain good solutions, to avoid the yard being the terminal bottleneck, and obtain them quickly, because the requirements change continuously over time.

In the study of this crane scheduling problem which takes into account its relationship with other terminal subsystems, the main contributions of our work are:

- An integer linear model that considers the problem as a special case of a pickup routing problem, but includes not only the sequence and timing of the crane movements but also the assignment of I/O points for each container that is stored in or retrieved from the yard, thus addressing potential congestion problems at the I/O points.
- An alternative integer linear model in which the problem is considered as a one-machine scheduling problem, again including the I/O point assignment.
- Several sequencing heuristics that use the characteristics of the problem to design priority rules and a local search to improve the sequences obtained.
- A benchmark of problems that includes release and due times for the container as well as the number and position of the I/O points.
- An extensive computational study that first shows the performance and limitations of the integer linear models and then the behaviour of the heuristic rules with and without local search.

The remainder of this paper is organized as follows. In Section 2, related studies are reviewed and in Section 3 the problem is described. Integer linear models are presented in Section 4, while heuristic algorithms are described in Section 5. The computational results are presented and discussed in Section 6 and Section 7 presents the conclusions of the study and outlines future work.

2. Literature review

Yard crane scheduling problems can be seen as a variant of the general crane scheduling problem of assigning storage and retrieval moves to cranes and deciding the order in which each crane performs these moves to maximize a measure of process efficiency. There are many types of cranes, appearing in different industrial settings: Automated Storage and Retrieval Systems for moving products in distribution centres; Quay Cranes for loading and unloading container vessels in ports; Yard or Gantry Cranes in container terminal yards used as container buffer storage, Gantry Cranes in railway terminals, transferring containers between trains and trucks; or Industrial Cranes, moving heavy goods in production processes. Although these problems have usually been studied independently, some recent surveys consider them together and propose unifying classifications. Boysen, Briskorn, & Meisel (2017) develop an $\alpha|\beta|\gamma$ scheme in which all the above crane types fit, where α refers to the terminal layout, β to the characteristics of container movements, and γ to the objectives. Boysen & Stephan (2016) focus on automated crane systems, which also include many cases in container terminals, and propose a similar classification scheme. A survey dealing only with yard crane scheduling was provided by Carlo, Vis, & Roodbergen (2014). These surveys cover all numbers and configurations of cranes. Since this paper only considers the case of a single yard crane, only previous studies with one crane will be reviewed.

One of the findings of Boysen et al. (2017) is that in the vast majority of studies the containers are accessed sideways to the crane path and they conclude that future research should intensify work on non-sideways cases, where I/O points are at both ends of the storage area. This is a very important issue in yard crane scheduling. Most early work studied the case of non-automated Asian terminals where there is a truck lane along the block and trucks can be positioned next to the bay where the container is to be stored or retrieved. Kim & Kim (1999) were the first to study the single yard crane problem, considering only retrieval moves and aiming to minimise the total container handling time. They proposed an integer linear model and, in a later work, genetic and beam search algorithms (Kim & Kim, 2003). Narasimhan & Palekar (2002) studied a similar problem, proving its *NP*-completeness and proposing exact and heuristic algorithms. Ng & Mak (2005b) extended the problem, including retrieval and storage moves and ready times, and proposed first a branch and bound algorithm and then a heuristic procedure (Ng & Mak, 2005a) to minimize the sum of the job waiting times.

In more recent years, the problem of automated European terminals, in which containers in a block in the yard can only be accessed through I/O points at the seaside and landside ends of the block, has been studied. Gharehgozli, Yu, de Koster, & Udding (2014) consider retrieval moves from given positions in the block and storage moves from I/O points to specified positions in the block and minimise the total crane travel time. The problem can be seen as an Asymmetric Traveling Salesman Problem and is therefore NP-hard. They propose an exact solution method in which the optimal solution of the assignment problem obtained by relaxing the subtour elimination constraints becomes a solution to the original problem by first merging subtours and then, if necessary, using a branch and bound procedure. More recently, Gharehgozli, Yu, de Koster, & Du (2019) have extended the problem by considering that storage moves do not have a position fixed in advance, but that the position has to be selected from a set of open locations. The problem then becomes a Generalised Asymmetric Traveling Salesman Problem and an exact three-phase algorithm is developed for solving small instances. For large instances, the authors propose a heuristic algorithm. Galle, Barnhart, & Jaillet (2018) consider the combined yard scheduling and container relocation problem for the single crane case and propose first an integer model and then a local search heuristic that can be applied to any yard configuration.

Another important issue in yard crane scheduling problems is the definition of objectives. As Boysen & Stephan (2016) and Kemme (2020) point out, yard crane moves are heavily interdependent with other seaside and landside processes and, therefore, minimising crane travel time may not be the most important objective, as it is not necessarily aligned with minimising turnaround times of vessels and external trucks. Instead, it may be more relevant to satisfy due times defining when containers are required. Although most papers dealing with yard crane scheduling consider the minimisation of the total crane travel time, there are some studies that take release and due times into account. The case of non-automated terminals with several cranes per block has been considered by Li, Wu, Petering, Goh, & de Souza (2009), Li et al. (2012), and Wu, Li, Petering, Goh, & de Souza (2015), focusing on the main sources of delay, crane interference and distance requirements between containers. Huang & Li (2017) propose a yard crane scheduling system to minimise the total weighted maximum tardiness of loading jobs. Zheng, Man, Chu, Liu, & Chu (2019) and Liu, Zhu, Wang, Yan, & Zhang (2021) extend the study, considering uncertain release times and proposing stochastic programming formulations. Gao & Ge (2022) jointly schedule trucks and yard cranes, minimising total crane distance and truck waiting times. Chen, He, Zhang, Tong, & Shanga (2020) address the problem of yard cranes and AGVs serving them as a single combined problem. Xing et al. (2023) consider the combined problem of quay cranes, AGVs, and yard cranes, minimizing crane makespan the AGV consumption. The case of automated terminals with two or three cranes per block has been studied by Choe, Park, Ok, & Ryu (2010), Choe, Yuan, Yang, & Ryu (2012), Dorndorf & Schneider (2010), Park, Choe, Ok, & Ryu (2010) and Gharehgozli, Laporte, Yu, & de Koster (2015), considering AGV delays on the seaside and external truck waiting times on the landside.

3. Description of the problem

The problem is to schedule the movements of a crane to serve a set of container storage and retrieval requests in a single block composed of R rows, B bays, and T tiers. The block is accessed through a set of seaside I/O points, IO^S, and a set of landside I/O points, IO^L. In Fig. 1, each block has 6 rows, 6 bays, 4 seaside I/O points, and 3 landside I/O points. A single yard crane moves containers between the block locations and the I/O points. The crane can only carry one container at a time and cannot leave a container at a temporary position. Its initial position, *Ini*, is known, but its final position is not fixed and corresponds to the destination of the last container in the sequence.

For a given time interval, the set of storage and retrieval requests of a block can be divided into four subsets:

- C_1 (Sea to Yard): containers coming from a vessel to be stored in the block at a known position. The expected arrival of container c is r_c and an I/O point on the seaside has to be assigned to it.
- C_2 (Land to Yard): containers coming from the landside by train or truck to be stored at a known position. The expected arrival of container c is denoted by r_c and must be assigned an I/O point on the landside.
- C_3 (Yard to Sea): containers to be retrieved from a known position in the block to be loaded onto a vessel. An I/O point on the seaside has to be assigned and the date on which the container has to be delivered at this point is d_c. Lateness and earliness with respect to this date are penalized.
- C_4 (Yard to Land): containers to be retrieved from a known position in the block to be loaded onto a truck or train. An I/O point on the landside has to be assigned and the date on which the pick-up vehicle arrives at this point is e_c, so the container cannot be delivered earlier.

Therefore, the set of containers is $C = \bigcup_{i=1}^{4} C_i$. The position in the block of the containers to be retrieved is known and for the containers to be stored the position in which they will be stored is also known. Therefore, each container c has a location in the block,

l_c, which is the origin in a retrieval movement and the destination in a storage movement. These locations are determined at a higher decision level, trying to ensure easy accessibility for future handling. It is assumed that the locations l_c are different for each container c, in line with previous studies. Moreover, as Gharehgozli et al. (2014) point out, this assumption is reasonable in practice. The number of requests is usually much smaller than the number of positions in the block and the terminal operators separate the retrieval and storage stacks to avoid unnecessary reshuffling. Containers are stacked according to type, limiting the number of options even further. However, the position at the I/O points is not fixed in advance. At any given time, the occupancy of the I/O points depends on the arrival times of the containers to be stored and the sequence of crane movements, so the solution to the problem will include assigning an empty I/O point to each request.

The position of each block location and each I/O point is given by their coordinates (x, y, z), where x denotes the row, y the bay, and z the tier. Using these coordinates, all pairwise travel times, t_{ij}, between locations (including I/O points) are calculated, following this expression, very similar to that proposed by Gharehgozli et al. (2014):

$$t_{ij}=|T+1-z_i|+max\{|x_i-x_j|,|y_i-y_j|\}+|T+1-z_j|+Res_{time} \quad (1)$$

where (x_i, y_i, z_i) are the coordinates of location i and each term has been divided by the speed of the crane in the corresponding direction, X, Y, Z. These calculations take into account that the crane can move in X and Y directions simultaneously. The vertical travel time considers the time needed to perform the container pick-up and release operations. If a container to be retrieved from position i is not on the top of its stack, the time to be moved to position j also includes an estimate of the time required for the necessary reshuffling (Res_{time}). The crane starts the first movement at the top, so for the initial crane position the z coordinate is set to $T+1$.

The objective of the problem is to find a sequence that minimises the weighted sum of the costs involved. There are several sources of inefficiency that are penalized in order to find the optimal crane schedule. On the one hand, there are delays. In the case of containers to be stored, $c \in C_1 \bigcup C_2$, we want to minimise their finishing times, putting them in relation to their release times r_c. This includes not only the delays of their starting times with respect to r_c, which will cause unnecessary occupation of I/O points, but also the time taken by the crane to perform the moves, which depends on the I/O point assigned. In the case of containers to be retrieved and loaded onto a vessel, $c \in C_3$, delays with respect to their due time d_c may cause delays in servicing the vessel. Containers of this type can be retrieved before d_c, but they will block I/O points, so this situation is also penalized. For containers $c \in C_4$ that are loaded on trucks or trains, they cannot be delivered before the arrival of the vehicle e_c, and delays with respect to this time will keep the vehicle waiting. This is especially important for trains, moving hundreds of containers, whose schedules are fixed. On the other hand, congestion penalization at I/O points is considered when a container arrives on one side of the block and all the I/O points on this side are occupied. Weights, w_i^R for the delays and w_i^C for the congestion of containers in C_1, C_2, and C_4, plus a weight w^E for the earliness in C_3, are applied to assign a different relative importance to each component in a weighted objective function. Usually, delays to containers being loaded on vessels are considered the most important.

4. Integer linear models

Two integer models are proposed for this problem. The first is based on considering the crane movements as a routing problem, from the initial position *Ini*. The route visits each location l_c of each container c exactly once, and some of the I/O points one or more times. The second model considers the problem as a scheduling problem with only one machine, the crane, and the jobs to be scheduled are the containers. The solution of the two models provides the sequence of movements of the crane transporting the containers and also the occupancy period of the I/O point corresponding to each container.

4.1. Routing model

This model considers the crane's movements as a route that visits all fixed locations of the containers and some I/O points, starting from a prefixed initial point in the block. During its route, the crane alternates empty trips with others in which it carries a container. In the latter case, the movement starts at an I/O point and ends at location l_c, for containers of types 1 or 2, and vice versa, for the other two types. In the first case, the container must have been deposited at the I/O point prior to the arrival of the crane, while in the other, the I/O point cannot be occupied by another container at the time it is deposited by the crane. Consequently, the model must determine the best I/O point to carry out the operation and the period in which this point will be occupied by the container.

While the starting point, *Ini*, of the crane's movement is predetermined, the end point is unknown and will be the point where the movement of the last container transported ends. For this reason, we create a dummy point, *Fin*, which will represent the end of the crane's movement. In addition, the model will include a fictitious final movement from the real point where the transport of the last container ends to *Fin*.

Variables:

Potential movements of the crane holding a container:

Z_{ij}^c	1 if the crane goes from i to j holding $c \in \mathcal{C}$; 0, otherwise where $i \in \mathcal{IO}^{s(c)}$, $j = l_c$ if $c \in \mathcal{C}_1 \cup \mathcal{C}_2$ or $i = l_c$, $j \in \mathcal{IO}^{s(c)}$ if $c \in \mathcal{C}_3 \cup \mathcal{C}_4$

Potential empty movements of the crane:

$X_{Ini\,j}^c$	1 if the crane goes from *Ini* to j to hold $c \in \mathcal{C}$; 0, otherwise where $j \in \mathcal{IO}^{s(c)}$ if $c \in \mathcal{C}_1 \cup \mathcal{C}_2$ or $j = l_c$ if $c \in \mathcal{C}_3 \cup \mathcal{C}_4$
$X_{i\,Fin}^c$	1 if the crane goes from i to *Fin* after holding $c \in \mathcal{C}$; 0, otherwise where $i = l_c$ if $c \in \mathcal{C}_1 \cup \mathcal{C}_2$ or $i \in \mathcal{IO}^{s(c)}$ if $c \in \mathcal{C}_3 \cup \mathcal{C}_4$
$X_{ij}^{cc'}$	1 if the crane goes from i to j after holding $c \in \mathcal{C}$ to move $c' \in \mathcal{C}$; 0, otherwise where $i = l_c$ if $c \in \mathcal{C}_1 \cup \mathcal{C}_2$ or $i \in \mathcal{IO}^{s(c)}$ if $c \in \mathcal{C}_3 \cup \mathcal{C}_4$ and $j \in \mathcal{IO}^{s(c')}$ if $c' \in \mathcal{C}_1 \cup \mathcal{C}_2$ or $j = l_{c'}$ if $c' \in \mathcal{C}_3 \cup \mathcal{C}_4$

Sorting of containers moved by the crane:

$Y_{cc'}$	1 if $c \in \mathcal{C}$ is moved before $c' \in \mathcal{C}$ (not necessarily just before); 0, otherwise

Times of the crane movements:

T_{Ini}	Starting time of the crane
SC_c	Starting time of the movement of $c \in \mathcal{C}$
FC_c	Finishing time of the movement of $c \in \mathcal{C}$

Occupancy times of the I/O points:

SO_c	Starting time of the occupancy of the I/O point used by $c \in \mathcal{C}$
FO_c	Finishing time of the occupancy of the I/O point used by $c \in \mathcal{C}$

Advance/delay of type 3 containers:

E_c, T_c	Earliness/tardiness, with respect to d_c, in exporting $c \in \mathcal{C}_3$ to an I/O point.

Below, for each block, we include the equations or inequalities that compose it and a brief description of them. The routing model is:

Objective function:

$$\text{Min} \sum_{i=1}^{2}\sum_{c\in\mathcal{C}_\rangle}\left(w_i^R(FC_c - r_c) + w_i^C(SO_c - r_c)\right) + \sum_{c\in\mathcal{C}_\ni}\left(w_3^R(FO_c - d_c) + w_3^E E_c\right) + \sum_{c\in\mathcal{C}_\triangle}\left(w_4^R(FO_c - e_c) + w_4^C(SO_c - e_c)\right) \quad (2)$$

The objective function is a weighting of the delay of the containers with respect to their final planned "time" and for congestion (no I/O point assignment).

In the case of containers to be stored, $c \in \mathcal{C}_1 \bigcup \mathcal{C}_2$, we want to minimise the delay $FC_c - r_c$ and the congestion $SO_c - r_c$. The former is the delay of its starting time with respect to r_c, which will cause unnecessary occupation of an I/O point, plus the time taken by the crane to perform the move, which depends on the I/O point assigned. The latter is the lateness in assigning an I/O point, i.e., the difference between the time when the container occupies the I/O, SO_c, and the time when it arrives at the block r_c.

In the case of containers to be retrieved, $c \in \mathcal{C}_3 \bigcup \mathcal{C}_4$, the delay is measured as the difference between the time when the vehicle (AGV or truck) takes the container and leaves the I/O point, FO_c, and the time when the vehicle arrives at the block, d_c and e_c respectively. For containers that are to be retrieved and loaded onto a vessel, $c \in \mathcal{C}_3$, congestion is caused by blocking an I/O point by anticipating the exit of the container, E_c. For containers $c \in \mathcal{C}_4$, congestion is measured by $SO_c - e_c$ for a reason similar to that of containers to be stored.

Block 1: *Assignment of I/O to containers*

$$\sum_{i\in\mathcal{IO}^{\int(\rfloor)}} Z_{il_c}^c = 1 \qquad \forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in \quad (3)$$

$$\sum_{j\in\mathcal{IO}^{\int(\rfloor)}} Z_{l_cj}^c = 1 \qquad \forall c \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle \quad (4)$$

Constraints (3) and (4) force each container to be assigned to a single I/O point.

Block 2: *Crane route from Ini to Fin*

$$\sum_{c\in\mathcal{C}_\infty\cup\mathcal{C}_\in}\sum_{j\in\mathcal{IO}^{\int(\rfloor)}} X_{Ini\,j}^c + \sum_{c\in\mathcal{C}_\ni\cup\mathcal{C}_\triangle} X_{Ini\,l_c}^c = 1 \quad (5)$$

$$\sum_{c\in\mathcal{C}_\infty\cup\mathcal{C}_\in} X_{l_c\,Fin}^c + \sum_{c\in\mathcal{C}_\ni\cup\mathcal{C}_\triangle}\sum_{i\in\mathcal{IO}^{\int(\rfloor)}} X_{i\,Fin}^c = 1 \quad (6)$$

$$\sum_{i\in\mathcal{IO}^{\int(\rfloor)}} Z_{il_c}^c = X_{l_c\,Fin}^c + \sum_{c'\in\mathcal{C}_\infty\cup\mathcal{C}_\in, c'\neq c}\sum_{i\in\mathcal{IO}^{\int(\rfloor')}} X_{l_ci}^{cc'} + \sum_{c'\in\mathcal{C}_\ni\cup\mathcal{C}_\triangle} X_{l_cl_{c'}}^{cc'}$$
$$\forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in \quad (7)$$

$$X_{Ini\,l_c}^c + \sum_{c'\in\mathcal{C}_\infty\cup\mathcal{C}_\in} X_{l_{c'}l_c}^{c'c} + \sum_{c'\in\mathcal{C}_\ni\cup\mathcal{C}_\triangle, c'\neq c}\sum_{i\in\mathcal{IO}^{\int(\rfloor')}} X_{il_c}^{c'c} = \sum_{j\in\mathcal{IO}^{\int(\rfloor)}} Z_{l_cj}^c$$
$$\forall c \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle \quad (8)$$

$$X_{Ini\,v}^c + \sum_{c'\in\mathcal{C}_\infty\cup\mathcal{C}_\in, c'\neq c} X_{l_{c'}v}^{c'c} + \sum_{c'\in\mathcal{C}_\ni\cup\mathcal{C}_\triangle}\sum_{i\in\mathcal{IO}^{\int(\rfloor')}} X_{iv}^{c'c} = Z_{vl_c}^c$$
$$\forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in \text{and}\, \forall v \in \mathcal{IO}^{\int(\rfloor)} \quad (9)$$

$$Z_{l_cv}^c = X_{v\,Fin}^c + \sum_{c'\in\mathcal{C}_\infty\cup\mathcal{C}_\in}\sum_{j\in\mathcal{IO}^{\int(\rfloor')}} X_{vj}^{cc'} + \sum_{c'\in\mathcal{C}_\ni\cup\mathcal{C}_\triangle, c'\neq c} X_{vl_{c'}}^{cc'}$$
$$\forall c \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle \text{and}\, \forall v \in \mathcal{IO}^{\int(\rfloor)} \quad (10)$$

Constraint (5) forces the crane to leave its initial position *Ini* and constraint (6) indicates that the crane must finish its route at some point. The continuity of the crane's route is guaranteed by the families of constraints (7), (8), (9) and (10). The first two families are associated with visits to the container location and the other two with visits to the corresponding I/O point.

Block 3: *Definition of Y-variables: precedence among containers*

$$Y_{cc'} + Y_{c'c} = 1 \qquad \forall c, c' \in \mathcal{C} \quad (11)$$

$$\sum_{v\in\mathcal{IO}^{\int(\rfloor')}} X_{l_cv}^{cc'} \leq Y_{cc'} \qquad \forall c, c' \in \mathcal{C}_\infty \cup \mathcal{C}_\in \quad (12)$$

$$X_{l_cl_{c'}}^{cc'} \leq Y_{cc'} \qquad \forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in, c' \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle \quad (13)$$

$$\sum_{i\in\mathcal{IO}^{\int(\rfloor)}}\sum_{j\in\mathcal{IO}^{\int(\rfloor')}} X_{ij}^{cc'} \leq Y_{cc'} \qquad \forall c \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle, c' \in \mathcal{C}_\infty \cup \mathcal{C}_\in \quad (14)$$

$$\sum_{v\in\mathcal{IO}^{\int(\rfloor)}} X_{vl_c}^{cc'} \leq Y_{cc'} \qquad \forall c, c' \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle \quad (15)$$

$$Y_{cc''} + Y_{c''c'} \leq Y_{cc'} + 1 \qquad \forall c, c', c'' \in \mathcal{C} \quad (16)$$

For each pair of containers, constraints (11) ensure that one of them precedes the other. Constraints (12), (13), (14) and (15) relate the empty movements between the transport of two containers and the Y-variables. The correct value of all Y-variables is assured, by transitivity, with constraints (16).

Block 4: *Time of each movement*

$$FC_c = SC_c + \sum_{v\in\mathcal{IO}^{\int(\rfloor)}} t_{vl_c} Z_{vl_c}^c \qquad \forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in \quad (17)$$

$$FC_c = SC_c + \sum_{v\in\mathcal{IO}^{\int(\rfloor)}} t_{l_cv} Z_{l_cv}^c \qquad \forall c \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle \quad (18)$$

$$T_{Ini} + \sum_{c\in\mathcal{C}_\infty\cup\mathcal{C}_\in}\sum_{j\in\mathcal{IO}^{\int(\rfloor)}} t_{Ini\,j} X_{Ini\,j}^c + \sum_{c\in\mathcal{C}_\ni\cup\mathcal{C}_\triangle} t_{Ini\,l_c} X_{Ini\,l_c}^c \leq SC_{c'}$$
$$\forall c' \in \mathcal{C} \quad (19)$$

$$FC_c + \sum_{v\in\mathcal{IO}^{\int(\rfloor')}} t_{l_cv} X_{l_cv}^{cc'} - SC_{c'} \leq M\left(1 - \sum_{v\in\mathcal{IO}^{\int(\rfloor')}} X_{l_cv}^{cc'}\right)$$
$$\forall c \in \mathcal{C}_\rangle, c' \in \mathcal{C}_|, i, j \in \{1, 2\} \quad (20)$$

$$FC_c + t_{l_cl_{c'}} X_{l_cl_{c'}}^{cc'} - SC_{c'} \leq M\left(1 - X_{l_cl_{c'}}^{cc'}\right)$$
$$\forall c \in \mathcal{C}_\rangle, c' \in \mathcal{C}_|, i \in \{1, 2\}, j \in \{3, 4\} \quad (21)$$

$$FC_c + \sum_{i\in\mathcal{IO}^{\int(\rfloor)}}\sum_{j\in\mathcal{IO}^{\int(\rfloor')}} t_{ij} X_{ij}^{cc'} - SC_{c'} \leq M\left(1 - \sum_{i\in\mathcal{IO}^{\int(\rfloor)}}\sum_{j\in\mathcal{IO}^{\int(\rfloor')}} X_{ij}^{cc'}\right)$$
$$\forall c \in \mathcal{C}_\rangle, c' \in \mathcal{C}_|, i \in \{3, 4\}, j \in \{1, 2\} \quad (22)$$

$$FC_c + \sum_{v\in\mathcal{IO}^{\int(\rfloor)}} t_{vl_c} X_{vl_c}^{cc'} - SC_{c'} \leq M\left(1 - \sum_{v\in\mathcal{IO}^{\int(\rfloor)}} X_{vl_c}^{cc'}\right)$$
$$\forall c \in \mathcal{C}_\rangle, c' \in \mathcal{C}_|, i, j \in \{3, 4\} \quad (23)$$

Constraints (17) and (18) relate the starting and the finishing time of the movement of the crane holding each container. Constraint (19) ensures that the starting time of the movement of the crane holding any container is greater than or equal to the finishing time of the first crane empty movement. Constraints (20), (21), (22) and (23) relate the end of the movement of one container with the start of the next.

Block 5: *Some relations between SO_c, FO_c, SC_c and FC_c*

$$SO_c \le SC_c \qquad \forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in \tag{24}$$

$$FO_c = SC_c \qquad \forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in \tag{25}$$

$$SO_c = FC_c \qquad \forall c \in \mathcal{C}_\ni \tag{26}$$

$$SO_c \le FC_c \qquad \forall c \in \mathcal{C}_\triangle \tag{27}$$

$$FC_c \le FO_c \qquad \forall c \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle \tag{28}$$

For containers of types 1 and 2, (24) guarantee that the container will be left in one I/O point before the crane picks it up, whereas (25) ensure that the occupation of the I/O point will end when the crane picks it up. For containers of type 3, (26) guarantee that the occupation of the I/O point starts when the crane leaves the container on it, and for those of type 4, (26) state that the occupation of the I/O point by a truck starts before the crane completes the transport of the container. For containers of types 3 and 4, (28) ensure that the occupation of the I/O point will end after the crane leaves the container on it.

Block 6: *Availability of the containers*

$$SO_c \ge r_c \qquad \forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in \tag{29}$$

$$FO_c \ge d_c \qquad \forall c \in \mathcal{C}_\ni \tag{30}$$

$$SO_c \ge e_c \qquad \forall c \in \mathcal{C}_\triangle \tag{31}$$

$$d_c - SO_c = E_c - T_c \qquad \forall c \in \mathcal{C}_\ni \tag{32}$$

Constraints (29) ensure that the beginning of the occupancy of an I/O point by a container of types 1 or 2 is later than its release date. Constraints (30) ensure that the end of the occupancy of an I/O point by a container of type 3 is greater than or equal to its due time. Constraints (31) guarantee that the beginning of the occupancy of an I/O point by a container of type 4 is later than its earliest time. Finally, constraints (32) establish whether a container of type 3 is exported before (earliness) or after (tardiness) its due time.

Block 7: *Monotony of crane times and occupancy times*

$$FC_c - SC_{c'} \le M(1 - Y_{cc'}) \quad \forall c, c' \in \mathcal{C}, c \ne c' \tag{33}$$

$$FO_c - SO_{c'} \le M(1 - Y_{cc'}) + M\big(2 - Z^c_{ij} - Z^{c'}_{i'j'}\big)$$

$$\forall c, c' \in \mathcal{C}, c \ne c', \text{ such that } io(Z^c_{ij}) = io\big(Z^{c'}_{i'j'}\big)$$

$$\text{where } io(Z^c_{ij}) = \begin{cases} i & \text{if } c \in \mathcal{C}_1 \cup \mathcal{C}_2 \\ j & \text{if } c \in \mathcal{C}_3 \cup \mathcal{C}_4 \end{cases} \tag{34}$$

Let us assume that container c precedes container c' in the sequence of movements of the crane, i.e., $Y_{cc'} = 1$. Constraints (33) establish that the end of the crane movement for c is prior to the beginning of the crane movement for c' and constraints (34) ensure that if both containers have been assigned to the same I/O point, the end of the occupancy for c precedes the beginning of the occupancy for c'.

Block 8: *Variable domain*

$$T_{Ini} \ge 0$$

$$SC_c,\ FC_c,\ SO_c,\ FO_c \ge 0 \qquad \forall c \in \mathcal{C}$$

$$E_c,\ T_c \ge 0 \qquad \forall c \in \mathcal{C}_\ni$$

$$Z_{vI_c},\ X^c_{Ini\,v},\ X^c_{I_c\,Fin} \in \{0,1\} \qquad \forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in, v \in \mathcal{IO}^{\int(\rfloor)}$$

$$Z_{I_c v},\ X^c_{Ini\,I_c},\ X^c_{v\,Fin} \in \{0,1\} \qquad \forall c \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle, v \in \mathcal{IO}^{\int(\rfloor)}$$

$$X^{cc'}_{ij} \in \{0,1\} \qquad \forall c, c' \in \mathcal{C}$$

$$Y_{cc'} \in \{0,1\} \qquad \forall c, c' \in \mathcal{C}$$

4.2. Scheduling model

This model is based on considering our problem as a scheduling problem. The crane is considered as a machine that must process several jobs, the movement of containers. For each container, the model provides the position in the job sequence where the crane moves the container, as well as the I/O point from which the container is imported into the block (for types 1 and 2) or to which it is exported (for types 3 and 4). Analogously to the routing model, it must be ensured that two containers do not simultaneously occupy the same I/O point.

Variables

$\forall c \in \mathcal{C}, \forall j \in \mathcal{IO}^{\int(\rfloor)}, \forall k \in \mathcal{K}$ (positions in the sequence, $1, 2, \ldots, K = |\mathcal{C}|$)

$$X_{ckj} = \begin{cases} 1, & \text{if the container } c \text{ is processed at the position } k, \\ & \text{setting its} \\ & \text{non-fixed origin/destination to } j \\ 0, & \text{otherwise} \end{cases}$$

SC_c	Starting time of the movement of $c \in \mathcal{C}$
FC_c	Finishing time of the movement of $c \in \mathcal{C}$
SO_c	Starting time of the occupancy of the input/output point used by $c \in \mathcal{C}$
FO_c	Finishing time of the occupancy of the input/output point used by $c \in \mathcal{C}$
T_{Ini}	Starting time of the movement of the crane
E_c, T_c	Earliness/tardiness, with respect to d_c, in exporting $c \in \mathcal{C}_3$ to an I/O point.

The scheduling model is:

Objective function:

$$\begin{aligned} \text{Min} \ & \sum_{c \in \mathcal{C}_\infty} w^R_1(FC_c - r_c) + \sum_{c \in \mathcal{C}_\in} w^R_2(FC_c - r_c) + \sum_{c \in \mathcal{C}_\ni} w^R_3(FO_c - d_c) \\ & + \sum_{c \in \mathcal{C}_\triangle} w^R_4(FO_c - e_c) + \sum_{c \in \mathcal{C}_\infty} w^C_1(SO_c - r_c) + \sum_{c \in \mathcal{C}_\in} w^C_2(SO_c - r_c) \\ & + \sum_{c \in \mathcal{C}_\ni} w^E_3 E_c + \sum_{c \in \mathcal{C}_\triangle} w^C_4(SO_c - e_c) \end{aligned} \tag{35}$$

The objective function is exactly the same as the one used in the routing model.

Block 1: *Assignment of position in the sequence and I/O to containers*

$$\sum_{k \in K} \sum_{j \in \mathcal{IO}^{\int(\rfloor)}} X_{ckj} = 1 \qquad \forall c \in \mathcal{C} \tag{36}$$

$$\sum_{c\in\mathcal{C}} \sum_{j\in\mathcal{IO}^{\int(\rfloor)}} X_{ckj} = 1 \qquad \forall k \in \mathcal{K} \tag{37}$$

Constraints (36) ensure that all containers are assigned to one position in the sequence and to one I/O point, whereas constraints (37) guarantee that each position in the sequence is assigned to one container.

Block 2: *Time of each movement*

$$SC_c \geq T_{Ini} + \sum_{j\in\mathcal{IO}^{\int(\rfloor)}} t_{Ini\,j} X_{c1j} \qquad \forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in \tag{38}$$

$$SC_c \geq T_{Ini} + \sum_{j\in\mathcal{IO}^{\int(\rfloor)}} t_{Ini\,l_c} X_{c1j} \qquad \forall c \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle \tag{39}$$

$$FC_c = SC_c + \sum_{k\in K} \sum_{j\in\mathcal{IO}^{\int(\rfloor)}} t_{jl_c} X_{ckj} \qquad \forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in \tag{40}$$

$$FC_c = SC_c + \sum_{k\in K} \sum_{j\in\mathcal{IO}^{\int(\rfloor)}} t_{l_c j} X_{ckj} \qquad \forall c \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle \tag{41}$$

Constraints (38) and (39) ensure that the start of the movement of any container is later than the first movement of the crane, empty, from its initial position. Constraints (40) and (41) establish the relationship between the starting time and the finishing time of the container movement.

Block 3: *Common constraints with the routing model*

$$SO_c \leq SC_c \qquad \forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in \tag{42}$$

$$FO_c = SC_c \qquad \forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in \tag{43}$$

$$SO_c = FC_c \qquad \forall c \in \mathcal{C}_\ni \tag{44}$$

$$SO_c \leq FC_c \qquad \forall c \in \mathcal{C}_\triangle \tag{45}$$

$$FO_c \geq FC_c \qquad \forall c \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle \tag{46}$$

$$SO_c \geq r_c \qquad \forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in \tag{47}$$

$$FO_c \geq d_c \qquad \forall c \in \mathcal{C}_\ni \tag{48}$$

$$SO_c \geq e_c \qquad \forall c \in \mathcal{C}_\triangle \tag{49}$$

$$d_c - SO_c = E_c - T_c \qquad \forall c \in \mathcal{C}_\ni \tag{50}$$

Constraints (42) to (50) are exactly the same as constraints (24) to (32) of the routing model.

Block 4: *Monotony of crane and occupancy times*

$$SC_{c'} \geq FC_c + t_{jl_{c'}} - M\left(2 - X_{c,k-1,j} - \sum_{r\in\mathcal{IO}^{\int(\rfloor')}} X_{c'kr}\right)$$
$$\forall c, c' \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle, \forall k \in \mathcal{K}, \forall j \in \mathcal{IO}^{\int(\rfloor)} \tag{51}$$

$$SC_{c'} \geq FC_c + t_{l_c j} - M\left(2 - \sum_{r\in\mathcal{IO}^{\int(\rfloor)}} X_{c,k-1,r} - X_{c'kj}\right)$$
$$\forall c, c' \in \mathcal{C}_\infty \cup \mathcal{C}_\in, \forall k \in \mathcal{K}, \forall j \in \mathcal{IO}^{\int(\rfloor')} \tag{52}$$

$$SC_{c'} \geq FC_c + t_{l_c l_{c'}} - M\left(2 - \sum_{r\in\mathcal{IO}^{\int(\rfloor)}} X_{c,k-1,r} - \sum_{r\in\mathcal{IO}^{\int(\rfloor')}} X_{c'kr}\right)$$
$$\forall c \in \mathcal{C}_\infty \cup \mathcal{C}_\in, \forall c' \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle, \forall k \in \mathcal{K} \tag{53}$$

$$SC_{c'} \geq FC_c + t_{jr} - M\big(2 - X_{c,k-1,j} - X_{c'kr}\big)$$
$$\forall c \in \mathcal{C}_\ni \cup \mathcal{C}_\triangle, \forall c' \in \mathcal{C}_\infty \cup \mathcal{C}_\in, \forall k \in \mathcal{K}, \forall j \in \mathcal{IO}^{\int(\rfloor)}, \forall r \in \mathcal{IO}^{\int(\rfloor')} \tag{54}$$

$$SO_{c'} \geq FO_c - M\left(2 - \sum_{\tau=1}^{k-1} X_{c\tau j} - \sum_{\tau=k}^{K} X_{c'\tau j}\right)$$
$$\forall c, c' \in \mathcal{C}_\infty \cup \mathcal{C}_\ni, \forall k \in \mathcal{K}, \forall j \in \mathcal{IO}^{\mathcal{S}} \tag{55}$$

$$SO_{c'} \geq FO_c - M\left(2 - \sum_{\tau=1}^{k-1} X_{c\tau j} - \sum_{\tau=k}^{K} X_{c'\tau j}\right)$$
$$\forall c, c' \in \mathcal{C}_\in \cup \mathcal{C}_\triangle, \forall k \in \mathcal{K}, \forall j \in \mathcal{IO}^{\mathcal{L}} \tag{56}$$

Constraints (51) to (54) establish that for two containers in consecutive sequence positions, the end of the crane movement with the first container in the sequence plus the time of the empty crane movement between the destination of the first container and the origin of the second container must be less than the beginning of the crane movement of the second container. Finally, (55) and (56) ensure that for two containers that have been assigned to the same I/O point, the end of the occupancy by the first container of the pair in the sequence is prior to the beginning of the occupancy by the second container.

Block 5: *Variable domain*

$$T_{Ini} \geq 0$$

$$SC_c,\ FC_c, SO_c,\ FO_c \geq 0 \qquad \forall c \in \mathcal{C}$$

$$E_c,\ T_c \geq 0 \qquad \forall c \in \mathcal{C}_\ni$$

$$X_{ckj} \in \{0, 1\} \qquad \forall c \in \mathcal{C}, \forall j \in \mathcal{IO}^{\int(\rfloor)}, \forall k \in \mathcal{K}$$

5. Heuristics

In this section three constructive rules and a local search method are proposed. Two decisions are involved: first, the order in which the containers are considered and, second, the I/O point assignment for each container.

5.1. Constructive algorithms

All three constructive algorithms consist of a first phase where the sequence of the containers is determined and a second phase where the I/O point assignment and timing are calculated. The second phase is common to all algorithms.

Table 1
Data for the MTPR example.

	c_1	c_2	c_3	c_4
Type	C_1	C_2	C_3	C_4
Time Parameter	3	2	1	4

Table 2
Data for the NCR example.

	c_1	c_2	c_3
Type	C_3	C_1	C_4
Time Parameter	3	2	1
w_i^R	3	4	2
Location(x,y,z)	(1,3,2)	(2,5,3)	(4,4,4)

5.1.1. First phase: Sequence

Three options are implemented:

- Time Parameter Rule (TPR): the containers are sequenced in non-decreasing order of their time parameter. An example with 4 containers (denoted as c_1 to c_4) is given in Table 1. According to the value of the time parameter, the sequence after applying the TPR would be $\{c_3, c_2, c_1, c_4\}$.
- Modified Time Parameter Rule (MTPR): After the application of TPR, the sequence can be modified according to the type of container. The main idea is to minimise the total distance of the empty crane movements. Therefore, if the crane movement ends in the yard, the next one should also start in the yard. Similarly, if the crane ends a movement on the sea (or land) side, the next one should start on the same side. Continuing with the example, from the sequence provided by TPR ($\{c_3, c_2, c_1, c_4\}$), container c_3 is of type C_3 (Yard to Sea). Therefore, the next container should be type C_1 (Sea to Yard). Container c_1 is of type C_1, so the new sequence is $\{c_3, c_1, c_2, c_4\}$. After container c_1, the next container should be of type C_4 (Yard to Land). Container c_4 is of type C_4, then the new sequence is $\{c_3, c_1, c_4, c_2\}$. If there are no containers of the required type, the next container is not changed from the original sequence.
- Nearest Container Rule (NCR): The main idea is that starting from the initial position of the crane, the sequence is constructed by moving the crane to the nearest available container according to a priority value (η_c). The priority value is calculated as the total time divided by the delay weight of the container c of type i according to equation (57). The crane will move towards the container with the minimum η_c.

$$\eta_c = \frac{TT_c}{w_i^R} \tag{57}$$

The total time for container c (TT_c) is computed as the time between the destination where the crane delivered the previous container and the destination to which the crane has to travel to deliver container c. To calculate this time, an I/O point is needed for each container and the I/O point is considered to be centrally located on the sea (or land) side. An example with 3 containers is provided in Table 2, where the last row corresponds to the known location according to the container type. Container c_1 is of type C_3 (yard to sea), so its initial location inside the yard is known. Container c_2 is of type C_1 (sea to yard), which means that its destination in the yard is known. Finally, container c_3 is of type C_4 (yard to land), so its initial location is known. Note that neither container c_1 nor container c_3 need reshuffling movements. The initial position of the crane (1,2,5) and the location of the sea (or land) I/O points, (1,0,1) and (1,43,2) respectively, need to be established. All speeds are set to 1 for simplicity. Table 3 shows the total time of each container, calculated as the time the crane takes to reach the initial location of the container plus the time it takes to move the container from its initial location to its destination.

According to Table 3, TT_{c_1} is computed as follows:
$TT_{c_1} = (5-5) + \max\{|1-1|, |3-2|\} + (5-2) + (5-2) + \max\{|1-1|, |3-0|\} + (5-1) = 4 + 10 = 14$,
where the first part corresponds to the movement from the initial location of the crane to the initial position of the container and the second part corresponds to the movement of the crane from the initial location of the container to its final position. Note that the crane has to go down in the first part of the movement and up in the second part. According to Table 3, the crane has to move from its initial location to the initial location of container c_2, since it has the minimum value of TT_c/w_i^R. Now, the initial location of the crane is the final location of container c_2 (2,5,3). The crane can move to the initial location of container c_1 or to the initial location of container c_3. Table 4 shows the results of the total distance for each option. According to the results, the crane should move to the initial location of container c_1. Finally, the crane should move from the final location of container c_1 to the initial location of container c_3. Therefore, the sequence for the crane is $\{c_2, c_1, c_3\}$.

Table 3
Selection of the first container.

	TT_{c_1}	TT_{c_2}	TT_{c_3}
Initial location: (1,2,5)	$4+10=14$	$6+11=17$	$4+43=47$
TT_c/w_i^R	$14/3=4.7$	$17/4=4.3$	$47/2=23.5$

Table 4
Selection of the second container.

	TT_{c_1}	TT_{c_3}
Initial location: (2,5,3)	$7+10=17$	$5+43=48$
TT_c/w_i^R	$17/3=5.7$	$48/2=24$

5.1.2. Second phase: Timing and I/O point assignment

Once the sequence of containers has been obtained, a second step is applied to obtain the timing, that is, the start and end time of each crane movement. In addition, it is necessary to assign an I/O point from the set of available I/O points. Continuing with the example provided by the NCR rule, the final sequence for the crane was $\{c_2, c_1, c_3\}$. Two I/O points are assumed on each side (sea and land) with the following locations: (1,0,1) and (8,0,1) for the seaside and (1,43,2) and (6,43,2) for the landside. The following steps are carried out for each container:

- **Step 1: Creating an ordered list of I/O points.** The main idea is to consider not only the current container but also the next one in order to get the best I/O point for the current container. A list of I/O points is created for each container c (denoted as $L_{IO,c}$), ordered from the best to the worst I/O option. Fig. 2 shows the different options for determining the best I/O point as a function of distance. The highlighted box in each subfigure corresponds to the I/O point to be assigned. The procedure starts from a known initial location for the crane (denoted as Ini). The crane has to move from Ini to the origin of the container (denoted as O_c), that is, to the yard or to the sea/land side, depending on the container type. Therefore, this first time (distance) is denoted as $T_1(Ini, O_c)$. Afterwards, the crane has to move from O_c to the container destination (denoted as D_c). This second time is

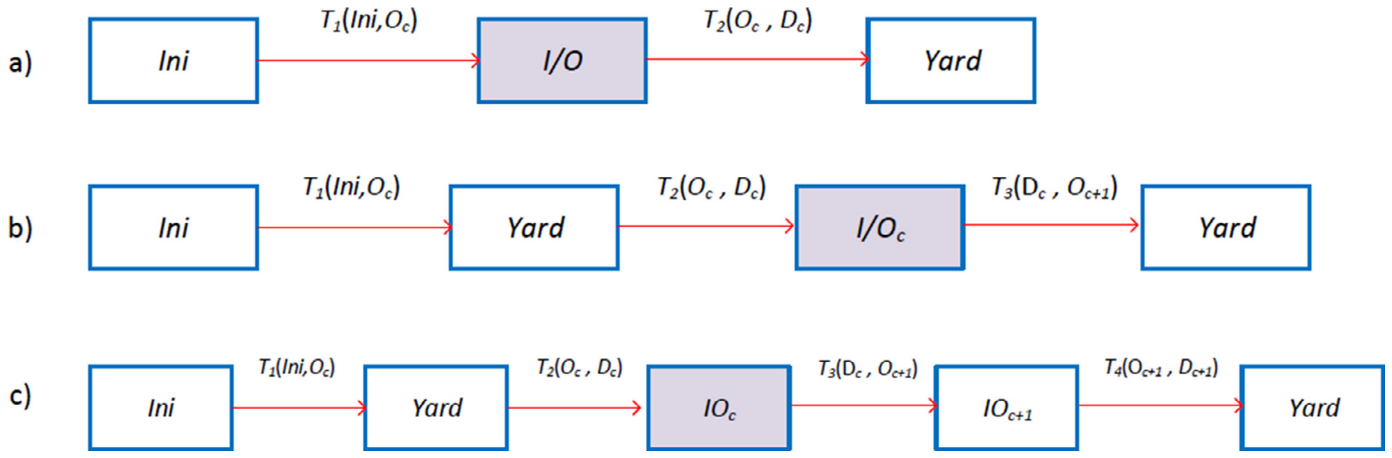


Fig. 2. Different situations in which the I/O point has to be assigned.

denoted as $T_2(O_c, D_c)$. For each container, an I/O point is required depending on the container type. For C_1 and C_2 types, the I/O point is the origin (O_c) of the container. In this case, the best I/O point will be the one that minimises the total time $TotalTime_c = T_1(Ini, O_c)$ + $T_2(O_c, D_c)$ (Fig. 2(a)). However, for types C_3 and C_4, the I/O point is the destination of the container (D_c). In this case, the best I/O point will be affected by the next container. If the origin of the next container is in the yard, the best I/O point will be the one that minimises the total time $TotalTime_c = T_1(Ini, O_c) + T_2(O_c, D_c)$ + $T_3(D_c, O_{c+1})$ (Fig. 2(b)). Nevertheless, if the origin of the next container is an I/O point, the best I/O point (for the current container) will be the one that minimises the total time $TotalTime_c = T_1(Ini, O_c)$ + $T_2(O_c, D_c)$ + $T_3(D_c, O_{c+1})$ + $T_4(O_{c+1}, D_{c+1})$ (Fig. 2(c)).

- **Step 2: Computation of start and finish times.** From the list provided by the previous step, the best I/O point is assigned to the container, if available. If not, the next I/O point is assigned and so on. Therefore, the I/O point start and finish times are calculated as the time at which the I/O point is assigned and the time at which it is released, respectively. If no I/O points are available, the first one to be released is assigned. The crane start and finish times are updated.

Once all containers have been considered, the objective function (Eq. (35)) is calculated. Algorithm 1 shows the pseudocode of the procedure to build a feasible solution from a sequence.

Algorithm 1: Timing and input/output assignment.

```
1  $S$: Solution and $S_e$ : Sequence according to any rule;
2  $S := S_e$;
3  for $c$ in $S$ do
4      Create the ordered list of I/O points for container $c$, $L_{IO,c}$
       according to Figure 2;
5      for $i$ in $L_{IO,c}$ do
6          if $L_{IO,c}[i]$ is free then
7              Assign $L_{IO,c}[i]$ to container $c$;
8              Compute $SO_c$, $FO_c$, $SC_c$, $FC_c$;
9              Update Objective Function Value;
10             break;
```

Following the example in Table 2, after the application of the NCR rule the sequence is $\{c_2, c_1, c_3\}$. The initial location of the crane is (1,2,5). Container c_2 is of type C_1, so its origin is a seaside I/O point. Two I/O points are available, so the one that minimises the total time will be chosen. Assigning the first I/O point located at (1,0,1), $TotalTime_{c_2} = T_1((1,2,5),(1,0,1)) + T_2((1,0,1),(2,5,3)) = (0+2+4)+(4+5+2) = 17$. Assigning the other I/O point located at (8,0,1), $TotalTime_{c_2} = T_1((1,2,5),(8,0,1)) + T_2((8,0,1),(2,5,3)) = (0+7+4)+(4+6+2) = 23$. The ordered list for container c_2 will be $L_{IO,2} = \{IO_{s1}, IO_{s2}\}$. Both I/O points are available (it is the first container of the sequence), so the first one will be assigned. As the time parameter of container c_2 is 2 (Table 2) and the time taken by the crane from its initial location to the origin of c_2 is 6, when the crane arrives there the container has already arrived. In this case, the start time of the I/O point is 2 and the finish time is 0+2+4=6, since the crane has to go down (from tier 5 to tier 1) to pick up the container. The crane start time for container c_2 is 6 and the finish time is 6 + 4 + 5 + 2 = 17.

The next container in the sequence is c_1 of type C_3 (yard to sea). The location of the crane is (2,5,3) and it will move to (1,3,2) to pick it up. The next in the sequence, c_3, is of type C_4 (yard to land), so the crane will take container c_1 to a seaside I/O point to be assigned and then move back from there to the position of container c_3 in the yard. If the first I/O point is assigned, $TotalTime_{c_1} = 17 + T_1((2,5,3),(1,3,2)) + T_2((1,3,2),(1,0,1)) + T_3((1,0,1),(4,4,4)) = 17 + (2+2+3) + (3+3+4) + (4+4+1) = 43$. If the second I/O point is assigned, $TotalTime_{c_1} = 17 + T_1((2,5,3),(1,3,2)) + T_2((1,3,2),(8,0,1)) + T_3((8,0,1),(4,4,4)) = 17 + (2+2+3) + (3+7+4) + (4+4+1) = 47$. Therefore, the best I/O point for container c_1 is the first one located at (1,0,1). This I/O point is available from time 6, so it is assigned to container c_1. The crane will arrive at the I/O point at time 17 + (2 + 2 + 3) + (3 + 3 + 4) = 34, but the container should be there at time 3, so there is a tardiness penalty. As soon as the crane arrives at the I/O point it will drop off the container and the container will be collected, so the I/O point will only be occupied in period 34.

The crane start time for container c_1 is 17 + (2 + 2 + 3) = 24 (considering the finish time of the previous container) and the finish time is 24 + (3 + 3 + 4) = 34. The same procedure is applied to calculate the crane start and finish times for container c_3. The value of the objective function is updated considering the start and finish times calculated above according to expression (2).

5.2. Local search

Local search methods are widely used in optimisation problems. The objective is to introduce moves in the current solution to obtain a new and better solution. Insertion neighbourhood is one of the most widely used local search procedures for problems where the solution representation is given by a permutation or list of jobs (containers in our case). The local search procedure is based on a first-improvement strategy, that is, each container of the solution is inserted in every position in the sequence while there is no improvement of the objective function, starting from the first one. If an insertion movement obtains an improvement in the objective function value, the container is finally located in that position (first improvement strategy). Due to the computational time needed, the procedure is iterative until a stopping criterion is met. Specifically, a maximum CPU time of one second per container is allowed, so that larger instances will have more time to run the local search. In Algorithm 2 the local search procedure is shown.

Algorithm 2: Local search (insertion neighbourhood).

```
$S$: Best Solution ;
$S'$: New Solution ;
$BestValue$ := ObjectiveFunctionValue($S$);
while $Time \leq MaximumTime$ do
    for $c_1$ in $S$ do
        for $c_2$ in $S$ do
            if $c_1 \neq c_2$ then
                $S' := S.InsertionMovement(c1, c2)$;
                $NewValue$:= ObjectiveFunctionValue($S'$);
                if $NewValue < BestValue$ then
                    $BestValue := NewValue$;
                    $S := S'$ ;
                    break ;
```

6. Computational results

In this section, computational results are provided for an extensive benchmark of instances, consisting of small, medium, and large instances. First, the proposed benchmark is explained in detail and then it is used to compare the proposed methods.

6.1. Benchmark of instances

Three sets of instances are generated for the benchmarking. Depending on the number of container requests, the instance is classified as small, medium, or large. The following values are considered: small set, $n \in \{5, 10\}$ container requests; medium set, $n \in \{15, 20, 30, 40\}$ container requests, and large set, $n \in \{50, 100, 150, 200\}$ container requests. Additional information is needed on the characteristics of the yard, related to the number of cranes, the number of rows (R), bays (B), tiers (T), and I/O points on both sides, land and sea, and the speed of the yard crane. In this case, one yard crane is considered, $R = 10$ rows, $B = 42$ bays, $T = 4$ tiers, 6 I/O points on the landside and 10 I/O points on the seaside. These values are based on actual size yards (Gharehgozli et al., 2019; Gharehgozli et al., 2014). An additional small set (small tight) is also generated, in which there are only two I/O points on the seaside and one on the landside. In this way, the effect of the congestion can also be evaluated in small instances. Regarding the speed of the yard crane, three different speeds can be considered: gantry, trolley and hoisting, depending on the crane movement. In this case, a speed of 1 is considered for all three speeds as in Speer & Fischer (2017)). Nevertheless, the speeds can be different, as in Gharehgozli et al. (2019), and that would only involve updating the calculations in (1). Moreover, in some cases, a reshuffling may be needed, so 2 unit times per reshuffling are also included in the instance. All information related to the yard characteristics is the same for all instances, regardless of size.

With respect to container requests, the following information is needed:

- Type of container movement: there are 4 types of container movements (see Section 3). A number between 1 and 4 is randomly generated.
- Location of the container: for type 1 and 2 containers, the final position within the yard is generated. For type 3 and 4 containers, the initial location within the yard is also generated. The locations are generated as (x, y, z) coordinates. All values are randomly generated: x varies from 1 to R (10 in our case), y varies from 1 to B (42 in our case), and z varies from 1 to T (4 in our case).
- Number of reshuffles: only types C_3 and C_4 can involve reshuffling. A number between 0 and $T - z$ is randomly generated. For example, if a container of type C_3 is located at (4,25,2), then the number of reshuffles can vary between 0 and $4 - 2 = 2$.
- Time parameter: each container is related to a time parameter, depending on the type. Types 1 and 2 have a release time (r_j), type 3 has a due time (d_j) and type 4 has a time parameter related to the arrival time of the truck at the I/O point on landside (e_j). These parameters are generated following the expression:
  $r_j = random(0, \rho * RoundTrip)$
  $d_j(e_j) = random(0, 0.4 * RoundTrip)$,
  where $RoundTrip = n \times B$, that is, a high value assuming that the crane goes through the yard for each container. The ρ parameter for the release time is set to {0.1,0.4,0.7}. In this way the effect of the arrival of containers can be analysed.
- Weights: two types of weights are generated, related to the container (w_i^R) and the congestion (w_i^C) respectively. Containers of the same type will have the same weights in both cases. For container weights (w_i^R), two sets are generated: in the first set, the weights are set to 1 for all types of container. In the second set, the weights of containers related to seaside, that is, types C_1 and C_3, are set to 3, while those related to the landside (types C_2 and C_4) are set to 1. For congestion weights (w_i^C), the same two combinations are generated. The set where all weights are 1 is denoted as *Equal* while the set with different weights depending on the container type is denoted as *Non Equal*.

For each instance size, three values of ρ and two ways of generating the weights are considered and 10 replicates are generated: that is, $10 \times 3 \times 2 = 60$ combinations. In total, there are 12 size combinations (2 for small, 2 for small tight, 4 for medium and 4 for large). Thus, in total, there are 720 instances.

Information on the location of the I/O points on both sides is also generated. For the seaside, 10 I/O points are given. Then, for each one, the $(x, 0, 1)$ coordinates are generated, where $x = 1, 2 \cdots, 10$. Note that in this case, an extra 0 bay is considered, just for the I/O points on the seaside. As for z, the value is 1 for all cases. For the landside, 6 I/O points are available. As before, each one has coordinates $(x, B + 1, 2)$, where $x = 2, 3, \cdots, 7$. In this case, an extra bay, 43 in our case, is considered, just for these I/O points. With respect to z, the value is 2, since the container is delivered on a truck. For the small tight set of instances, there are only 2 I/O points on the seaside whose coordinates are generated as $(x, 0, 1)$, where $x = 4, 5$. For the landside, there is only 1 I/O point in $(x, B + 1, 2)$, where $x = 4$.

The initial location of the crane also consists of coordinates (x, y, z). In this case, the z value is set to $T + 1$ (5 in our case). As for x and y, they are randomly generated between 1 and R and 0 and $B + 1$ respectively.

Table 5
Number of variables and constraints in the two models for small and small tight instances.

				Routing model		Scheduling model	
Set	n	w_j	R_j	*#var*	*#rows*	*#var*	*#rows*
Small	5	E	0.1	573.2	299.4	231.4	2161.0
			0.4	541.4	277.6	219.0	1977.6
			0.7	542.8	285.2	223.8	1999.6
		NE	0.1	555.8	294.8	217.0	2108.8
			0.4	559.8	299.7	225.2	2124.1
			0.7	492.4	285.2	217.0	1816.8
	10	E	0.1	2140.1	1549.2	843.0	19814.5
			0.4	2188.5	1583.6	879.0	20462.5
			0.7	2236.7	1599.1	862.0	21092.0
		NE	0.1	2077.7	1530.7	833.2	19131.4
			0.4	2135.8	1553.9	845.2	19830.7
			0.7	2302.4	1575.9	866.0	21463.7
Small Tight	5	E	0.1	95.6	190.0	61.4	221.0
			0.4	99.1	195.3	64.5	243.8
			0.7	95.3	193.6	61.2	232.2
		NE	0.1	93.5	189.7	59.9	215.0
			0.4	93.0	191.6	59.4	221.8
			0.7	93.8	189.5	60.4	217.3
	10	E	0.1	308.3	1183.0	189.4	1813.7
			0.4	316.1	1191.7	195.2	1938.3
			0.7	313.6	1188.6	194.6	1875.1
		NE	0.1	310.4	1189.5	191.2	1878.0
			0.4	317.2	1193.3	197.2	1967.5
			0.7	307.8	1188.1	188.8	1862.9

All the instances and the detailed results of each instance are available to the research community upon request from the authors.

6.2. Results

The proposed mathematical models and the heuristic methods (with and without local search) were tested using the 720 instances of the benchmark.

All the experiments were run on an Intel Core i7, 3.4 GHz and 20 GB RAM. Microsoft Visual Studio 2019 with C# was used to code the heuristic methods. The mathematical models were solved using CPLEX 20.10.

6.2.1. Results of the models

The two models presented in Section 4 were tested using the small and small tight sets, with 120 instances each. First, in Table 5, we compare the average number of variables and the average number of constraints in the two models. Then, in Table 6, we study the results obtained with the two models with a running time of 3600 seconds.

Table 5 shows the average size of both models in terms of the number of variables and constraints. Column n refers to the number of containers in the instance; column w_j is related to the set of weights, E for equal weights regardless of the type of container and NE for non-equal weights, depending on the type of container (1 for landside-related and 3 for seaside-related containers); column R_j is the ρ value defined to generate release times; the next columns denote, for each model, average number of variables (column *#var*) and the average number of constraints (column *#rows*) of the 10 instances with the characteristics described in columns 2, 3, and 4. The routing model contains more variables than the scheduling model, but has far fewer constraints.

The comparison of the results obtained by the two models with a maximum run time of one hour is shown in Table 6. The first four columns in Table 6 are the same as in Table 5. For each of the models, the number of optimums obtained (*#opt*), the average deviation of the lower bound with respect to the optimum (GAP_{lb}) and the average CPU time in seconds (CPU) are shown next. The value of GAP_{lb} is calculated as follows

$$GAP_{lb} = \frac{Opt - LB}{Opt} \cdot 100, \tag{58}$$

where LB is the lower bound/optimal value obtained with the model and Opt is the optimal value of the instance. For all the instances in small and small tight sets Opt was obtained by the routing model without time limit.

First, both models were able to solve all the instances with 5 containers in a few seconds, with the scheduling model needing more time than the routing model. However, for instances with 10 containers, the scheduling model was not able to solve any of the instances of the small set (regardless of the weight set) and solved 42 instances from the small tight set (21 with equal weights and the other 21 with non-equal weights). In contrast, the routing model was able to solve 101 out of 120 instances with 10 containers (45 small and 56 small tight). For no instance was the scheduling model better than the routing model.

In general, the small tight set was also easier to solve than the small set. In addition, as expected, increasing the value of ρ makes the instances easier to solve, both in terms of gap value and time required. Finally, neither of the models was able to solve any medium or large instance.

6.2.2. Results of the heuristics

The effectiveness of heuristics is measured by the relative percentage deviation (RPD), which is computed for each instance according to the standard expression:

$$\text{Relative Percentage Deviation}(RPD) = \frac{Heu_{sol} - Best_{sol}}{Best_{sol}} \cdot 100, \tag{59}$$

where Heu_{sol} is the solution obtained with a given proposed heuristic and $Best_{sol}$ is the best known solution for a given instance. For small and small tight instances, $Best_{sol}$ is the optimal solution obtained by the routing model (Section 4.1). For medium and large instances, $Best_{sol}$ is the best known solution obtained among all the heuristic methods.

Table 7 shows the number of optimal solutions obtained by the proposed heuristic methods for the small and small tight sets (120 small instances and 120 small tight instances, so that each row corresponds to a subset of 10 instances). The first four columns are the same as in Table 5 and the next columns denote the heuristic method, with and without local search, explained in Section 5. For example, TPR refers to the Time Parameter Rule and TPR_{ls} refers to the Time Parameter Rule with local search.

From the results in Table 7, it can be observed that the NCR_{ls} heuristic (Nearest Container Rule with local search) shows the better performance, being able to solve 113 instances optimally out of 240. The second best method is the TPR_{ls}, solving 101 instances. Obviously methods without local search are the ones that obtain the fewest optimal solutions, being able to solve only a few instances with 5 containers optimally. However, it can be seen that the NCR rule solves many more instances than the other methods without local search. Considering the instance size, the proposed methods with local search obtain fewer optimal solutions when the number of containers increases. For example, the NCR_{ls} method gets 14 optimal solutions for small instances with 10 containers versus 50 optimal solutions for the same set with 5 containers. As for the small tight set, the NCR_{ls} method obtains 12 optimal solutions for instances with 10 containers and 37 optimal solutions for instances with 5 containers.

Tables 8 and 9 show the RPD for small and small tight instances respectively. The structure of the tables is the same as that of Table 7. Each row contains the average RPD value for 10 instances

Table 6
Comparison of the results of the two models for small and small tight instances.

				Routing model			Scheduling model		
Set	*n*	w_j	R_j	*#opt*	GAP_{lb}	*CPU*	*#opt*	GAP_{lb}	*CPU*
Small	5	E	0.1	10	0.00	0.34	10	0.00	84.29
			0.4	10	0.00	0.32	10	0.00	50.14
			0.7	10	0.00	0.28	10	0.00	24.17
		NE	0.1	10	0.00	0.29	10	0.00	53.52
			0.4	10	0.00	0.28	10	0.00	83.94
			0.7	10	0.00	0.20	10	0.00	13.53
	10	E	0.1	2	12.98	3339.53	0	89.43	3600
			0.4	6	4.50	2453.82	0	87.74	3600
			0.7	10	0.00	697.17	0	84.91	3600
		NE	0.1	7	4.14	2321.44	0	87.23	3600
			0.4	10	0.00	1047.30	0	82.46	3600
			0.7	10	0.00	424.45	0	79.85	3600
Small Tight	5	E	0.1	10	0.00	0.13	10	0.00	0.22
			0.4	10	0.00	0.13	10	0.00	0.25
			0.7	10	0.00	0.11	10	0.00	0.20
		NE	0.1	10	0.00	0.13	10	0.00	0.12
			0.4	10	0.00	0.10	10	0.00	0.13
			0.7	10	0.00	0.10	10	0.00	0.13
	10	E	0.1	8	6.31	1970.19	7	15.31	1617.78
			0.4	10	0.00	1082.05	7	16.71	1600.89
			0.7	9	4.21	609.96	7	11.60	1461.60
		NE	0.1	10	0.00	2173.78	7	10.97	2367.61
			0.4	9	5.17	1810.43	5	24.39	2558.95
			0.7	10	0.00	421.28	9	4.96	711.65

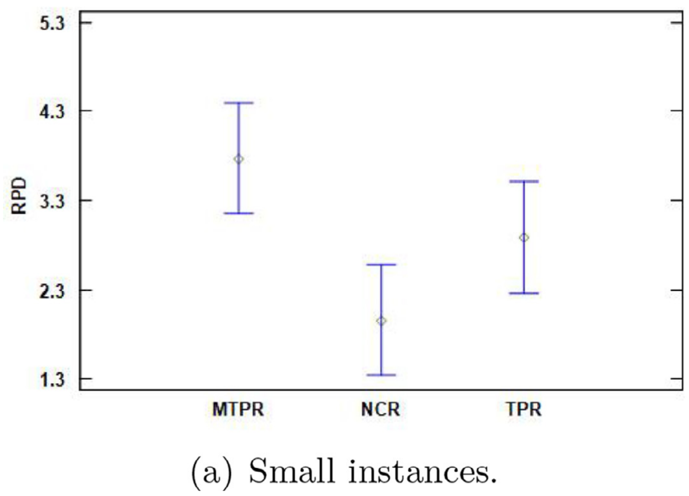
Table 7
Number of optimal solutions obtained for small and small tight instances.

Set	*n*	w_j	R_j	*TPR*	TPR_{ls}	*MTPR*	$MTPR_{ls}$	*NCR*	NCR_{ls}
Small	5	E	0.1	1	8	1	8	8	9
			0.4	0	8	1	6	2	10
			0.7	0	6	1	8	1	8
		NE	0.1	0	6	0	7	3	7
			0.4	0	8	0	7	3	9
			0.7	1	8	0	8	2	7
	10	E	0.1	0	2	0	0	0	0
			0.4	0	0	0	1	0	3
			0.7	0	1	0	1	1	1
		NE	0.1	0	3	0	1	0	2
			0.4	0	3	0	2	0	2
			0.7	0	3	0	3	0	6
Small Tight	5	E	0.1	0	3	1	3	2	7
			0.4	1	6	2	5	2	5
			0.7	0	7	0	6	2	5
		NE	0.1	0	6	0	5	0	5
			0.4	0	8	0	7	5	7
			0.7	2	7	2	7	0	8
	10	E	0.1	0	1	0	0	0	2
			0.4	0	1	0	1	0	0
			0.7	0	3	0	1	0	3
		NE	0.1	0	0	0	0	0	2
			0.4	0	2	0	1	0	2
			0.7	0	1	0	2	0	3
	Total			5	101	8	90	31	113

(the 10 replicas of each combination) for each heuristic method. As all these instances are optimally solved, the *RPD* values are the percentage deviations from the optimal solutions.

Comparing the results of the heuristic methods without local search, it can be seen that, on average, the performance of the *NCR* method is better for small and small tight instances. If the local search is included, it can be seen that the improvement for all methods is very significant in all cases. Now, the best method for all instances is NCR_{ls}, although for small tight instances the TPR_{ls} shows a similar average *RPD*. Moreover, it can be determined that the average *RPD* value for all methods with local search is much higher for small tight instances than for small instances, that is, the congestion at the I/O points affects their performance. To test whether these differences in the *RPD* values are statistically significant, an analysis of variance (ANOVA) is applied (Montgomery (2019)), once the three hypotheses of this statistical test -normality, homoscedasticity and independence of the residuals- have been tested. Figs. 3(a) and 3(b) plot the means with LSD intervals ($\alpha = 0.05$) for the algorithms with local search (small and small tight instances). Several overlapping intervals can be observed; however, there are statistically significant differences between the average *RPD* obtained by *NCR* and *MTPR* with local search (small instances).

To analyse the R_j and W_j parameters, small instances are considered, since the optimal solutions are known, and only in the methods with local search, as they perform much better than their counterparts without local search. As for the R_j parameter, Fig. 4(a) shows the plot of means with LSD intervals ($\alpha = 0.05$) for the



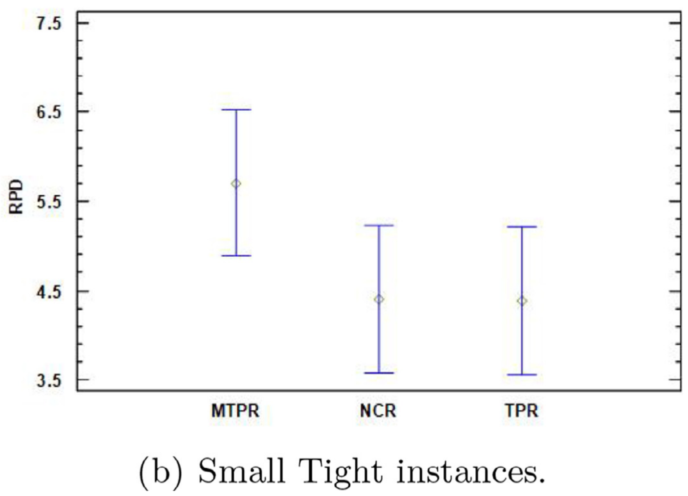
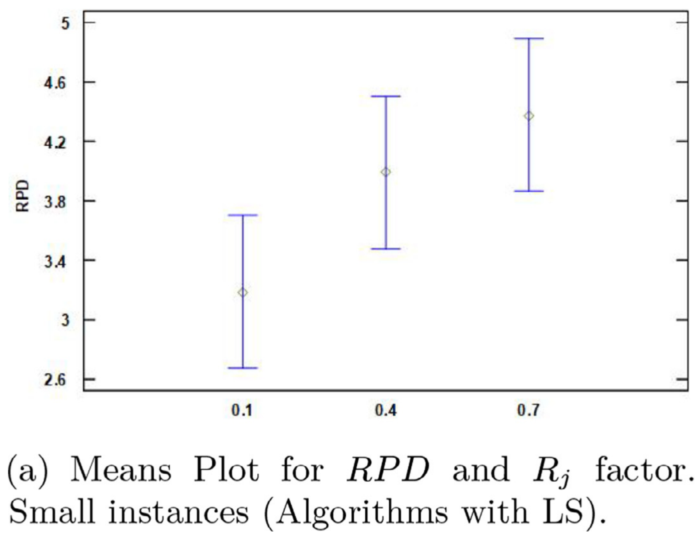


Fig. 3. Means Plot and LSD intervals at the 95% confidence level for the algorithms with local search (Small and Small Tight instances).



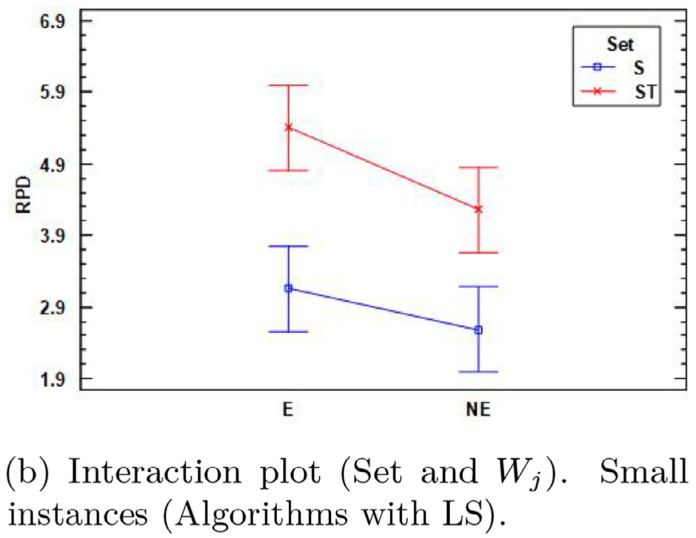


Fig. 4. Means Plot and Interaction plot with LSD intervals at the 95% confidence level for the algorithms with LS (Small instances).

Table 8
Average RPD over the optimal solution for small instances.

n	w_j	R_j	TPR	TPR_{ls}	$MTPR$	$MTPR_{ls}$	NCR	NCR_{ls}
5	E	0.1	30.5	0.9	24.4	0.2	4.4	1.3
		0.4	34.1	4.8	21.2	2.2	10.6	0.0
		0.7	55.8	5.3	57.6	2.8	19.5	0.8
	NE	0.1	72.0	0.3	58.5	0.3	4.7	0.2
		0.4	53.4	2.4	46.6	2.8	9.9	0.5
		0.7	38.1	2.4	57.9	2.4	12.3	2.3
10	E	0.1	75.0	3.3	32.4	4.7	15.0	4.4
		0.4	68.1	4.5	56.4	4.0	22.2	2.1
		0.7	69.3	4.2	55.4	7.8	14.8	3.6
	NE	0.1	98.0	2.1	68.6	7.0	13.1	2.5
		0.4	105.5	1.4	72.2	6.9	32.1	2.4
		0.7	89.2	2.9	106.7	4.3	18.9	3.2
Average			65.7	2.9	54.8	3.8	14.8	1.9

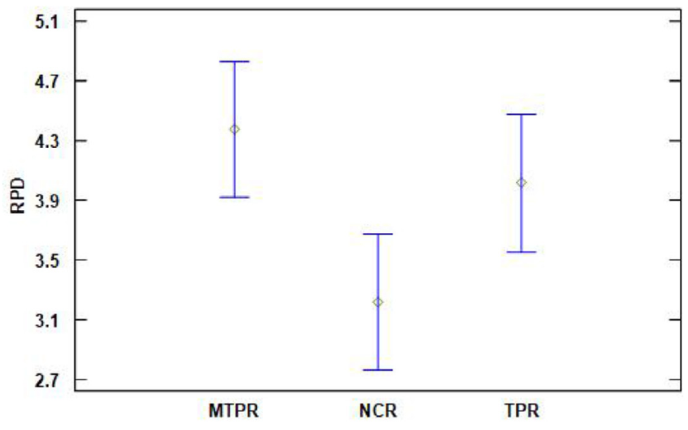
Table 9
Average RPD over the optimal solution for small tight instances.

n	w_j	R_j	TPR	TPR_{ls}	$MTPR$	$MTPR_{ls}$	NCR	NCR_{ls}
5	E	0.1	58.5	2.6	26.0	2.1	8.9	1.2
		0.4	50.1	2.2	48.2	3.7	31.8	2.5
		0.7	48.7	2.4	35.7	2.2	22.3	7.6
	NE	0.1	44.2	1.8	39.3	2.5	22.2	2.0
		0.4	62.2	0.8	33.0	2.1	14.1	3.7
		0.7	58.8	1.4	33.1	1.4	49.9	1.3
10	E	0.1	61.3	6.4	47.9	7.3	21.0	5.0
		0.4	96.1	6.7	69.7	8.7	36.0	12.1
		0.7	109.2	7.3	119.1	10.2	60.2	6.1
	NE	0.1	73.5	7.0	51.6	8.5	23.8	2.6
		0.4	80.9	5.1	51.5	10.4	23.1	3.6
		0.7	106.8	8.5	115.9	9.1	43.0	4.8
Average			70.9	4.4	55.9	5.7	29.7	4.4

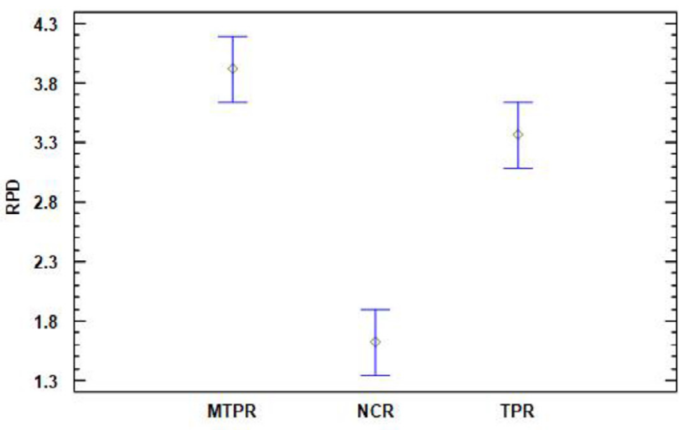
three values of ρ. The ANOVA test shows statistically significant differences among the *RPD* value between 0.1 and 0.7, that is, it seems that the higher the ρ, the higher the *RPD* value.

Fig. 4 (b) shows the interaction plot for W_j and the set of small instances (small and small tight). The effect of W_j can be seen in each set and it can be stated that there are no statistically significant differences between the average *RPD* for the small set. However, for the small tight set differences are observed (the intervals do not overlap). In that case, the *RPD* values obtained for the Equal set (E) are higher than the ones obtained with the Non-Equal (NE) set.

We now focus on medium and large instances. In this case optimal solutions are not known and the *RPD* values are calculated over the best known solution obtained among all methods.



(a) Medium instances (Algorithms with LS).



(b) Large instances (Algorithms with LS).

Fig. 5. Means Plot and LSD intervals at the 95% confidence level for the algorithms with local search (Medium and large instances).

Table 10
Average *RPD* over the best known solution for medium instances.

n	w_j	R_j	TPR	TPR_{ls}	$MTPR$	$MTPR_{ls}$	NCR	NCR_{ls}
15	E	0.1	67.6	3.0	33.8	2.7	8.8	1.9
		0.4	80.3	6.4	45.0	3.1	28.4	4.9
		0.7	65.4	3.0	81.7	2.2	27.8	2.4
	NE	0.1	111.6	1.9	70.2	3.4	27.3	0.4
		0.4	118.0	7.0	77.5	3.2	38.3	4.6
		0.7	115.2	1.3	120.3	9.4	39.7	7.0
Average			93.0	3.8	71.4	4.0	28.4	3.5
20	E	0.1	81.8	3.9	49.3	3.0	10.2	0.8
		0.4	77.6	4.5	55.8	2.4	35.5	4.3
		0.7	103.0	4.9	106.8	6.6	38.4	2.5
	NE	0.1	102.1	2.0	55.5	4.5	30.7	0.6
		0.4	125.6	4.9	66.3	7.8	32.2	2.9
		0.7	153.0	4.0	182.6	9.7	79.6	6.1
Average			107.2	4.0	86.1	5.7	37.8	2.9
30	E	0.1	94.4	1.7	52.0	3.6	19.0	2.8
		0.4	118.3	5.9	66.2	2.4	53.9	3.6
		0.7	118.1	3.8	114.4	3.1	86.0	4.6
	NE	0.1	99.5	4.2	58.0	5.0	43.6	0.6
		0.4	138.5	4.4	76.1	4.2	86.2	3.6
		0.7	137.8	4.0	120.5	5.1	69.7	0.9
Average			117.8	4.0	81.2	3.9	59.7	2.7
40	E	0.1	95.7	2.5	51.7	3.3	18.4	3.5
		0.4	111.4	8.3	52.6	2.5	67.0	6.0
		0.7	137.2	6.6	133.1	2.3	93.9	6.0
	NE	0.1	111.9	2.5	54.4	2.6	58.1	1.8
		0.4	133.6	3.7	81.3	6.9	75.1	2.9
		0.7	173.2	2.6	157.8	5.8	178.7	3.1
Average			127.2	4.4	88.5	3.9	81.9	3.9
Total Average			111.3	4.0	81.8	4.4	51.9	3.3

Table 11
Average *RPD* over the best known solution for large instances.

n	w_j	R_j	TPR	TPR_{ls}	$MTPR$	$MTPR_{ls}$	NCR	NCR_{ls}
50	E	0.1	100.2	1.6	62.2	4.9	29.5	1.8
		0.4	137.0	5.4	77.4	2.9	58.3	6.0
		0.7	143.4	6.8	114.5	7.6	92.2	4.6
	NE	0.1	112.6	3.0	83.7	2.7	43.0	1.2
		0.4	132.0	3.8	69.7	3.6	97.0	2.0
		0.7	158.7	5.6	113.7	6.2	109.7	1.7
Average			130.7	4.4	86.9	4.7	71.7	2.9
100	E	0.1	100.3	1.9	56.3	3.8	79.3	0.6
		0.4	139.5	3.0	78.7	3.7	134.3	2.3
		0.7	191.1	5.0	104.9	3.9	187.9	2.9
	NE	0.1	98.8	2.6	74.0	3.4	73.2	0.8
		0.4	169.7	4.8	90.9	4.3	141.4	0.5
		0.7	191.8	2.7	93.5	5.1	211.2	0.6
Average			148.5	3.3	83.1	4.0	137.9	1.3
150	E	0.1	95.7	0.5	52.0	1.3	88.3	0.5
		0.4	138.1	2.2	67.0	2.2	148.0	2.1
		0.7	196.8	3.3	101.6	3.0	224.7	2.5
	NE	0.1	102.2	2.4	65.4	3.1	102.8	0.8
		0.4	163.1	5.2	87.8	9.0	170.0	0.1
		0.7	216.4	4.0	116.8	6.3	257.6	1.2
Average			152.1	2.9	81.8	4.2	165.2	1.2
200	E	0.1	87.5	1.5	45.5	2.9	96.9	0.3
		0.4	134.0	4.8	58.4	3.2	152.1	1.1
		0.7	180.4	3.5	81.3	2.9	209.3	2.1
	NE	0.1	95.3	1.5	65.6	3.6	106.1	0.4
		0.4	152.7	4.4	73.9	7.3	159.1	0.9
		0.7	220.8	4.2	97.2	2.7	244.7	1.7
Average			145.1	3.3	70.3	3.8	161.4	1.1
Total Average			144.1	3.5	80.5	4.2	134.0	1.6

Tables 10 and 11 show these values for medium and large instances. Now the best performing algorithms without local search are *NCR* for medium instances and *MTPR* for large instances. Again, local search significantly improves the efficiency of the algorithms, and once local search is applied, the best performance is for the NCR_{ls} method for both medium and large instances. The tables show the evolution of average *RPD* values as the number of containers increases. In general, heuristic methods without local search are more sensitive to instance size than their counterparts with local search, that is, the larger the size, the higher the *RPD* value. Statistical analysis (ANOVA) is applied and Figs. 5(a) and 5(b) show the means plot with LSD intervals ($\alpha = 0.05$) for the algorithms with local search. It can be seen that there are statistically significant differences between the *NCR* and *MTPR* methods for medium instances and that there are statistically significant differences among all methods for large instances. In both cases, NCR_{ls} has the best performance.

Regarding the R_j and W_j parameters for medium and large instances, the same conclusions as for the small and small tight instances are obtained.

With respect to CPU times, all heuristic methods (without local search) are run in less than 7 milliseconds on average even for large instances. Heuristic methods with local search are run with a stopping criterion related to CPU time, one second per container. Then, the CPU times for the heuristic methods with local search vary from 5 seconds to 200 seconds, depending on the size of the instance.

After the analysis of the results, the conclusion is that, on average, the best performing method is the Nearest Container Rule with local search (NCR_{ls}). However, the contribution of the local search is essential for all methods, improving the results significantly.

7. Conclusions

Container terminal subsystems cannot be planned separately if good overall solutions are sought. This study focuses on the scheduling of the tasks of a yard crane in a block, in which some containers are retrieved for transfer to the sea or land, and others, coming from sea or land, are stored. Its relationships with the other parts of the terminal are given by the release times of the containers entering the block and the due times of those being retrieved. In addition, the number of I/O points on each side of the block has been considered and one of them has been assigned to each container. The effect of congestion, when there are more containers than I/O points, is also taken into account.

The problem is addressed from two points of view, first as a pick-up routing problem, and then as one-machine scheduling problem. For both approaches, integer linear models have been developed. The computational results show that the routing model produces optimal or near-optimal solutions for instances with up to 10 containers, while the scheduling model performs clearly worse. To obtain good solutions for larger instances, with up to 200 containers, several heuristic rules are developed, using some of the characteristics of the problem. Nevertheless, the results show that none of the proposed rules can capture all the features of the problem and their solutions are far from optimal. Fortunately, these solutions can be significantly improved by using a local search that rearranges the sequence in which containers are dealt with. The resulting algorithms obtain optimal or near-optimal solutions for small instances and can obtain good solutions for large problems in very short times.

This study advances in the integration of all parts of automatized container terminals. If the seaside and landside subsystems provide information on retrieval and storage times, yard crane movements can be sequenced to minimise the delays and waiting times. The solution process can also take into account the number and positions of I/O points to avoid congestion.

The study can be extended in several ways. On the one hand, more complex metaheuristics can be developed to improve on the results obtained. On the other, the approaches proposed here can be applied to cases in which more than one yard crane serves a block.

Acknowledgement

The authors are partially supported by the Spanish Ministry of Science and Innovation under the project “OPTEP-Port Terminal Operations Optimization” (No. RTI2018-094940-B-I00), partially financed with FEDER funds, by the Spanish Ministry of Science and Innovation under the project “OPRES-Realistic Optimization in Problems in Public Healt”(No. PID2021-124975OB-I00), partially financed with FEDER funds and by the Agència Valenciana de la Innovació (AVI) under the project ”ireves (innovación en vehículos de emergencia sanitaria): una herramienta inteligente de decisión” (No. INNACC/2021/26) partially financed with FEDER funds (interested readers can visit http://ireves.upv.es). Some of the authors are supported by the Faculty of Business Administration and Management at the Universitat Politècnica de València. Special thanks are due to Gerardo Gabriel Minella from Instituto Tecnológico de Informática.

References

Boysen, N., Briskorn, D., & Meisel, F. (2017). A generalized classification scheme for crane scheduling with interference. *European Journal of Operational Research, 258*, 343–357.
Boysen, N., & Stephan, K. (2016). A survey on single crane scheduling in automated storage/retrieval systems. *European Journal of Operational Research, 254*, 691–704.
Carlo, H. J., Vis, I. F. A., & Roodbergen, K. J. (2014). Storage yard operations in container terminals: Literature overview, trends, and research directions. *European Journal of Operational Research, 235*, 412–430.
Chen, X., He, S., Zhang, Y., Tong, L. C., & Shanga, P. (2020). Yard crane and AGV scheduling in automated container terminal: A multi-robot task allocation framework. *Transportation Research Part C, 1141*, 241–271.
Choe, R., Park, T., Ok, S. M., & Ryu, K. R. (2010). Real-time scheduling for non-crossing stacking cranes in an automated container terminal, pp. 625–631. In M. A. Orgun, & J. Thornton (Eds.), *AI 2007: Advances in artificial intelligence, LNAI 4830*. Springer.
Choe, R., Yuan, H., Yang, Y., & Ryu, K. R. (2012). Real-time scheduling of twin stacking cranes in an auto-mated container terminal using a genetic algorithm, pp. 238–243. In *Proceedings of the 27th annual ACM symposium on applied computing*. SAC'12. Riva del Garda.
Dorndorf, U., & Schneider, F. (2010). Scheduling automated triple cross-over stacking cranes in a container yard. *OR Spectrum, 32*, 617–632.
Galle, V., Barnhart, C., & Jaillet, P. (2018). Yard crane scheduling for container storage, retrieval, and relocation. *European Journal of Operational Research, 271*, 288–316.
Gao, Y., & Ge, Y. (2022). Integrated scheduling of yard cranes, external trucks, and internal trucks in maritime container terminal operation. In *Maritime policy & management*. https://doi.org/10.1080/03088839.2022.213517. In press
Gharehgozli, A. H., Laporte, G., Yu, Y., & de Koster, R. (2015). Scheduling twin yard cranes in a container block. *Transportation Science, 49*, 686–705.
Gharehgozli, A. H., Yu, Y., de Koster, R., & Du, S. (2019). Sequencing storage and retrieval requests in a container block with multiple open locations. *Transportation Research Part E, 125*, 261–284.
Gharehgozli, A. H., Yu, Y., de Koster, R., & Udding, K. T. (2014). An exact method for scheduling a yard crane. *European Journal of Operational Research, 235*, 431–437.
Huang, S. Y., & Li, Y. (2017). Yard crane scheduling to minimize total weighted vessel loading time in container terminals. *Flexible Services and Manufacturing Journal, 29*, 689–720.
Kemme, N. (2020). State-of-the-art yard crane scheduling and stacking, pp. 383–414. In J. W. Böse (Ed.), *Handbook of terminal planning. second edition*. Springer.
Kim, K. H., & Kim, K. Y. (1999). An optimal routing algorithm for a transfer crane in port container terminals. *Transportation Science, 33*, 17–33.
Kim, K. Y., & Kim, K. H. (2003). Heuristic algorithms for routing yard-side equipment for minimizing loading times in container terminals. *Naval Research Logistics, 50*, 498–514.
Li, W., Goh, M., Wu, Y., Petering, M. E. H., de Souza, R., & Wu, Y. (2012). A continuous time model for multiple yard crane scheduling with last minute job arrivals. *International Journal of Production Economics, 136*, 332–343.
Li, W., Wu, Y., Petering, M. E. H., Goh, M., & de Souza, R. (2009). Discrete time model and algorithms for container yard crane scheduling. *European Journal of Operational Research, 198*, 165–172.
Liu, W., Zhu, X., Wang, L., Yan, B., & Zhang, X. (2021). Optimization approach for yard crane scheduling problem with uncertain parameters in container terminals. *Journal of Advanced Transportation, 2021*, 5537114.
Montgomery, D. C. (2019). *Design and analysis of experiments*. New York, tenth edition: John Wiley & Sons.
Narasimhan, A., & Palekar, U. S. (2002). Analysis and algorithms for the transtainer routing problem in container port operations. *Transportation Science, 36*, 63–78.
Ng, W. C., & Mak, K. L. (2005a). An effective heuristic for scheduling a yard crane to handle jobs with different ready times. *Engineering Optimization, 37*, 867–877.
Ng, W. C., & Mak, K. L. (2005b). Yard crane scheduling in port container terminals. *Applied Mathematical Modelling, 29*, 263–276.
Park, T., Choe, R., Ok, S. M., & Ryu, K. R. (2010). Real-time scheduling for twin rmgs in an automated container yard. *OR Spectrum, 32*, 593–615.
Speer, U., & Fischer, K. (2017). Scheduling of different automated yard crane systems at container terminals. *Transportation Science, 51*, 305–324.
UNCTAD (2021). Review of Maritime Transport. https://unctad.org/system/files/official-document/rmt2021_en_0.pdf, Last accessed on 2022-07-14.
Wu, Y., Li, W., Petering, M. E. H., Goh, M., & de Souza, R. (2015). Scheduling multiple yard cranes with crane interference and safety distance requirement. *Transportation Science, 49*, 990–1005.
Xing, Z., Liu, H., Wang, T., Chew, E., Lee, L., & Tan, K. C. (2023). Integrated automated guided vehicle dispatching and equipment scheduling with speed optimization. *Transportation Research Part E: Logistics and Transportation Review, 169*, 102993.
Zheng, F., Man, X., Chu, F., Liu, M., & Chu, C. (2019). A two-stage stochastic programming for single yard crane scheduling with uncertain release times of retrieval tasks. *International Journal of Production Research, 57*, 4132–4147.