Crane-operated warehouses: Integrating location assignment and crane scheduling

Sam Heshmati[a,b], Túlio A. M. Toffolo[a,c], Wim Vancroonenburg[a,d,*], Greet Vanden Berghe[a]

[a]KU Leuven, Department of Computer Science, CODeS - *Belgium*

[b]University of Porto, Faculty of Engineering, INESC TEC – Portugal

[c]Federal University of Ouro Preto, Department of Computing – Brazil

[d]Research Foundation Flanders – FWO Vlaanderen

***Corresponding author**:

Wim Vancroonenburg, Ph.D. - wim.vancroonenburg@cs.kuleuven.be

Ghent Technology Campus,

Gebroeders De Smetstraat 1, 9000 Gent, Belgium.

Author notes

Wim Vancroonenburg is a postdoctoral researcher funded by Research Foundation Flanders - FWO Vlaanderen. Editorial consultation provided by Luke Connolly (KU Leuven).

Data instances + validator available at : https://bitbucket.org/Sam-Hes/cwsp/

Crane-operated warehouses: Integrating location assignment and crane scheduling

Abstract

Crane-operated warehouses constitute an essential asset for the many industries which must temporarily store products on their way from manufacturers to consumers. Such warehouses are a practical necessity rather than an explicitly desired service and they introduce significant operational costs which should be minimized. The problem addressed by the current paper, the Crane-operated Warehouse Scheduling Problem (CWSP), concerns the location assignment of input products and the scheduling of cranes for product movement in such warehouses. Several constraints are associated with the problem, for example certain products should not be stored close to each other (due perhaps to a difference in temperature or aroma) and cranes must respect operational safety distances between each other in order to prevent dangerous collisions. The present paper explores a novel methodology which combines these two decisions – location assignment and crane scheduling - instead of solving them sequentially. In addition to mathematical formulations for location assignment and crane scheduling, both an integrated mathematical formulation and a fast heuristic are presented for the CWSP. The quality of the mathematical formulation and the heuristic are compared against the conventional sequential approaches. Experimentation upon an extensive range of instances show significantly improved results are attainable when integrating location assignment and crane scheduling, despite some (expected) increase in computational time.

Keywords: Crane-operated warehouse scheduling, Crane scheduling, Crane interference, Location assignment

1. Introduction

Warehouses constitute a form of infrastructure commonly employed by manufacturers, wholesalers and retailers to store goods not only during the production process but also during their distribution. Warehouse efficiency therefore plays a crucial role in global economy. Their efficiency enhances the capacity of supply chains, providing significant economic and service benefits to both businesses and end users. Reducing storage and handling costs, increasing warehouse capacity and improving the timeliness of deliveries are essential to further sustaining and strengthening supply chains. The present study focuses on the first aspect, in particular in *Crane-operated warehouses*.

The term 'Craned-operated warehouse' refers to a type of warehouse or storage area which employs any type of overhead crane such as rubber-tired gantry cranes (RTGCs) or rail mounted gantry cranes (RMGCs). Overhead cranes are commonly used in industrial warehouses where the stored products are rather heavy and large-sized in nature (examples of which include steel coils, large pallets of goods, . . .) or in container terminals where containers are temporarily stored in stacks before being transferred to their next destinations. The Crane-operated Warehouse Scheduling Problem (CWSP) studied throughout this paper concerns the optimization of both the products' storage location and the crane operations which are necessary to do so in warehouses which employ such overhead cranes. A warehouse typically consists of a set of input and output points which are located in the periphery of a storage area. Products are stored subject to a range of operational constraints and a set of cranes are employed for handling operations. In many cases, cranes cannot overtake each other (such as when they are operating on the same pair of rails), thereby necessitating proper safety measures to avoid collisions. The CWSP as such is composed of two constituent optimization problems, namely:

i) The Location Assignment Problem (LAP): assigning the storage locations to incoming products and those which must be relocated within the storage area.

ii) The Crane Scheduling Problem (CSP): scheduling the cranes' operations.

The objective is to minimize both total storage cost and tardiness of crane operations.

The CWSP is conventionally split into the two aforementioned sub-problems – the LAP and CSP - which are solved sequentially. First, the LAP is solved and the resulting storage locations for incoming/relocated products are fixed. Next, the CSP is solved to determine the best schedule for the handling operations. To date, there has been a considerable lack of research which assesses the impact of integrating these two sub-problems.

Container terminals represent one specific real-world application where the CWSP is encountered. Given the continuously increasing volume of containers being handled in terminals worldwide, which places significant pressure on terminals' infrastructure and operations, it is unsurprising that there exists a vast body of container terminal literature relevant to the problem.

The majority of studies related to the LAP involve optimization problems in container terminals such as the re-handling problem (Jovanovic and Voß, 2014; Ku and Arthanari, 2016) and the container stacking problem (Zhang et al., 2014; Gharehgozli et al., 2014). The re-handling problem concerns removing containers from stacks to enable a given set of container retrievals where the objective is to minimize the number of moves. Studies addressing the container stacking problem mostly focus on minimizing reshuffling, namely those unproductive moves required to gain access to a desired container which is blocked (Chen and Lu, 2012; Boysen and Emde, 2016). Other objectives include minimizing travelling distance, wasted space, or estimated retrieval cost (Park et al., 2011).

The objective function of the LAP in the present study derives itself directly from operational practices found in production industries and differs from objective functions found in references related to container terminals. It includes cost terms related to storing a product in a specific location and others related to storing certain products adjacent to one another. The former cost terms are used to model the retrieval costs (specified as a distance from an output point), while the latter model operational constraints of production industries which seek to avoid storing certain products in close proximity. For instance, companies may wish to avoid storing aromatic products next to each other or may require hot products to be stored away from those which have already cooled down. While companies may often disallow such neighbouring location assignments altogether, in situations of high storage occupancy it may not always be feasible to do so. Addressing such situations as soft constraints, penalized as costs in the objective function, enables the necessary modelling flexibility and avoids infeasibility.

Many studies address the CSP independent from the LAP, considering the LAP's solution as a fixed input. The most relevant references to the present study are those focused on scheduling multiple cranes. For single crane scheduling, interested readers are referred to the survey by Boysen and Stephan (2016). Due to the increasing necessity to accelerate handling operations in warehouses, many recent papers have focused on scheduling multiple cranes operating simultaneously within the same storage area. Dorndorf and Schneider (2010) studied a container yard in which a pair of cranes operates on the same rails with another larger crane operating above them (cross-over crane) on its own pair of rails. Each of the two smaller cranes has its own distinct working area to avoid collisions. Given independent and mutually-exclusive working areas and the presence of a separate cross-over crane, crane interference does not pose a problem in such yards. By contrast, Li et al. (2009) considered a container terminal which employs multiple cranes that may interfere with one another. They proposed a discrete-time MIP model for the problem and a heuristic to solve it. Li et al. (2012) extended Li et al. (2009)'s work by proposing a continuous time MIP model capable of handling instances with a higher number of storage and retrieval requests. Wu et al. (2015) also considered a container terminal with multiple cranes, as Li et al. (2009, 2012), and proposed a polynomial time heuristic to solve their optimization problem. These studies are particularly interesting with regard to how they model the scheduling of multiple cranes operating in storage areas with inter-crane interference. However a noteworthy and significant difference with respect to the present work lies in how within all the aforementioned studies containers are delivered. This means cranes remain static at the stacking piles and do not move during handling operations. Consequently, the duration of all operations can be assumed to be equal. This simplifies the problem modeling by enforcing equal time durations for all operations. In a general setting, however, input and output may occur anywhere around the storage area and cranes move over that storage area while handling products. Gharehgozli et al. (2017) investigated a set of rules and their influence on the effect of temporary locations in a so-called *handshake area* which facilitates container handover between cranes. The paper presented some managerial insights on the size, location, and number of such handshake areas.

Gharehgozli et al. (2015) attempted to integrate location assignment and crane scheduling problems in a container terminal, wherein the water-side crane performs all requests which must be stacked or retrieved from the water-side, and land-side operations are carried out similarly by a land-side crane. However a significant limitation to their model is that storage and retrieval requests are already assigned to cranes in advance. Moreover, the model was designed for only one land-side and one water-side crane, and thus cannot accommodate cases with more than two cranes, or cases where both cranes may handle requests from anywhere throughout the storage area.

In practice, warehouse managers are becoming increasingly aware that warehouse efficiency may be bolstered by exploiting an integrated optimization approach, where location assignment and crane scheduling decisions are simultaneously taken into consideration and jointly optimized (Darvish and Coelho, 2018). The literature is however lacking studies that investigate this. The present research, therefore, focuses on this integrated approach of handling the CWSP. It provides a general setting which may be easily adapted to other warehouses, land-side container terminals or any other industry employing multiple gantry cranes for product handling.

Mathematical formulations and heuristics are developed and tested upon a set of instances which are randomly-generated using probability distributions and insights extracted from a relevant industrial case. Results are compared against those obtained with a heuristic based on the dispatching rules and manual strategies employed in practice. The findings from this computational study reveal the significant benefits of combining the LAP and CSP when solving the CWSP.

The remainder of the paper is structured as follows. Section 2 provides a detailed problem definition of the CWSP. Section 3 presents mathematical formulations for the LAP and the CSP, and also formulates the CWSP by means of a continuous-time mixed integer programming model which considers realistic constraints. Section 4 presents a heuristic algorithm for solving the LAP, CSP and CWSP. Computational experiments and a comparative algorithmic performance analysis are detailed throughout Section 5. Finally, Section 6 summarizes the paper's primary findings and discusses possible future research directions.

2. Problem definition

Throughout this study, a crane-operated warehouse is considered which consists of a storage area within which products are placed. The storage area is composed of locations, with each location storing at most one product. A set of special locations representing input/output (I/O) points around the storage area is defined where input requests originate and output requests must be delivered. Each I/O point either originates input or collects output requests which must be processed by their due time.

Each request consists of a product that must be moved. Requests are divided into two sets:

- *Input requests* ($\mathcal{R}^I$): requests which require location assignment. $\mathcal{R}^I$ consists of requests for products at an input point requiring transfer to the yard or products that must be moved within the yard to enable cranes to access locations, located beneath them, associated with *output requests*;
- *Output requests* ($\mathcal{R}^O$): requests consisting of products within the yard requiring transfer to an output point.

Set $\mathcal{R}$ represents the union of the two sets: $\mathcal{R} = \mathcal{R}^I \cup \mathcal{R}^O$. A release time and due time are associated with each request, defining when the product is available for transfer and when it is due to be transferred.

The set of available locations L consists of locations that are already free or will become free during the scheduling horizon when their stored product has been moved. This includes the origin location of the *output* requests and those requests which move products inside the yard. L excludes locations which store products that will not be moved during the scheduling horizon. Following convention, the storage area length is mapped to a horizontal coordinate axis. A horizontal coordinate h_l is associated with each individual location $l \in L$. The horizontal coordinates in the yard are ordered from left to right. A product may be stored in a location above ground level, stacked on another product. Therefore, in addition to its horizontal and lateral coordinates, a location l is also defined by its level above the ground. To be able to store a product in a location above ground level, all locations beneath the product must be occupied by other products. Cranes are employed to execute input and output requests. A set of available, identical cranes $\mathcal{C}$ is defined, each being capable of handling one request at a time. Cranes are mounted on a pair of rails along the horizontal axis, and are ordered and indexed from left to right in the storage area. Additionally, cranes have no predefined working areas, the only restriction being that they cannot cross and that a safety distance must be respected between neighbouring cranes while moving throughout the storage area. This study assumes that cranes can reach all locations.

Figure 1 illustrates a top-view of a crane-operated storage area in which the gray border represents the input/output points. Three cranes are ordered from left to right and operate across the storage area. Note that the safety distance must be respected and therefore, cranes cannot pass over each other.

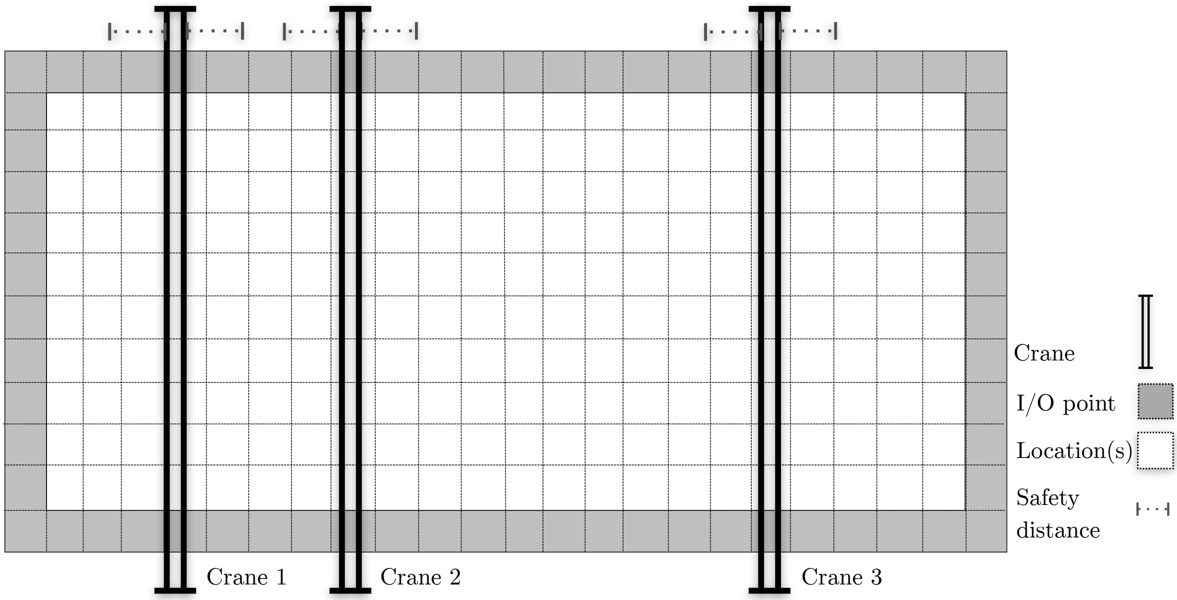


Figure 1: Top view of a warehouse employing three cranes.

The CWSP consists of two optimization problems, the LAP and the CSP. The LAP's objective is to minimize the total storage cost of *input* requests. The storage cost for a product is defined in terms of an assignment in the neighbourhood of other products in the storage area. The total storage cost includes the cost, summed over all input requests, of assigning an *input* request to a location in the storage area (pre-calculated and corresponding with the distance to neighbouring products which will not move during the scheduling horizon) and the cost of assigning two *input* requests in neighbouring locations.

The CSP consists of deciding when and by which crane each request will be executed, while respecting precedence constraints and constraints concerning safety distances. Precedence constraints may be predetermined or introduced during location assignment. Predetermined precedence constraints follow from when a product is stacked on top of a product associated with an *output* request, the top product must be removed first after which the output request may be executed. Precedence constraints introduced during location assignment follow from assigning an input request to the location of a request originated within the yard (either input or output) or assigning two input requests on top of each other. The objective is to minimize total tardiness of all requests. A request's tardiness equals the difference between its completion time and due time if positive, or zero otherwise.

The combined problem of solving both the LAP and CSP simultaneously is referred to as the CWSP. The objective of the CWSP is to minimize the weighted linear expression presented in Equation (1), where α and β are weights defining the relative importance of the terms, while E_{LA} and E_{CS} correspond to total storage cost and total tardiness, respectively.

$$\text{Total cost} = \alpha \cdot E_{LA} + \beta \cdot E_{CS} \tag{1}$$

3. Mathematical formulation

This section presents mathematical formulations for the LAP (Section 3.1) and CSP (Section 3.2), followed by a formulation for the CWSP (Section 3.3) which considers the LAP and CSP simultaneously.

3.1 Location assignment problem

Formulation $\mathcal{F}_{LA}$ concerns the assignment of destination locations to input requests. Table 1 summarizes the notation employed for the LAP formulation.

$$
\mathcal{F}_{LA}
\begin{cases}
min \displaystyle\sum_{i \in \mathcal{R}^I} \sum_{l \in L} \gamma_{il} x_{il} + \sum_{i \in \mathcal{R}^I} \sum_{j \in \mathcal{R}^I} \omega_{ij} z_{ij} & & (2)\\
s.t. \quad \displaystyle\sum_{l \in L : l \neq b_i} x_{il} = 1 & \forall\, i \in \mathcal{R}^I & (3)\\
\displaystyle\sum_{i \in \mathcal{R}^I} x_{il} \leq 1 & \forall\, l \in L & (4)\\
x_{il} \leq \displaystyle\sum_{j \in \mathcal{R}^I} x_{jk} & \forall\, i \in \mathcal{R}^I,\ l \in L,\ k \in U_l & (5)\\
x_{il} + x_{jk} \leq 1 + z_{ij} & \forall\, i, j \in \mathcal{R}^I,\ l \in L,\ k \in N_l & (6)\\
x_{il} \in \{0, 1\} & \forall\, i \in \mathcal{R}^I,\ l \in L & (7)\\
z_{ij} \in \{0, 1\} & \forall\, i, j \in \mathcal{R}^I, & (8)
\end{cases}
$$

Objective function (2) minimizes the total storage cost. The storage cost is divided into two parts: the cost of assigning request i to location l and the cost of assigning requests i and $j \in \mathcal{R}^I$ in each other's neighbourhood, denoted by ω_{ij} and γ_{il} respectively. Constraints (3) and (4) are classic assignment constraints ensuring exactly one location is assigned to each input request and that each location receives, at most, a single request, respectively. Constraints (3) also prevent assigning *input* requests to their origin locations. If request i represents a product that must be moved within the yard to access a product below, other input requests may use b_i as their destination, after the product below b_i has been moved. Constraints (5) force all *available* locations underneath location $l \in L$ to have an *input* request assigned, thus ensuring that no product is stacked atop an empty location. It is sufficient to assure there is an *input* request assigned to each *available* location $k \in U_l$, since the products not associated with any requests will not be moved during the scheduling horizon. Constraints (6) set the value of z_{ij} to 1 if requests i and j are placed in neighbouring locations and 0 otherwise. Constraints (7) and (8) state variables x_{il} and z_{ij} are binary.

Following the LAP, precedence constraints may be implied when input requests are assigned to the origin locations of output requests or when two input requests are assigned to locations where one is on top of the other. The outcome of the LAP (destination location for input requests) along with the set of precedence requests is the input for the CSP.

3.2 Crane scheduling problem

Formulation $\mathcal{F}_{CS}$ models the CSP which considers the crane assignment for requests and the sequencing of requests per crane. $\mathcal{F}_{CS}$ implements various realistic operational constraints such as

Table 1: Notations for the LAP formulation

Sets:	
$\mathcal{R}$:	set of all requests
$\mathcal{R}^I$:	set of input requests, $\mathcal{R}^I \subseteq \mathcal{R}$
L :	set of available locations, those locations that are currently empty or will become empty due to product movements.
U_l :	set of all available locations underneath location l, $U_l \subset L$
N_l :	set of all neighbouring locations of location l, $N_l \subset L$
Parameters:	
ω_{ij} :	storage cost of assigning request i to a neighbouring location of product associated with request j
γ_{il} :	storage cost of assigning request i to location l
b_i :	origin location of request i
Decision variables:	
x_{il} :	binary variable equal to 1 if request i is assigned to location l and 0 otherwise
z_{ij} :	binary variable equal to 1 if request i is assigned to a neighbouring location of request j's destination and 0 otherwise

multiple cranes working simultaneously in the storage area and precedence constraints. Note that each crane can traverse the entire storage area provided safety distances between all cranes are respected. This means that a crane may move beyond the storage area boundary to provide space for another crane to access storage location at or close to the area's perimeter.

The continuous-time formulation for the CSP presented in this paper was inspired by Li et al. (2012), who showed that for the CSP with multiple cranes this formulation significantly reduced the model's size and enabled larger instances to be solved compared to a discrete-time formulation for the same problem. Recall from Section 1 that Li et al. (2012) considered the CSP in a container terminal where containers were brought directly in front of the stacking pile. As a consequence, cranes do not move along the rails when moving a product. They instead move products laterally (along the crane beam). However, in a general setting of crane-operated warehouses the cranes move along the storage area to reach the respective input/output point during their operations. Conflicting requests and variable operation durations are consequently inevitable. The model presented in the following section accounts for this additional complexity. Table 2 summarizes the notation employed to formulate the CSP.

The constraints of $\mathcal{F}_{CS}$ are organised into three categories: (i) *request assignments for cranes*, (ii) *handling conflicting requests* and, finally, (iii) *setting the requests' starting times.*

(i) Request assignments for cranes

Constraints (9) ensure that exactly one crane is assigned to each request. Constraints (10) and (11) determine the value of variable n_{ij} which must be 1 if request i finishes before the starting time of request j and 0 otherwise.

Table 2: Notations for the CSP formulation

Sets:	
$\mathcal{R}^O$:	set of output requests, $\mathcal{R}^O \subseteq \mathcal{R}$
Γ_i :	set of requests that must be executed before request i, $\Gamma_i \subseteq \mathcal{R}$
$\mathcal{C}$:	set of cranes
Parameters:	
$sd_{cc'}$:	safety distance required between cranes c and c'
$st_{cc'}$:	time required by a crane to travel the safety distance between cranes c and c'
M_L :	yard length + $\sum_{c \in \mathcal{C}} sd_{c(c+1)}$
M_T :	large number, 2 $\times$ required time for a crane to travel the yard length $\times$ number of requests
η_c :	order of crane c in the storage yard, $0 \leq \eta_c < NC$
d_i :	duration of request i
b_i :	origin location of request i
e_i :	destination location of request i
r_i :	release time of request i
τ_i :	due time of request i
h_l :	horizontal coordinate of location l, denoting the coordinate along the rails
h_i^- :	horizontal coordinate of the leftmost location of request i's trajectory
h_i^+ :	horizontal coordinate of the rightmost location of request i's trajectory
$g_{icjc'}$:	required waiting time between the start time of request i by crane c and the start time of request j by crane c' (due to possible conflicts)
t_{lk} :	time required by a crane to travel from location l to location k
o_{ij}^a :	equal to 1 if trajectory of request i is to the left of request j's trajectory and 0 otherwise
Decision variables:	
y_{ic} :	binary variable equal to 1 if request i is handled by crane c and 0 otherwise
s_i :	continuous variable indicating the start time of request i
δ_i :	continuous variable indicating the tardiness of request i
Auxiliary variables:	
n_{ij} :	binary variable equal to 1 if request i finishes before the start time of request j and 0 otherwise
q_{ij} :	binary variable equal to 1 if request i begins before the start time of request j and 0 otherwise
o_{ij} :	binary variable equal to 1 if requests i and j are conflicting and 0 otherwise
o_{ij}^b :	binary variable equal to 1 if the crane assigned to request i is to the right of the crane assigned to request j and 0 otherwise.
o_{ij}^c :	binary variable equal to 1 if the distance between h_i^- and h_j^+ is less than the safety distance required between cranes handling them and 0 otherwise

$$\sum_{c \in \mathcal{C}} y_{ic} = 1 \qquad \forall i \in \mathcal{R} \tag{9}$$

$$s_i + d_i \geq s_j - M_T n_{ij} \qquad \forall i, j \in \mathcal{R} : i \neq j \tag{10}$$

$$s_i + d_i \leq s_j + (1 - n_{ij}) M_T \qquad \forall i, j \in \mathcal{R} : i \neq j \tag{11}$$

Each crane may move only one product at a time. When two requests are scheduled within overlapping times ($n_{ij} = n_{ji} = 0$), they must be assigned to different cranes. Figure 2 presents two cases involving requests i and j where the horizontal axis represents time (t). In the first case, s_j (starting time of j) is larger than s_i and smaller than $s_i + d_i$ (finishing time of i), and thus $n_{ij} = n_{ji} = 0$ (time overlapping requests). In the second case, request j is executed after request i is finished and, therefore, $n_{ij} = 1$ and $n_{ji} = 0$ (non-overlapping moves).

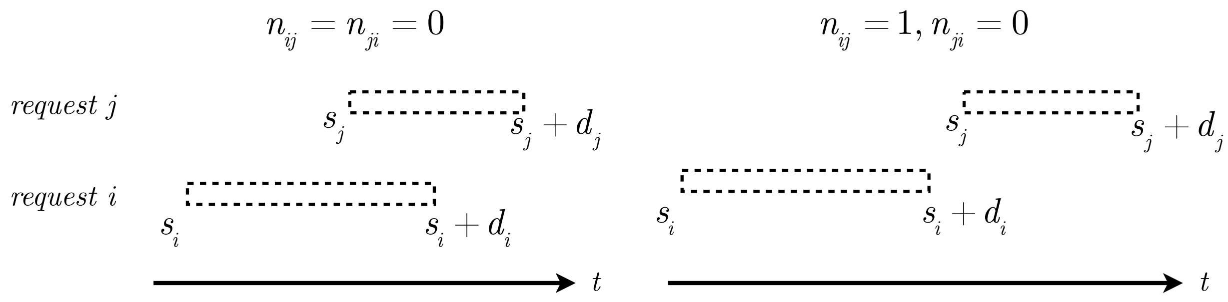


Figure 2: An example of overlapping and non-overlapping requests with respect to time.

Constraints (12) prevent the assignment of time-overlapping requests to the same crane.

$$y_{ic} + y_{jc} \leq 1 + n_{ij} + n_{ji} \qquad \forall i, j \in \mathcal{R} : i \neq j, \forall c \in \mathcal{C} \tag{12}$$

(ii) Handling conflicting requests

Cranes cannot pass each other and must respect a safety distance to avoid collision. The physical constraints due to non-crossing and safety requirements of cranes pose a significant challenge. If simultaneously executing requests i and j violates the safety distance, then these requests are conflicting and must be scheduled at different times. When this situation occurs, binary auxiliary variable o_{ij} is set to 1, indicating requests i and j are conflicting.

To assist in identifying conflicting requests, the *minimum horizontal coordinate* (h_i^-) and *maximum horizontal coordinate* (h_i^+) of a request i are employed and are independent of the requests movement direction. Since the origin and destination locations of requests are given by the location assignment, values $h_i^- = \min(h_{b_i}, h_{e_i})$ and $h_i^+ = \max(h_{b_i}, h_{e_i})$ are easy to pre-compute.

Requests assigned to different cranes may be conflicting depending on their minimum and maximum horizontal coordinates and on the position of the assigned cranes. Given two requests i and j handled by cranes c and c' respectively and a required safety distance $sd_{cc'}$, two different situations are possible. The first situation arises when the cranes must pass each other to handle the requests whereas the second situation occurs when there is insufficient space for them to respect the safety distance and handle the requests.

Figure 3 illustrates trajectories of two requests i and j as well as of cranes c and c'. Two cases may be considered for these two requests. In the first case where crane c is assigned to request i and crane c' to request j, ($y_{ic} = y_{jc'} = 1$), the allocation is such that no conflict occurs. In the second scenario, however, inverting the crane assignments ($y_{ic'} = y_{jc} = 1$) renders the simultaneous handling of requests impossible, given that cranes cannot pass each other therefore, requests i and j are conflicting, and starting times s_i and s_j must be different.

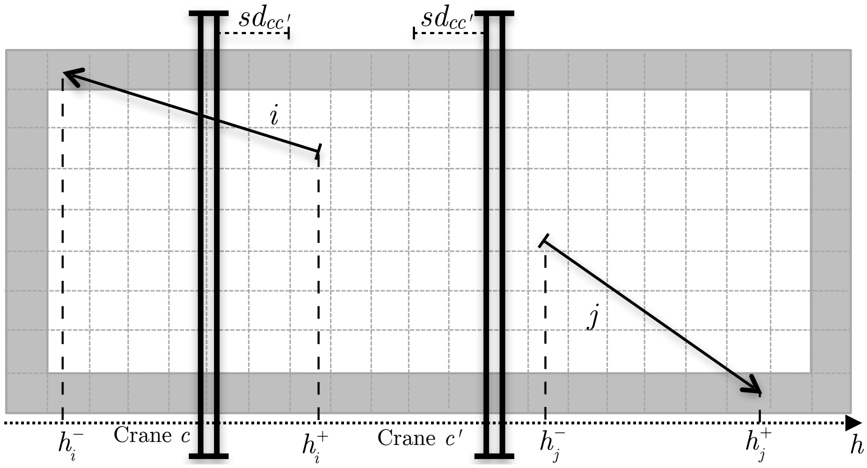


Figure 3: Conflicting requests due to crane assignments.

Parameter o^a_{ij} and auxiliary variable o^b_{ij} identify conflicting requests due to cranes requiring to pass each other. o^a_{ij} indicates whether the trajectory of request i is completely to the left of j's trajectory, such that $h_i^+ < h_j^- \Rightarrow o^a_{ij} = 1$. Constraints (13) are employed to define the values of o^b_{ij}, which equals 1 if the crane assigned to request i is to the right of the crane assigned to request j, $o^b_{ij} = 1$.

$$\sum_{c \in \mathcal{C}} \eta_c \, y_{jc} \geq \sum_{c \in \mathcal{C}} \eta_c \, y_{ic} - |\mathcal{C}| \cdot o^b_{ij} \qquad \forall i, j \in \mathcal{R} : i \neq j \tag{13}$$

Whenever both o^a_{ij} and o^b_{ij} equal one, i's trajectory is to the left of j's trajectory while i's crane is to the right of j's crane, resulting in a conflict. Constraints (14) force o_{ij} to take a value of 1 whenever $o^a_{ij} = o^b_{ij} = 1$. Likewise, Constraints (15) force o_{ij} to take value 1 whenever $o^a_{ji} = o^b_{ji} = 1$. When request i is conflicting with request j, then request j is conflicting with request i, implying $o_{ij} = o_{ji}$.

$$o_{ij} \geq o^a_{ij} + o^b_{ij} - 1 \qquad \forall i, j \in \mathcal{R} \tag{14}$$

$$o_{ij} \geq o^a_{ji} + o^b_{ji} - 1 \qquad \forall i, j \in \mathcal{R} \tag{15}$$

Another cause of request conflict concerns the safety distance between cranes. Figure 4 illustrates the trajectory of two requests i and j and cranes c and c'. Requests i and j are conflicting as there is insufficient space for the cranes to begin handling the requests while respecting the safety distance ($h_i^- - h_j^+ < sd_{cc'}$). Given the position of the requests' minimum and maximum horizontal coordinates, they are conflicting and the cranes cannot begin executing them simultaneously.

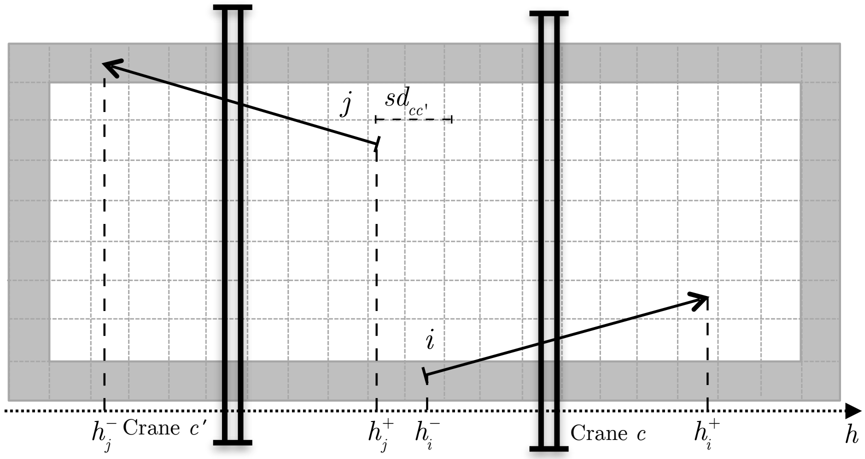


Figure 4: Conflicting requests with overlapping trajectories.

Auxiliary binary variables o^c_{ij} are introduced to identify such conflict. Constraints (16) determine the value of o^c_{ij}. Whenever i and j are handled by cranes c and c' ($y_{ic} = y_{jc'} = 1$ or $y_{ic'} = y_{jc} = 1$), the maximum value for $y_{ic} + y_{ic'} + y_{jc} + y_{jc'}$ equals two and therefore, Constraints (16) ensure o^c_{ij} equals one whenever the *minimum horizontal coordinate* of i, h^-_i, conflicts with the *maximum horizontal coordinate* of j, h^+_j, and 0 otherwise ($h^-_i - h^+_j < sd_{cc'} \Rightarrow o^c_{ij} = 1$).

$$h^-_i - h^+_j \geq sd_{cc'} - (2 - y_{ic} - y_{ic'} - y_{jc} - y_{jc'} + o^c_{ij})M_L \qquad \forall i, j \in \mathcal{R} : i \neq j, c, c' \in \mathcal{C} : c \neq c' \quad (16)$$

Whenever both o^c_{ij} and o^c_{ji} equal one, a conflict is detected. Constraints (17) force o_{ij} (and o_{ji}) to take a value of 1 whenever $o^c_{ij} = o^c_{ji} = 1$.

$$o_{ij} \geq o^c_{ij} + o^c_{ji} - 1 \qquad \forall i, j \in \mathcal{R} : i \neq j \quad (17)$$

(iii) Setting the requests' starting times

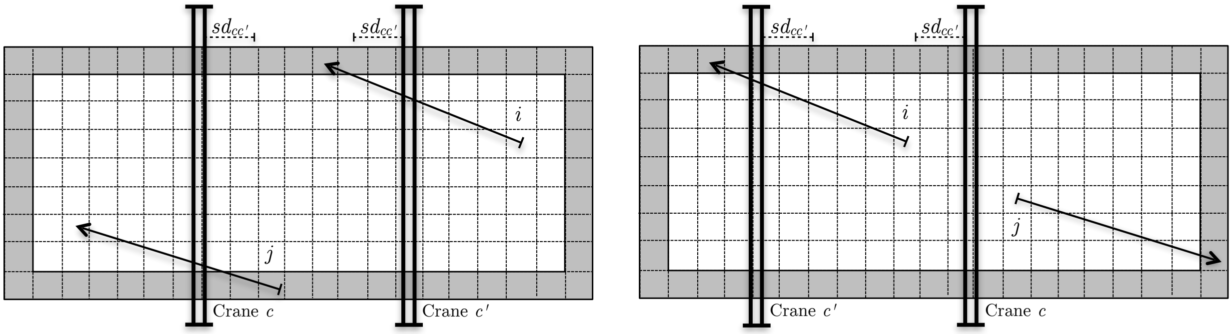
A crane can handle one request at a time, therefore, to execute two consecutive requests, a crane requires sufficient time to finish executing the first request and then travel from its destination to the second request's origin.

If two requests i and j are handled by a single crane c, ($y_{ic} = y_{jc} = 1$), i and j must be handled one at a time ($n_{ij} = 1$ or $n_{ji} = 1$). Assume request j is executed after finishing i ($n_{ij} = 1$). Constraints (18) ensure starting time s_j is greater than or equal to the sum of i's finishing time ($s_i + d_i$) and the time required by the crane to travel from i's destination to j's origin ($t_{e_i b_j}$). Constraints (18) hold when $y_{ic} + y_{jc} + n_{ij} = 3$ (requests are handled by a single crane and j is executed after finishing i), $\Rightarrow s_j \geq s_i + d_i + t_{e_i b_j}$.

$$s_j \geq s_i + d_i + t_{e_i b_j} - (3 - y_{ic} - y_{jc} - n_{ij})M_T \quad \forall i, j \in \mathcal{R}, i \neq j, \forall c \in \mathcal{C} \quad (18)$$

The starting time of two requests handled by different cranes depends on whether they are conflicting or not. When the cranes' trajectories while handling two requests are not conflicting, then the requests' starting times do not influence each other. However, if the two requests are conflicting, a waiting time must be applied between starting the requests. $g_{icjc'}$ denotes the required waiting time between request i handled by crane c and request j handled by crane c'. The value of $g_{icjc'}$ depends upon the position of the cranes involved and on their movement direction resulting in three different cases reflected by Equations (19), (20) and (21).

The first case (Figure 5(a)) occurs when $y_{ic} = y_{jc'} = 1$, crane c is to the left of c', $\eta_c < \eta_{c'}$ and i's destination is to the right of j's origin, $h_{e_i} > h_{b_j}$. In such situation crane c' cannot immediately begin executing j after the starting time of i, and must instead wait for c to finish i, d_i, plus the time c requires to travel from i's destination to j's origin, $t_{e_i b_j}$. Therefore the waiting time of crane c' to execute request j is $d_i + t_{e_i b_j}$. Another situation which results in the same value for $g_{icjc'}$ occurs when crane c is to the right of c', $\eta_c > \eta_{c'}$ and i's destination is to the left of j's origin, $h_{e_i} < h_{b_j}$ (Figure 5(b)). Figure 5 presents these situations both resulting in $g_{icjc'} = d_i + t_{e_i b_j}$.



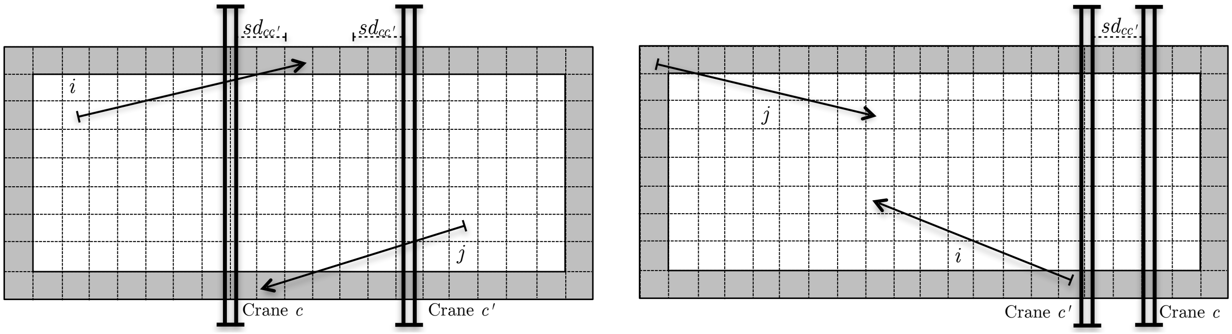
(a) $h_{e_i} > h_{b_j}, \eta_c < \eta_{c'}$ (b) $h_{e_i} < h_{b_j}, \eta_c > \eta_{c'}$

Figure 5: Two conditions which result in $g_{icjc'} = d_i + t_{e_i b_j}$.

The value of $g_{icjc'}$ is determined by Equations (19).

$$g_{icjc'} = d_i + t_{e_i b_j} \qquad \forall i \in \mathcal{R}, \begin{cases} j \in \mathcal{R} : h_{e_i} > h_{b_j}, \forall c, c' \in \mathcal{C} : \eta_c < \eta_{c'} \\ j \in \mathcal{R} : h_{e_i} < h_{b_j}, \forall c, c' \in \mathcal{C} : \eta_c > \eta_{c'} \end{cases} \tag{19}$$

Conflicts may also occur when crane c' does not necessarily need to wait until the end of request i. Figure 6 illustrates examples where crane c' may begin executing request j, $t_{e_i b_j}$ time units before concluding request i. Such value for $g_{icjc'}$ guarantees that by the time crane c finishes executing request i, crane c''s distance to crane c exceeds $sd_{cc'}$.



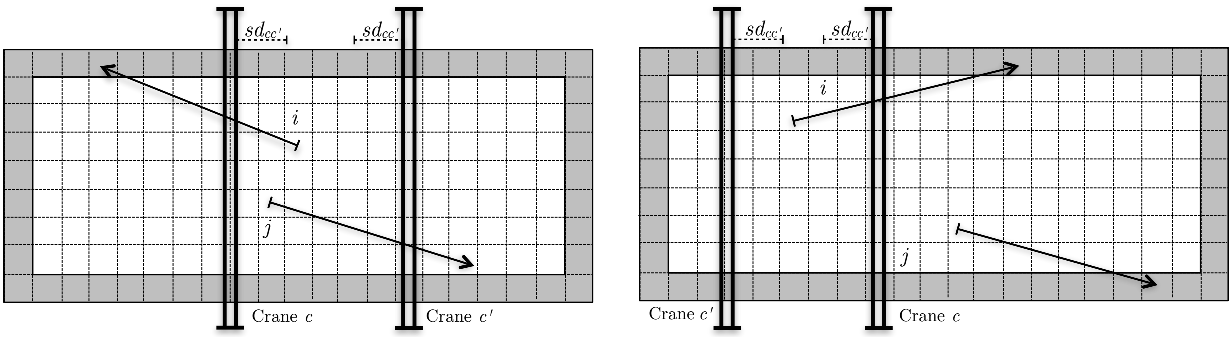
(a) $h_{b_i} < h_{e_i} < h_{b_j}$, $\eta_c < \eta_{c'}$ (b) $h_{b_j} < h_{e_i} < h_{b_i}$, $\eta_c > \eta_{c'}$

Figure 6: Two conditions which result in $g_{icjc'} = d_i - t_{e_i b_j}$.

Equations 20 define the conditions for $g_{icjc'} = d_i - t_{e_i b_j}$.

$$g_{icjc'} = d_i - t_{e_i b_j} \qquad \forall i \in \mathcal{R}, \begin{cases} j \in \mathcal{R} : h_{b_i} < h_{e_i} < h_{b_j}, \forall c, c' \in \mathcal{C} : \eta_c < \eta_{c'} \\ j \in \mathcal{R} : h_{b_j} < h_{e_i} < h_{b_i}, \forall c, c' \in \mathcal{C} : \eta_c > \eta_{c'} \end{cases} \tag{20}$$

Figure 7 presents two other scenarios which influence the value of $g_{icjc'}$. The first scenario occurs when crane c is to the left and c' is to the right, $\eta_c < \eta_{c'}$, and the destination of request i is to the left of j's origin, $h_{e_i} < h_{b_j}$ and $h_{e_i} < h_{b_i}$, or when crane c' is to the left of c, $\eta_c > \eta_{c'}$, and the destination of i is to the right of j's origin, $h_{e_i} > h_{b_j}$ and $h_{e_i} > h_{b_i}$. In these cases $g_{icjc'}$ is equal to the travel time from i's to j's origin, $t_{b_i b_j}$.



(a) $h_{e_i} < h_{b_j}$, $h_{e_i} < h_{b_i}$, $\eta_c < \eta_{c'}$ (b) $h_{e_i} > h_{b_j}$, $h_{e_i} > h_{b_i}$, $\eta_c > \eta_{c'}$

Figure 7: Two conditions which result in $g_{icjc'} = t_{b_i b_j}$.

When i is an output request and the conditions presented in Figure 7 are satisfied, Equations (21) determine the value of $g_{icjc'}$.

$$g_{icjc'} = t_{b_i b_j} \qquad \forall i \in \mathcal{R}, \begin{cases} j \in \mathcal{R} : h_{e_i} < h_{b_j},\ h_{e_i} < h_{b_i},\ \forall c, c' \in \mathcal{C} : \eta_c < \eta_{c'} \\ j \in \mathcal{R} : h_{e_i} > h_{b_j},\ h_{e_i} > h_{b_i},\ \forall c, c' \in \mathcal{C} : \eta_c > \eta_{c'} \end{cases} \tag{21}$$

Constraints (22) and (23) define the value of variable q_{ij} which must equal 1 if request i begins before j's starting time and 0 otherwise.

$$s_i \geq s_j - M_T q_{ij} \qquad \forall i,j \in \mathcal{R} : i \neq j \tag{22}$$
$$s_i \leq s_j + (1 - q_{ij}) M_T \qquad \forall i,j \in \mathcal{R} : i \neq j \tag{23}$$

Assume requests i and j are handled by two separate cranes, c and c'. If the requests are not conflicting, i's starting time does not influence j's, since the requests are assigned to different cranes. If, however, the requests are conflicting, then the starting time of j must consider i's starting time coupled with the position and direction of both cranes. Constraints (24) coordinate the starting times of consecutive requests if the following conditions are satisfied: (i) two different cranes (c and c') are assigned to execute requests i and j, $y_{ic} = y_{jc'} = 1$, (ii) the requests are conflicting, $o_{ij} = 1$, and (iii) request j begins after request i, $q_{ij} = 1$.

$$s_j \geq s_i + g_{icjc'} + st_{cc'} - (4 - y_{ic} - y_{jc'} - o_{ij} - q_{ij}) M_T \qquad \forall i,j \in \mathcal{R}, c, c' \in \mathcal{C}, i \neq j, c \neq c' \tag{24}$$

When the three conditions are satisfied, then request j must begin after the starting time of i plus the waiting time required between requests i and j ($g_{icjc'}$) and the time required for a crane to travel the safety distance between c and c' ($st_{cc'}$).

Constraints (25) ensure the starting time of request i occurs after its release time r_i.

$$s_i \geq r_i \qquad \forall i \in \mathcal{R} \tag{25}$$

When products are stacked on top of each other and the bottom one should be moved, the requests associated with the top products must precede those associated with products situated on the lower levels. A set of precedence constraints, Γ_i, indicates the set of requests which must precede i. Constraints (26) specify these precedence relations and state that request i may only begin after all its preceding requests are finished.

$$s_i \geq s_j + d_j \qquad \forall i \in \mathcal{R}, j \in \Gamma_i \tag{26}$$

The tardiness associated with request i is denoted by $\delta_i \geq 0$. Constraints (27) set the delay of each request to be at least the request's finishing time ($s_i + d_i$) minus its due time, τ_i.

$$\delta_i \geq s_i + d_i - \tau_i \qquad \forall i \in \mathcal{R} \tag{27}$$

All variables, except s_i (starting time of request i) and δ_i (tardiness of request i), are binary:

$$y_{ic} \in \{0,1\} \qquad \forall i \in \mathcal{R}, c \in \mathcal{C} \tag{28}$$
$$n_{ij}, q_{ij}, o_{ij}, o^a_{ij}, o^b_{ij}, o^c_{ij} \in \{0,1\} \qquad \forall i,j \in \mathcal{R} : i \neq j \tag{29}$$
$$s_i, \delta_i \geq 0 \qquad \forall i \in \mathcal{R} \tag{30}$$

The CSP formulation is given by $\mathcal{F}_{CS}$, where the objective function (31) minimises the tardiness of all requests.

$$\mathcal{F}_{CS} \begin{cases} min. & \sum_{i \in \mathcal{R}} \delta_i \\ s.t. & (9) - (30) \end{cases} \tag{31}$$

3.3 Integrated formulation for crane-operated warehouse scheduling problems

This section introduces an integrated continuous-time formulation for the CWSP ($\mathcal{F}_{CWS}$) which considers the location assignment for input requests, the crane assignment for all requests, and the sequencing of the requests per crane. $\mathcal{F}_{CWS}$ includes all constraints associated with the LAP and CSP plus some additional constraints. Since the *input* requests' destinations are undefined (whereas for the CSP, they are determined by first solving the LAP), the following parameters in $\mathcal{F}_{CS}$ become variables in $\mathcal{F}_{CWS}$: h_i^-, h_i^+, d_i and e_i for all requests $i \in \mathcal{R}^I$ and o_{ij}^a when at least one of requests i and j is an *input request*. Note that since these variables are defined only for *input* requests, the definitions in Table 2 remain valid for output requests.

Additional constraints are required to assist in identifying the conflicting requests. Equations (32) and (33) are employed to obtain h_i^- and h_i^+ for request $i \in \mathcal{R}^I$, respectively. Constraints (34) are employed to define the values of o_{ij}^a.

$$h_i^- = \sum_{l \in L: h_l \leq h_{b_i}} h_l \, x_{il} + \sum_{l \in L: h_l > h_{b_i}} h_{b_i} \, x_{il} \qquad \forall i \in \mathcal{R}^I \tag{32}$$

$$h_i^+ = \sum_{l \in L: h_l \leq h_{b_i}} h_{b_i} \, x_{il} + \sum_{l \in L: h_l > h_{b_i}} h_l \, x_{il} \qquad \forall i \in \mathcal{R}^I \tag{33}$$

$$h_i^+ \geq h_j^- - M_L \cdot o_{ij}^a \qquad \forall i, j \in \mathcal{R} : i \neq j \tag{34}$$

Output requests have a fixed duration, while *input* requests' durations depend on the chosen destination. Constraints (35) are employed to compute the duration of *input* requests.

$$d_i = \sum_{l \in L} t_{b_i l} \, x_{il} \quad \forall i \in \mathcal{R}^I \tag{35}$$

As the destinations of *input* requests are decision variables, setting the value of $g_{icjc'}$ when request i is an *input* request requires additional constraints. Equations (36) define the value of $g_{icjc'}$ when crane c handling request i is to the left of crane c' which handles request j. When $h_{e_i} < h_{b_j}$ and $h_{e_i} < h_{b_i}$ then $g_{icjc'} = t_{b_i b_j}$ as in Constraints (21), if $h_{e_i} < h_{b_j}$ and $h_{e_i} > h_{b_i}$ then $g_{icjc'} = d_i - t_{e_i b_j}$ as in Constraints (20), and finally $h_{e_i} > h_{b_j}$ then $g_{icjc'} = d_i + t_{e_i b_j}$ as in Constraints (19).

$$g_{icjc'} = \sum_{l \in L: h_l < h_{b_j}, h_l < h_{b_i}} t_{b_i b_j} x_{il} + \sum_{l \in L: h_l < h_{b_j}, h_l > h_{b_i}} (d_i - t_{e_i b_j}) \, x_{il} + \sum_{l \in L: h_l > h_{b_j}} (d_i + t_{e_i b_j}) \, x_{il}$$
$$\forall i \in \mathcal{R}^I, j \in \mathcal{R}, \forall c, c' \in \mathcal{C} : \eta_c < \eta_{c'} \tag{36}$$

Similarly, Equations (37) set the value of $g_{icjc'}$ for *input* requests in case crane c handling request i is to the right of crane c' which handles request j.

$$g_{icjc'} = \sum_{l \in L: h_l < h_{b_j}} \left(d_i + t_{e_i b_j}\right) x_{il} + \sum_{l \in L: h_l > h_{b_j}, h_l < h_{b_i}} \left(d_i - t_{e_i b_j}\right) x_{il} + \sum_{l \in L: h_l > h_{b_j}, h_l > h_{b_i}} t_{b_i b_j} x_{il}$$
$$\forall i \in \mathcal{R}^I, j \in \mathcal{R}, \forall c, c' \in \mathcal{C} : \eta_c > \eta_{c'} \tag{37}$$

In case of *output* requests, as their destinations are given, the value of $g_{icjc'}$ is obtained in a similar way as in $\mathcal{F}_{\mathcal{CS}}$. Constraints (38), (39) and (40) are modified based on Constraints (19), (20) and (21) respectively to set $g_{icjc'}$ for request $i \in \mathcal{R}^O$ and request $j \in \mathcal{R}$.

$$g_{icjc'} = t_{b_i b_j} \qquad \forall i \in \mathcal{R}^O, \begin{cases} j \in \mathcal{R} : h_{e_i} < h_{b_i}, h_{e_i} < h_{b_j}, \forall c, c' \in \mathcal{C} : \eta_c < \eta_{c'} \\ j \in \mathcal{R} : h_{b_i} < h_{e_i}, h_{b_j} < h_{e_i}, \forall c, c' \in \mathcal{C} : \eta_c > \eta_{c'} \end{cases} \tag{38}$$

$$g_{icjc'} = d_i - t_{e_i b_j} \qquad \forall i \in \mathcal{R}^O, \begin{cases} j \in \mathcal{R} : h_{b_i} < h_{e_i} < h_{b_j}, \forall c, c' \in \mathcal{C} : \eta_c < \eta_{c'} \\ j \in \mathcal{R} : h_{b_i} > h_{e_i} > h_{b_j}, \forall c, c' \in \mathcal{C} : \eta_c > \eta_{c'} \end{cases} \tag{39}$$

$$g_{icjc'} = d_i + t_{e_i b_j} \qquad \forall i \in \mathcal{R}^O, \begin{cases} j \in \mathcal{R} : h_{e_i} > h_{b_j}, \forall c, c' \in \mathcal{C} : \eta_c < \eta_{c'} \\ j \in \mathcal{R} : h_{e_i} < h_{b_j}, \forall c, c' \in \mathcal{C} : \eta_c > \eta_{c'} \end{cases} \tag{40}$$

Another set of additional constraints is required due to implied precedence constraints during the location assignment. Constraints (41) assert that if an input request is assigned atop another input request, the bottom request is placed first. Constraints (42) ensure j is moved after i if it has been assigned to i's origin location.

$$n_{ij} \geq x_{jl} + x_{ik} - 1 \qquad \forall i, j \in \mathcal{R}^I : i \neq j, l \in L, k \in U_l \tag{41}$$
$$n_{ij} \geq x_{jb_i} \qquad \forall i \in \mathcal{R}, j \in \mathcal{R}^I \tag{42}$$

The objective function is a weighted linear combination of the LAP's objectives (Equation (2)) and those of the CSP (Equation (31)). The CWSP formulation is given by $\mathcal{F}_{CWS}$:

$$\mathcal{F}_{CWS} \begin{cases} min. \quad \alpha \left(\sum_{i \in \mathcal{R}^I} \sum_{l \in L} \gamma_{il} x_{il} + \sum_{i \in \mathcal{R}^I} \sum_{j \in \mathcal{R}^I} \omega_{ij} z_{ij} \right) + \beta \sum_{i \in \mathcal{R}} \delta_i \\ s.t. \quad (3) - (12),\ (16) - (18),\ (22) - (30),\ (32) - (42) \end{cases} \tag{43}$$

The CWSP is a complex problem that generalizes the Sequential Ordering Problem (SOP) (Escudero, 1988), which represents a special case of the CWSP scheduling component with a single crane and fixed request destinations. Consequently, the CWSP is considered at least as hard as the SOP. Given the SOP is known to be $\mathcal{NP}$-hard (Montemanni et al., 2009), by consequence, CWSP is an $\mathcal{NP}$-hard problem.

4. Heuristic approach

A local search based algorithm is proposed consisting of constructive and improvement phases for solving the CWSP. During both phases, an *indirect* solution representation capable of reducing the search space is employed (Section 4.1). All the algorithm's components are explained throughout the following sections. The constructive phase (4.2) consists of a greedy constructive heuristic inspired by a set of dispatching rules. During the improvement phase (4.3) a Late Acceptance Hill Climbing (LAHC) meta-heuristic (Burke and Bykov, 2017) is employed which considers several neighbourhood structures (4.4). Figure 8 presents a general overview of the proposed algorithm.

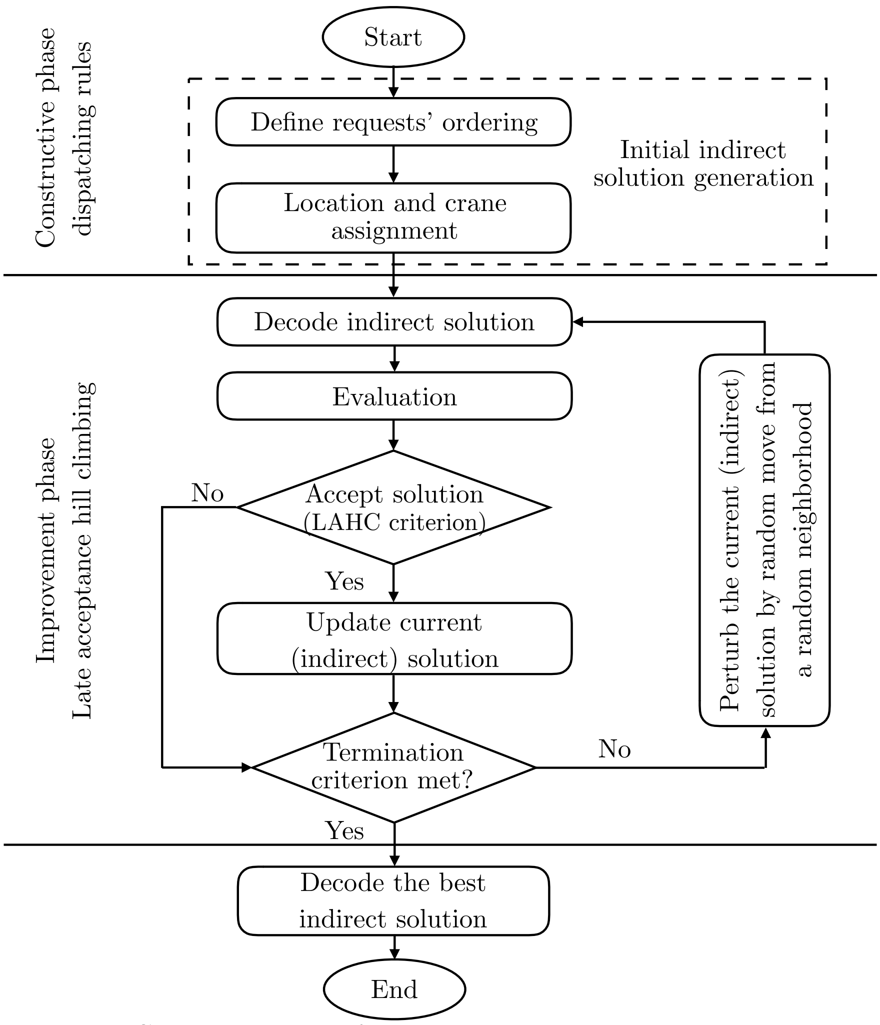


Figure 8: General overview of the heuristic approach

4.1 Solution representation

The local search-based algorithm considers an *indirect* solution representation. Each solution is represented by a list $\mathcal{L}$ of (request, location, crane)-tuples. The actual schedule is produced by a *decoder* which utilizes both the ordering of requests, the locations and crane assignments included in list $\mathcal{L}$.

The decoder is also employed to evaluate solutions by computing their storage cost and total tardiness. Whereas the total tardiness can be obtained by applying Equation (2), computing the total tardiness is less straightforward. It requires all requests' starting times, which determine on the requests' execution order in $\mathcal{L}$ in such a way that all conflict-related constraints are satisfied.

The decoder is presented by Algorithm 1. For each tuple (i, e_i, c) in $\mathcal{L}$, request i has its starting time s_i initialized as its release time (lines 1-2). If crane c was previously assigned to another request, the starting time s_i is set to be after the execution of the crane's previous request plus the time required for the crane to begin executing request i (lines 3-5). To handle conflicts with requests assigned to other cranes, the algorithm loops over the schedule of all other cranes from end to beginning (lines 6-8). In case request i conflicts with a request k of crane c''s schedule, s_i is set to be at least $s_k + g_{kc'ic} + st_{cc'}$ (lines 9-10), where $g_{kc'ic}$ is the waiting time defined in Section 3. Once the algorithm finds a conflicting request k in crane c''s schedule, it is unnecessary to proceed checking, as all other requests assigned to c' are scheduled earlier than k. The algorithm then moves on to the next crane, if any remain. Therefore, the decoder will execute only $O(|\mathcal{R}| \times |\mathcal{C}|)$ operations in the best case. Since in most applications the number of cranes $|C|$ may be fixed as a constant, in the best case the decoder runs in linear time. In the worst case, however, $O(|\mathcal{R}|^2)$ operations may be required by the decoder. After the schedules of all cranes are checked, the minimum value for s_i is calculated and the algorithm includes request i into crane c's schedule (lines 12). Once all starting times are computed, the tardiness can be easily calculated employing Equations (27) and (31).

One of the advantages of the indirect solution representation proposed is that it prevents the generation of a class of unattractive solutions which include avoidable idle times between operations. Indeed, given that the indirect representation is no more than a sequence, the decoder will produce a left-active schedule which is always better than alternative schedules based on this sequence, but which include idle times.

Moreover, the indirect solution is simple to modify, requiring only a few verifications to guarantee feasibility. Its main disadvantage lies in the additional $O(|R|^2)$ operations required for decoding and evaluating solutions.

4.2 Constructive heuristic

An initial solution is constructed by a greedy algorithm which applies a set of dispatching rules. Algorithm 2 details this procedure. A directed acyclic graph is considered where nodes represent

Algorithm 1: Decoding an *indirect* solution

Input: Ordered list $\mathcal{L}$ of tuples {request, location, crane}

1 **foreach** tuple $(i, e_i, c) \in \mathcal{L}$ **do**
2     $s_i \leftarrow r_i$ // minimum starting time s_i is the release time r_i
3   **if** crane c's schedule is not empty **then**
4         $j \leftarrow$ last request in crane c's schedule
5         $s_i \leftarrow max(s_i\ ,\ s_j + d_j + t_{e_j b_i})$ // s_i must consider when crane c is available at position b_i
6   **foreach** crane $c' \in \mathcal{C}, c' \neq c$ **do**
7   let schedule S^{-1} be the reverse of crane c''s schedule
8   **foreach** request $k \in S^{-1}$ **do**
9   **if** request i conflicts with request k **then**
10                 $s_i \leftarrow max(s_i\ ,\ s_k + g_{kc'ic} + st_{cc'})$ // s_i must also take conflicting moves into account
11   **break** foreach-loop
12   add request i (ending at location e_i) into crane c's schedule

requests and arcs directing from r_i to r_j indicate precedence constraints (r_i must precede r_j). In this graph, requests are first sorted topologically. By sorting the directed acyclic graph topologically, for every directed arc (r_i, r_j), r_i precedes r_j in the ordering ensuring that the precedence constraints are respected. Requests within the same topological level are sorted by their release times. This strategy defines the initial ordering of the requests (line 1). The solution is initialized as an empty list (line 2) and, afterwards, two steps are applied for each request (lines 3-6). First, the available location which has the lowest storage cost is assigned to *input* requests (lines 4-5). Note that *output* requests have preassigned locations and therefore do not require such an assignment. In the second step, requests are assigned to the nearest available crane (line 6). Both the requests' starting times and the cost of the solution corresponding with $\mathcal{L}$ are computed employing the decoder presented in Algorithm 1.

Algorithm 2: Constructive CWSP algorithm

1 $R \leftarrow$ list of requests sorted topologically and chronologically
2 $\mathcal{L} \leftarrow$ empty list // or empty ordered set
3 **foreach** request $i \in R$ **do**
4   **if** $i \in \mathcal{R}^I$ **then** // checking whether request i is an *input* request (these requests require assigning a location)
5         $e_i \leftarrow$ the nearest available location to the origin of input request i
6     $c \leftarrow$ nearest crane to b_i // greedy criterion: nearest crane to the origin of request i is selected
7   insert (i, e_i, c) to $\mathcal{L}$
8 **return** list of tuples $\mathcal{L}$

4.3 Late Acceptance Hill Climbing

The LAHC meta-heuristic represents an extension of the greedy hill-climbing algorithm which compares the candidate solution against the solution which was 'current' l iterations before. Consequently, the meta-heuristic permits the acceptance of worsening solutions, thus avoiding local optima. This study employs the LAHC meta-heuristic presented in Algorithm 3 which requires the following arguments: (*i*) initial solution s_0, (*ii*) parameter l, (*iii*) set of neighbourhoods $\mathcal{N}$, (*iv*) maximum number of iterations without improvement it^{max} and (*v*) *timeout*, which indicates the runtime of LAHC.

The LAHC meta-heuristic maintains a fixed-length list v containing objective function values of the solutions visited during the last l iterations. Initially, all v elements are set to the initial solution's objective value, given by $f(s_0)$ (line 1). Next, current solution s, best solution s^* and index i are initialized (lines 2-3). At each iteration a new candidate solution s' is generated by applying a randomly generated move from a randomly selected neighbourhood to the current solution s (lines 5-6). The candidate solution's objective value, $f(s')$, is compared against v_i and the current solution's value $f(s)$ (line 7). s' is accepted (line 12) to replace the current solution if its objective value is less than or equal to v_i or $f(s)$. If the candidate solution s' has a better objective value than the best solution s^* generated thus far, it replaces s^* (lines 13-14). If the objective value of s' is greater than v_i or $f(s)$, the number of iterations without improvement, it, increments by one (lines 15-16). Finally, v_i and i are updated: $v_i \leftarrow f(s)$ (replacing the oldest value), and i is set to point to the next position of list l (lines 17-18). Note that i acts as a cyclic

Algorithm 3: Late acceptance hill climbing algorithm

Input: Initial solution s_0, list size l, set of neighbourhoods $\mathcal{N}$, maximum number of consecutive iterations without improvement it^{max}, and runtime limit *timeout*

1 $v_i \leftarrow f(s_0) \;\; \forall\, i \in \{0, ..., l-1\}$
2 $s^* \leftarrow s \leftarrow s_0$
3 $i \leftarrow it \leftarrow 0$
4 **while** $it < it^{max}$ **and** *elapsed time* $<$ *timeout* **do**
5   Select a random neighbourhood $N \in \mathcal{N}$
6     $s' \leftarrow$ random neighbour solution $m \in N(s)$
7   **if** $f(s') \leq v_i$ **or** $f(s') \leq f(s)$ **then**
8   **if** $f(s') < f(s)$ **then**
9             $it \leftarrow 0$
10   **else**
11             $it \leftarrow it + 1$
12         $s \leftarrow s'$
13   **if** $f(s') < f(s^*)$ **then**
14             $s^* \leftarrow s$
15   **else**
16         $it \leftarrow it + 1$
17     $v_i \leftarrow f(s)$
18     $i \leftarrow (i + 1) \; mod \;\; l$
19 **return** s^*

pointer. This iterative process repeats until the elapsed time reaches *timeout* or the number of iterations without improvement reaches it^{max}. The latter criterion prevents the heuristic from continuing the search in situations whre no more improvements are generated during a long period of time. Finally, the best solution obtained is then returned (line 19).

4.4 Neighbourhood structures

Ten neighbourhoods were developed to explore the CWSP's solution space. The neighbourhoods are grouped into two categories: (*i*) Location assignment and (*ii*) Crane scheduling neighbourhoods. All ten neighbourhoods operate over the list of tuples $\mathcal{L}$ and therefore modify only the indirect solution. Infeasible solutions may be obtained by applying some of these neighbourhood operators to a solution. For instance, both assigning a product atop an empty location and ignoring the precedence constraints when changing the requests' order result in infeasibility. Such infeasible solutions are discarded during the search.

The ten neighbourhoods are detailed as follows.

Location assignment neighbourhoods

- Location Re-assignment (LR): a random *input* request is selected and its destination location is replaced with another randomly selected available location.

- Location Swap (LS): two random input requests are selected and their destination locations are swapped.
- Greedy Location Assignment (GLA): a set of input requests is randomly selected and their location assignments are removed. Then, all possible locations are considered and the requests are greedily assigned to the **lowest-cost** location.

Crane scheduling neighbourhoods

- Crane Re-assignment (CR): a set of three requests is selected randomly and all crane assignment combinations are enumerated. The resulting solution is the one with the best quality among the enumerated solutions.
- Order Swap (OS): two requests are randomly selected and their tuples are swapped, changing their execution order.
- Random Insertion (RI): a random tuple is removed and re-inserted into a random position in the list.
- Random Best Insertion (RBI): a request is randomly selected and its tuple is inserted into the lowest-cost position within the assigned crane's schedule. Note that this neighbourhood requires $O(|R|)$ operations to identify the lowest-cost position.
- Nearest Location Assignment (NLA): a set of input requests is randomly selected and their location assignments are removed. Then, all possible locations are considered and each request is greedily assigned to the location nearest to its origin.

  This neighbourhood is employed as a crane scheduling neighborhood since it reduces the duration of requests and, as a consequence, reduces the potential risk of generating conflicting requests.
- Best Order Permutation (BOP): a range of requests in a crane's schedule is randomly selected; their tuples are subsequently removed and the best permutation of the selected tuples, determined by enumeration, is inserted into the list.

  This neighbourhood has exponential complexity and therefore the range must be limited to prevent prohibitive runtimes.
- Moving Best Order Permutation (MBOP): This neighbourhood begins from the first tuple in the list $\mathcal{L}$ and executes the BOP move within a range of three tuples. The procedure moves forward in $\mathcal{L}$ by one tuple and executes the BOP move for next three tuples. It ends after $|\mathcal{L}| - 2$ iterations by executing the BOP move for the last three tuples in the list.

5. Computational study

This section investigates the impact of integrating the location assignment and crane scheduling problems. The performance of both the formulations and heuristic are assessed across different scenarios: sequential and integrated.

The sequential approach, on the one hand, solves two problems. It begins by solving the LAP, after which the assignments obtained are fixed when solving the CSP. The integrated approach, on the other hand, solves only one large problem: the CWSP.

Table 3: Summary of sequential and integrated approaches

	Formulation	Objective Function	LAHC Neighbourhoods
Sequential approach	$\mathcal{F}_{LA}$ $\mathcal{F}_{CS}$	Equation (2) Equation (31)	LR, LS, GLA CR, OS, RI, RBI, BOP, MBOP
Integrated approach	$\mathcal{F}_{CWS}$	Equation (43)	LR, LS, GLA, CR, OS, RI, RBI, NLA, BOP, MBOP

Table 3 summarizes the formulations, objective functions and neighbourhood structures employed by each approach. The sequential approach employs formulations $\mathcal{F}_{LA}$ and $\mathcal{F}_{CS}$ to solve the LAP and CSP, respectively. It also utilizes Equation (2) and *location assignment neighbourhoods* to address the LAP heuristically. To solve the CSP by LAHC, the sequential approach employs Equation (31) and *crane scheduling neighbourhoods*. Whereas, the integrated approach solves formulation $\mathcal{F}_{CWS}$ and considers Equation (43) and all the neighbourhoods when employing LAHC. Table 3 summarizes the objective functions and neighbourhoods employed in the heuristic approaches to both the sequential and integrated problems.

The remainder of this section is organized as follows. First Section 5.1 presents the set of benchmark instances considered throughout the experiments. The comparison of the sequential and integrated approaches by employing the MIP formulations is presented in Section 5.2, while Section 5.3 presents the results of the sequential and integrated approaches employing the heuristic. Finally, Section 5.4 presents a discussion on the weights α and β employed within the objective function (see Equation (1)) in addition to an analysis of the instances.

5.1 Instances

A set of benchmark instances was generated in correspondance with the data obtained from a real-world warehouse. These instances are available online[1] to enable transparent comparison of the proposed formulations and algorithms. Four characteristics were considered for the instance generation:

Number of requests: the benchmark set includes instances with *five* levels denoting the number of requests: 10, 20, 30, 50 and 70.

Storage size: the dataset considers three storage sizes. *Small* size storage areas have 250 (10×25) locations per level. *Medium* size storage areas have 525 (15×35) locations per level and *large* storage areas 1000 (20×50) locations per level. A large storage area may have up to 5000 locations. Based on the data obtained from the real-world problem, a realistic storage area contains approximately 3000 locations.

Maximum stacking level: this limit is imposed for each storage area depending on various criteria such as the product type and safety requirements. Three stacking levels are considered: *one*, *three* and *five* levels, where *one* denotes the ground level.

Storage load: this value is defined as a percentage indicating the initial occupancy level of the storage area. Three *load factor* values are considered: 30%, 50% and 70%.

[1]CWSP data instances, solutions & validator to be made publicly available in a data repository after paper acceptance due to incompatibility with double-blind review process. The data repository is available at a public url shared with the editor through the title-page.

The value of each attribute is reflected in the instance name, which indicates, from left to right: (i) number of requests, (ii) storage size, represented by the letters s, m and l signifying *small*, *medium* and *large*, respectively, (iii) maximum stacking level, and (iv) storage load. Instance name `50s_1_30`, for example, indicates an instance with 50 requests, a small yard, maximum stacking level 1, and storage load equal to 30%.

Release times were generated according to a Poisson distribution with a fixed "*average frequency*" within the scheduling horizon. The scheduling horizon is computed by multiplying "*Number of requests*" by "*average frequency*". Due times were calculated as the sum of a request's "*release time*" and its "*time window*", wherein the "*time window*" was generated according to a log-normal distribution.

The number of yard products is calculated based on the *yard size* and *yard load.* For each stored product a location in the yard is randomly selected.

The storage cost of assigning an input request to a neighbouring location of any product is generated according to a uniform distribution. The cranes' travel times are measured by the Chebychev distance (the maximum of the lateral and longitudinal distance). It is assumed that the lateral and longitudinal speed of the cranes is identical, and thus the travel time may be substituted by the Chebychev distance. Table 4 summarizes the parameters of the distributions used in the instance generator.

Table 4: Instance generator distributions and parameters

Instance Parameter	Distribution	Distribution Parameter
Release time	Poisson	mean = 26.5
Time windows	Log normal	mean = 0.0666 * storage area's length standard deviation = 0.1 * storage area's width
Stored product location	Uniform	range = locations in the storage area
Storage cost	Uniform	range = [0, 50)

The cranes are initially lined-up according to the required safety distance. The crane's travel speed is set to one time unit per horizontal coordinate.

5.2 *Comparison of sequential and integrated approaches by MIP formulations*

In the sequential approach, the LAP completely disregards the possible impact of attaining good solutions for the CSP. The integrated approach, however, enables a trade-off between the LAP and the CSP by setting different values for their corresponding weights α and β in the objective function, Equation (1). Two settings are considered: (i) $\alpha >> \beta$, making the integrated approach put the highest priority to the LAP. This enables determining the benefit of an integrated approach over the sequential approach where both primarily focus on optimizing the storage cost. (ii) $\alpha = \beta$, which considers equal importance for storage cost and delays.

The mathematical formulations were solved by Gurobi Optimizer 7.5 and run on an Intel® Xeon® CPU E5-2650 v2 @ 2.6GHz with one hour runtime limit. Table 5 reports the results obtained by solving the sequential ($\mathcal{F}_{LA}$+$\mathcal{F}_{CS}$) and integrated ($\mathcal{F}_{CWS}$) formulations. The results of unsolved instances (no integer solution found within the runtime limit) are excluded from the table for brevity. Location assignment evaluation (E_{LA}), crane scheduling evaluation (E_{CS}) and total weighted cost (E_τ), both when $\alpha = \beta$ and $\alpha >> \beta$, are compared. Emboldened numbers indicate that optimal solutions were obtained. When both the LAP and CSP are solved to optimality only E_τ is emboldened.

Table 5: Computational results obtained by solving the mathematical formulations (considered scenarios: $\alpha = 100000$, $\beta = 1$ ($\alpha >> \beta$) and $\alpha = \beta = 0.50$).

Instance	$\mathcal{F}_{LA} + \mathcal{F}_{CS}$				$\mathcal{F}_{CWS}$ ($\alpha >> \beta$)			$\mathcal{F}_{CWS}$ ($\alpha = \beta$)		
	E_{LA}	E_{CS}	$E_\tau(\alpha >> \beta)$	$E_\tau(\alpha = \beta)$	E_{LA}	E_{CS}	E_τ	E_{LA}	E_{CS}	E_τ
10s_1_30	0	267.03	**2.670E+02**	**133.51**	0	55.13	**5.513E+01**	1	51.53	**26.26**
10m_1_30	0	478.41	**4.784E+02**	**239.20**	0	37.62	**3.762E+01**	0	37.62	**18.81**
10l_1_30	0	515.69	**5.157E+02**	**257.84**	0	76.82	**7.682E+01**	0	76.82	**38.41**
10s_1_50	0	229.25	**2.293E+02**	**114.63**	0	123.93	**1.239E+02**	11	78.92	**44.96**
10m_1_50	0	219.18	**2.192E+02**	**109.59**	0	56.45	**5.645E+01**	7	40.07	**23.53**
10l_1_50	0	447.85	**4.479E+02**	**223.92**	0	44.49	**4.449E+01**	4	32.49	**18.24**
10s_1_70	132	186.97	**1.320E+07**	**159.49**	132	135.97	**1.320E+07**	156	59.25	**107.62**
10m_1_70	48	348.86	**4.800E+06**	**198.43**	48	449.55	**4.800E+06**	86	185.92	**135.96**
10l_1_70	11	271.44	**1.100E+06**	**141.22**	11	148.17	**1.100E+06**	25	73.78	**49.39**
10s_3_30	13	352.10	**1.300E+06**	**182.55**	13	179.72	**1.300E+06**	49	76.07	62.53
10m_3_30	1	354.66	**1.004E+05**	**177.83**	1	107.51	**1.001E+05**	9	70.37	39.68
10l_3_30	0	295.28	**2.953E+02**	**147.64**	0	79.51	**7.951E+01**	-	-	-
10s_3_50	53	317.77	**5.300E+06**	**185.39**	53	201.77	**5.300E+06**	64	109.83	86.91
10m_3_50	31	409.77	**3.100E+06**	**220.39**	31	332.35	**3.100E+06**	41	126.77	83.88
10l_3_50	9	454.43	**9.005E+05**	**231.72**	9	145.71	**9.001E+05**	12	85.91	**48.95**
10s_3_70	130	392.07	**1.300E+07**	**187.73**	130	279.13	**1.300E+07**	190	102.40	146.20
10m_3_70	64	433.66	**6.400E+06**	**248.83**	64	492.57	**6.400E+06**	93	77.90	85.45
10l_3_70	98	441.02	**9.800E+06**	**269.51**	98	566.37	**9.800E+06**	134	118.11	126.05
10s_5_30	18	322.97	**1.800E+06**	**170.48**	18	182.53	**1.800E+06**	35	93.23	64.11
10m_5_30	653	**477.18**	**6.530E+07**	565.09	9	275.46	9.002E+05	24	76.36	50.18
10l_5_30	464	**457.64**	**4.640E+07**	460.82	6	244.62	**6.002E+05**	16	206.88	111.44
10s_5_50	66	435.57	**6.600E+06**	**250.79**	66	292.55	6.600E+06	-	-	-
10m_5_50	**25**	-	-	-	25	294.46	2.500E+06	42	196.59	119.30
10l_5_50	13	602.07	**1.301E+06**	**307.54**	13	159.11	**1.300E+06**	21	82.93	51.96
10s_5_70	83	247.93	**8.300E+06**	**165.64**	83	247.93	**8.300E+06**	100	91.85	95.92
10m_5_70	59	419.26	**5.900E+06**	**239.13**	59	263.06	**5.900E+06**	75	211.27	143.13
10l_5_70	43	541.38	**4.301E+06**	**292.19**	43	418.37	**4.300E+06**	51	87.02	69.01
20s_1_30	0	1221.04	**1.221E+03**	**610.52**	0	160.38	1.603E+02	-	-	-
20m_1_30	0	1821.80	**1.822E+03**	**910.90**	0	225.29	2.252E+02	1	235.90	118.45
20l_1_30	0	2171.42	**2.171E+03**	**1085.71**	0	198.40	**1.984E+02**	0	261.18	130.59
20s_1_50	4	1412.59	**4.014E+05**	**708.30**	4	425.09	4.004E+02	-	-	-
20m_1_50	0	1373.24	**1.373E+03**	**686.21**	0	445.37	4.453E+02	9	351.42	180.21
20l_1_50	0	1851.12	**1.851E+03**	**925.56**	0	155.85	1.558E+02	8	118.23	63.11
20s_1_70	448	801.34	**4.480E+07**	**624.67**	448	692.26	4.480E+07	480	196.16	338.08
20m_1_70	241	2396.80	**2.410E+07**	**1318.90**	241	1287.71	2.410E+07	333	640.11	486.55
20l_1_70	77	1210.96	**7.701E+06**	**643.98**	77	758.37	7.700E+06	88	337.18	212.59
20s_3_30	54	1328.04	5.401E+06	691.02	-	-	-	-	-	-
20m_3_30	995	2031.35	9.950E+07	1513.18	15	740.32	1.500E+06	64	676.63	370.31
20l_3_30	648	-	-	-	0	277.50	2.775E+02	37	417.42	227.21
20s_3_50	168	1434.26	1.680E+07	799.13	-	-	-	-	-	-
20m_3_50	99	2203.80	9.902E+06	1151.40	-	-	-	181	366.17	273.58
20l_3_50	1344	1852.22	1.344E+08	1598.11	45	1123.48	4.501E+06	49	578.48	313.74
20s_3_70	423	1552.35	4.230E+07	987.67	424	2902.26	4.240E+07	528	390.82	459.41
20m_3_70	**168**	1491.69	1.680E+07	829.84	168	1407.24	1.680E+07	-	-	-
20l_3_70	274	2989.96	2.740E+07	1631.98	-	-	-	-	-	-
20s_5_30	923	939.96	9.230E+07	931.48	68	1630.26	6.801E+06	-	-	-
20m_5_30	886	-	-	-	43	2017.09	4.302E+06	-	-	-
20s_5_50	1527	1051.42	1.527E+08	1289.21	-	-	-	-	-	-
20m_5_50	1450	-	-	-	87	2062.80	8.702E+06	-	-	-
20s_5_70	330	1499.62	3.300E+07	914.81	-	-	-	-	-	-
20m_5_70	249	1435.31	2.490E+07	842.15	385	1738.69	3.850E+07	-	-	-
20l_5_70	129	-	-	-	-	-	-	158	812.23	485.11

(continued on next page)

Table 5: Computational results obtained by solving the mathematical formulations (continued).

Instance	$\mathcal{F}_{LA} + \mathcal{F}_{CS}$				$\mathcal{F}_{CWS}$ $(\alpha >> \beta)$			$\mathcal{F}_{CWS}$ $(\alpha = \beta)$		
	E_{LA}	E_{CS}	$E_{\tau}(\alpha >> \beta)$	$E_{\tau}(\alpha = \beta)$	E_{LA}	E_{CS}	E_{τ}	E_{LA}	E_{CS}	E_{τ}
`30s_1_30`	0	3495.86	**3.496E+03**	**1747.93**	0	668.80	6.688E+02	-	-	-
`30m_1_30`	0	4501.77	**4.502E+03**	**2250.88**	0	921.33	9.213E+02	-	-	-
`30l_1_30`	**0**	4911.02	4.911E+03	2455.51	0	634.49	6.344E+02	-	-	-
`30s_1_50`	98	3299.06	**9.803E+06**	**1698.53**	98	1303.66	9.801E+06	210	729.63	469.81
`30m_1_50`	19	3742.17	1.904E+06	1880.58	19	2616.36	1.902E+06	75	1880.72	977.86
`30l_1_50`	0	4683.93	**4.684E+03**	**2737.80**	0	724.51	7.245E+02	-	-	-
`30s_1_70`	934	2416.98	9.340E+07	1675.49	937	2667.16	9.370E+07	-	-	-
`30m_1_70`	544	5919.16	5.441E+07	3231.58	554	3078.50	5.540E+07	631	2046.16	1338.58
`30l_1_70`	249	3310.89	**2.490E+07**	**1779.94**	-	-	-	268	1623.10	945.55
`30s_3_30`	1585	2858.33	1.585E+08	2221.66	-	-	-	-	-	-
`30s_3_50`	330	3643.15	3.300E+07	1986.57	-	-	-	-	-	-
`30s_3_70`	821	3058.59	8.210E+07	1939.79	840	3613.93	8.400E+07	-	-	-
`30m_3_70`	371	3394.37	3.710E+07	1882.68	-	-	-	-	-	-
`30l_3_70`	2589	-	-	-	-	-	-	684	9339.08	5019.04
`30s_5_70`	756	4492.94	7.560E+07	2624.47	-	-	-	-	-	-
`50s_1_30`	0	16686.92	**1.669E+04**	**8343.46**	-	-	-	-	-	-
`50m_1_30`	0	17645.35	1.765E+04	8822.67	0	8264.20	8.264E+03	-	-	-
`50s_1_50`	401	12290.55	4.011E+07	6345.77	-	-	-	-	-	-
`50m_1_50`	250	13869.28	2.501E+07	7059.64	-	-	-	-	-	-
`50l_1_50`	11	15907.70	1.116E+06	7959.35	-	-	-	146	9867.24	5006.62
`50s_1_70`	1743	11544.01	1.743E+08	6643.50	1772	6898.15	1.772E+08	2370	4006.59	3188.29
`50m_1_70`	1017	19637.64	1.017E+08	10327.32	-	-	-	-	-	-
`50l_1_70`	599	20988.08	5.992E+07	10793.54	-	-	-	-	-	-
`50s_3_70`	3904	12233.50	3.904E+08	8068.75	-	-	-	-	-	-
`70s_1_30`	67	19600.96	6.720E+06	9833.98	-	-	-	-	-	-
`70s_1_50`	1436	20060.26	1.436E+08	10748.13	-	-	-	-	-	-
`70s_1_70`	3225	16797.21	3.225E+08	10011.10	-	-	-	-	-	-
`70m_1_70`	2023	29211.91	2.023E+08	15617.45	-	-	-	-	-	-
`70l_1_70`	1332	41511.74	1.332E+08	21421.87	-	-	-	-	-	-

The results indicate that the sequential approach yields better LAP solutions for only five instances, `20m_5_70`, `30s_1_70`, `30m_1_70`, `30s_3_70` and `50s_1_70`. This can be explained by the observation that in these two cases, none of the alternative approaches finds an optimal solution. For the remaining instances, the value of E_{LA} obtained by $\mathcal{F}_{CWS}(\alpha >> \beta)$ is better than or equal to the value of E_{LA} obtained by $\mathcal{F}_{LA} + \mathcal{F}_{CS}$, while $\mathcal{F}_{CWS}(\alpha >> \beta)$ always finds a better crane scheduling solution. However, $\mathcal{F}_{CWS}(\alpha >> \beta)$ was only able to find feasible solutions for two large instances (number of requests greater than 50). $\mathcal{F}_{CWS}(\alpha >> \beta)$ may change location assignments and is therefore able to find better crane scheduling while still achieving the same quality of location assignments. This shows how by integrating the LAP and the CSP, better location assignments and crane schedules are achievable. Generating better crane schedules by $\mathcal{F}_{CWS}(\alpha >> \beta)$ may require compromises with respect to computational runtimes. The results show that $\mathcal{F}_{CWS}(\alpha >> \beta)$ tends to take more time to achieve optima for small instances. However for larger instances both approaches take the entire runtime.

When α is equal to β, it is evident that the E_{LA} values obtained by $\mathcal{F}_{LA}+\mathcal{F}_{CS}$ and $\mathcal{F}_{CWS}(\alpha >> \beta)$ are lower than those obtained by $\mathcal{F}_{CWS}(\alpha = \beta)$. However, E_{τ} is significantly lower in $\mathcal{F}_{CWS}(\alpha = \beta)$ than in $\mathcal{F}_{LA} + \mathcal{F}_{CS}(\alpha = \beta)$. Although $\mathcal{F}_{CWS}(\alpha >> \beta)$ is capable of finding good crane schedules while achieving high quality location assignments, the E_{CS} obtained by $\mathcal{F}_{CWS}(\alpha = \beta)$ is lower. There are only two instances, `20l_3_30` and `20l_1_30`, where $\mathcal{F}_{CWS}(\alpha >> \beta)$ achieved both better location assignments and crane scheduling within the available runtime. For the remaining instances, $\mathcal{F}_{CS}(\alpha = \beta)$ compromises as regards location assignment to achieve better crane schedules. The number of instances solved to optimality decreases as the number of requests grows.

The detailed table of results including the optimality gap and computation time (*cpu*) of both approaches is presented in the Appendix (Table A.8).

5.3 Comparison of sequential and integrated approaches by the proposed heuristics

The heuristic was implemented in C++11. All experiments were executed for maximum 300 seconds or maximum 10,000 consecutive iterations without improvement for all 135 instances. The heuristic's parameters were tuned using the irace R-package which implements the iterated racing procedure for automatic algorithm configuration (López-Ibáñez et al., 2016) with a budget of 4,000 runs. The purpose of irace is to automate the task of configuring an optimization algorithm's parameters. It generates and tests a sample of parameter configurations for a given optimization algorithm on a set of instances. When sufficient statistical evidence is collected (by means of a Friedman test) that a certain parameter configuration is outperformed by others, it is discarded so as to focus on the remaining configurations. The best-performing configurations are reported.

Table 6 summarizes the parameters by their role, type, range and tuned values for both approaches (sequential and integrated).

Table 6: Tuning parameters by irace package

Parameter	Role	Type	Range	irace results	
				integrated	*sequential*
l_{cws}	$\frac{\text{Size of the late acceptance list for CWSP LAHC}}{\text{number of requests}}$	integer	(10, 250)	56	-
l_{lap}	$\frac{\text{Size of the late acceptance list for LAP LAHC}}{\text{number of requests}}$	integer	(10, 250)	-	174
l_{csp}	$\frac{\text{Size of the late acceptance list for CSP LAHC}}{\text{number of requests}}$	integer	(10, 250)	-	104
p_{lr}	Weight of using LR neighbourhood	real	(0.0, 1.0)	0.47	0.13
p_{ls}	Weight of using LS neighbourhood	real	(0.0, 1.0)	0.18	0.07
p_{gla}	Weight of using GLA neighbourhood	real	(0.0, 1.0)	0.40	0.87
p_{cr}	Weight of using CR neighbourhood	real	(0.0, 1.0)	0.18	0.59
p_{os}	Weight of using OS neighbourhood	real	(0.0, 1.0)	0.34	0.36
p_{ri}	Weight of using RI neighbourhood	real	(0.0, 1.0)	0.08	0.22
p_{rbi}	Weight of using RBI neighbourhood	real	(0.0, 1.0)	0.47	0.64
p_{nla}	Weight of using NLA neighbourhood	real	(0.0, 1.0)	0.70	-
p_{bop}	Weight of using BOP neighbourhood	real	(0.0, 1.0)	0.52	0.38
p_{mbo}	Weight of using $MBOP$ neighbourhood	real	(0.0, 1.0)	0.07	0.18

Table 7 reports the results of the three heuristics when the values of α and β are equal to 0.5 ($\alpha = \beta = 0.5$). The location assignment evaluation (E_{LA}), crane scheduling evaluation (E_{CS}) and total cost ($E_{\mathcal{T}}$) obtained by the three heuristics are compared. The heuristic was run five times with different random seeds, and the average of these results is reported. The improvement of the integrated and sequential heuristics over the constructive heuristic (G), the upper bound (UB) defined as the best feasible solution obtained by either the sequential or integrated formulation, and the relative optimality gap (gap) and the computational times (*cpu*) are reported. The optimality gap is measured by comparing $E_{\mathcal{T}}$ obtained by the integrated heuristic and the lower bound generated by the mathematical formulations. The gaps may be relatively large as the lower bounds are weak due to employing large numbers (M_L and M_T) in the mathematical formulations. When $\mathcal{F}_{CWS}(\alpha = \beta)$ was unable to find a lower bound due to an out-of-memory error, OOM is reported. Emboldened numbers indicate that optimum solutions were obtained.

The results show how the integrated heuristics find high quality solutions and are even able to find optimum solutions for the instances with ten requests and stacking level equal to one, in a short amount of time. There are only two instances `10s_1_70` and `10m_1_70` where the integrated heuristic was unable to find the optimum solution. Nevertheless, the optimality gaps are only 1.1% and 4.8%, respectively. There are instances for which the sequential heuristic finds better solutions than the optimum obtained by $\mathcal{F}_{LA} + \mathcal{F}_{CS}$. The explanation for these interesting results lies in how the sequential heuristic finds location assignments which enable better crane scheduling.

Table 7: Computational results of Constructive, Sequential and Integrated heuristics ($\alpha = \beta = 0.5$)

Instance	Dispatching rules			Sequential					Integrated						Best MIP bounds	
	E_{LA}	E_{CS}	E_τ	E_{LA}	E_{CS}	E_τ	*cpu*	G%	E_{LA}	E_{CS}	E_τ	*cpu*	G%	*gap*	UB	LB
10s_1_30	0	108.0	54.0	0	59.9	30.0	3.7	44.5	1	51.5	**26.3**	4.9	51.4	0.0	26.3	26.3
10m_1_30	0	234.1	117.1	0	37.6	**18.8**	6.1	83.9	0	37.6	**18.8**	3.5	83.9	0.0	18.8	18.8
10l_1_30	0	384.0	192.0	0	79.8	39.9	10.5	79.2	0	76.8	**38.4**	6.5	80.0	0.0	38.4	38.4
10s_1_50	0	235.8	117.9	0	137.9	69.0	4.2	41.5	11	79.0	**45.0**	9.2	61.8	0.0	45.0	45.0
10m_1_50	0	366.8	183.4	0	96.3	48.1	7.7	73.8	7	40.1	**23.5**	7.8	87.2	0.0	23.5	23.5
10l_1_50	0	350.4	175.2	0	45.4	22.7	13.0	87.0	1	35.5	**18.2**	8.1	89.6	0.0	18.2	18.2
10s_1_70	180	767.9	474.0	132	158.4	145.2	4.4	69.4	150	67.7	108.8	4.8	77.0	1.1	107.6	107.6
10m_1_70	48	1031.6	539.8	48	405.8	226.9	9.1	58.0	86	199.8	142.9	10.2	73.5	4.8	136.0	136.0
10l_1_70	11	843.8	427.4	11	166.1	88.5	14.9	79.3	25	73.8	**49.4**	29.3	88.4	0.0	49.4	49.4
10s_3_30	13	324.8	168.9	13	230.8	121.9	9.0	27.8	50	86.1	68.0	13.4	59.7	40.8	62.5	40.3
10m_3_30	1	400.8	200.9	1	185.9	93.5	17.8	53.5	6	76.4	41.2	24.8	79.5	81.1	39.7	7.8
10l_3_30	0	700.8	350.4	0	98.5	49.3	32.2	85.9	3	60.3	31.7	22.1	91.0	78.2	147.6	6.9
10s_3_50	58	538.0	298.0	53	275.2	164.1	11.1	44.9	60	111.5	85.8	12.9	71.2	48.8	86.9	43.9
10m_3_50	68	686.9	377.4	31	333.3	182.2	24.8	51.7	42	90.6	66.3	54.1	82.4	75.6	83.9	16.2
10l_3_50	9	1181.3	595.2	9	258.0	133.5	48.4	77.6	12	85.9	**49.0**	82.9	91.8	0.0	49.0	49.0
10s_3_70	150	1309.0	729.5	130	304.4	217.2	12.0	70.2	168	117.4	142.7	8.1	80.4	36.9	146.2	90.0
10m_3_70	64	1210.5	637.2	64	273.8	168.9	25.5	73.5	79	104.8	91.9	69.0	85.6	26.3	85.5	67.7
10l_3_70	106	697.3	401.7	98	321.4	209.7	57.2	47.8	115	112.8	113.9	102.3	71.6	50.4	126.1	56.5
10s_5_30	18	509.4	263.7	18	201.5	109.8	36.6	58.4	47	84.7	65.9	25.9	75.0	67.2	64.1	21.6
10m_5_30	9	972.4	490.7	9	372.8	190.9	32.2	61.1	29	66.8	47.9	47.2	90.2	77.5	50.2	10.8
10l_5_30	6	988.0	497.0	6	417.5	211.7	85.0	57.4	18	95.1	56.6	137.1	88.6	70.6	111.4	16.6
10s_5_50	66	717.0	391.5	66	341.7	203.8	23.1	47.9	71	123.9	97.5	22.4	75.1	56.6	250.8	42.3
10m_5_50	91	619.3	355.1	26	405.8	215.9	51.3	39.2	59	149.6	104.3	80.7	70.6	85.0	119.3	15.6
10l_5_50	51	517.4	284.2	13	199.3	106.1	59.8	62.7	19	85.1	52.1	105.1	81.7	56.2	52.0	22.8
10s_5_70	89	374.7	231.9	83	272.6	177.8	18.1	23.3	100	91.9	95.9	12.9	58.6	32.3	95.9	64.9
10m_5_70	84	1009.5	546.7	59	427.5	243.2	41.2	55.5	75	183.2	129.1	120.3	76.4	76.6	143.1	30.2
10l_5_70	44	1426.0	735.0	44	354.8	199.4	85.6	72.9	49	98.2	73.6	82.7	90.0	33.7	69.0	48.8
20s_1_30	0	358.9	179.4	0	190.5	95.2	14.8	46.9	18	134.7	76.4	42.2	57.4	62.8	610.5	28.4
20m_1_30	0	854.5	427.2	0	243.9	122.0	25.3	71.5	10	200.4	105.2	38.8	75.4	83.0	118.5	17.9
20l_1_30	0	2741.4	1370.7	0	205.4	102.7	51.5	92.5	0	195.2	97.6	269.5	92.9	67.5	130.6	31.7
20s_1_50	4	1652.1	828.1	4	539.1	271.5	21.8	67.2	44	198.5	121.3	54.8	85.4	71.0	708.3	35.2
20m_1_50	0	1556.1	778.1	0	467.9	233.9	32.0	69.9	28	263.9	146.0	74.2	81.2	88.2	180.2	17.2
20l_1_50	0	904.6	452.3	0	185.7	92.9	45.4	79.5	4	134.9	69.4	113.3	84.7	94.2	63.1	4.0
20s_1_70	550	2862.2	1706.1	448	629.7	538.8	24.3	68.4	485	186.1	335.6	37.8	80.3	29.5	338.1	236.6
20m_1_70	258	4543.9	2401.0	242	1188.2	715.1	38.6	70.2	380	402.6	391.3	47.8	83.7	62.4	486.6	147.1
20l_1_70	77	3110.3	1593.6	77	726.1	401.5	63.3	74.8	91	241.1	166.1	113.6	89.6	61.0	212.6	64.7
20s_3_30	100	2680.3	1390.1	54	902.8	478.4	30.1	65.6	133	163.8	148.4	104.3	89.3	79.7	691.0	30.1
20m_3_30	15	2364.1	1189.6	15	826.4	420.7	89.4	64.6	99	273.0	186.0	159.1	84.4	89.9	370.3	18.7
20l_3_30	0	2033.1	1016.6	0	307.8	153.9	98.8	84.9	35	148.0	91.5	168.2	91.0	92.0	227.2	7.3
20s_3_50	225	1467.3	846.1	139	603.1	371.0	36.1	56.2	174	313.4	243.7	89.4	71.2	75.9	799.1	58.8
20m_3_50	130	3485.8	1807.9	97	1162.3	629.6	78.0	65.2	193	257.8	225.4	197.9	87.5	86.4	273.6	30.7
20l_3_50	45	4481.3	2263.1	45	839.2	442.1	161.9	80.5	90	151.2	120.6	300.0	94.7	74.9	313.7	30.3
20s_3_70	543	5436.3	2989.6	438	1000.6	719.3	106.4	75.9	493	210.1	351.5	102.2	88.2	42.3	459.4	202.8

(continued on next page)

Table 7: Computational results of Dispatching rules, Sequential and Integrated heuristics (continued)

Instance	Dispatching rules			Sequential					Integrated						Best MIP bounds	
	E_{LA}	E_{CS}	E_τ	E_{LA}	E_{CS}	E_τ	*cpu*	G%	E_{LA}	E_{CS}	E_τ	*cpu*	G%	*gap*	UB	LB
20m_3_70	197	4097.8	2147.4	170	1308.0	739.0	104.8	65.6	210	361.5	285.7	163.6	86.7	72.5	829.8	78.5
20l_3_70	307	3384.8	1845.9	275	1637.5	956.2	160.0	48.2	335	250.5	292.7	300.0	84.1	65.2	1632.0	101.8
20s_5_30	80	2458.7	1269.4	73	865.1	469.0	48.6	63.0	127	273.4	200.2	99.6	84.2	88.0	931.5	24.0
20m_5_30	37	2977.7	1507.3	37	1041.4	539.2	163.6	64.2	130	260.8	195.4	129.8	87.0	90.0	-	19.5
20l_5_30	30	3035.3	1532.6	30	850.9	440.5	212.7	71.3	80	317.1	198.5	299.6	87.0	-	-	OOM
20s_5_50	210	2552.3	1381.1	194	895.0	544.5	71.2	60.6	213	266.6	239.8	133.0	82.6	79.8	1289.2	48.4
20m_5_50	192	2472.6	1332.3	83	1078.2	580.6	210.9	56.4	298	355.1	326.5	300.0	75.5	-	-	OOM
20l_5_50	146	2917.4	1531.7	68	953.2	510.6	201.2	66.7	81	287.6	184.3	300.0	88.0	-	-	OOM
20s_5_70	390	3077.3	1733.6	321	933.3	627.1	79.1	63.8	465	282.9	373.9	156.4	78.4	73.0	914.8	100.8
20m_5_70	350	4295.8	2322.9	252	1521.8	886.9	156.2	61.8	321	496.2	408.6	283.6	82.4	85.2	842.2	60.6
20l_5_70	130	5138.0	2634.0	123	1035.4	579.2	176.2	78.0	224	334.3	279.1	300.0	89.4	81.5	485.1	51.7
30s_1_30	0	1777.2	888.6	0	447.0	223.5	53.0	74.8	35	283.2	159.1	126.8	82.1	-	1747.9	19.7
30m_1_30	0	2025.8	1012.9	0	630.3	315.2	49.1	68.9	41	407.0	224.0	115.0	77.9	-	2250.9	26.8
30l_1_30	0	6808.6	3404.3	0	348.5	174.3	115.5	94.9	0	337.3	168.6	178.8	95.0	-	2455.5	24.2
30s_1_50	113	4217.1	2165.0	104	1353.9	729.0	63.1	66.3	140	458.9	299.4	114.4	86.2	-	469.8	67.2
30m_1_50	52	4486.5	2269.2	19	2011.0	1015.0	232.2	55.3	137	492.1	314.5	244.1	86.1	-	977.9	21.6
30l_1_50	7	2601.7	1304.3	0	1459.3	729.6	273.2	44.1	9	280.8	144.9	268.9	88.9	-	2737.8	13.9
30s_1_70	1121	6534.2	3827.6	934	1450.6	1192.3	59.5	68.8	1034	345.9	689.9	109.7	82.0	-	1675.5	43.8
30m_1_70	616	9826.3	5221.1	548	2912.2	1730.1	153.2	66.9	777	628.3	702.7	246.5	86.5	58.1	1338.6	294.2
30l_1_70	287	7409.5	3848.2	249	1572.4	910.7	119.3	76.3	260	390.4	325.2	300.0	91.5	57.5	945.6	138.1
30s_3_30	173	5440.2	2806.6	132	1642.9	887.4	115.0	68.4	217	345.6	281.3	300.0	90.0	-	2221.7	37.0
30m_3_30	59	5332.9	2695.9	59	2237.0	1148.0	161.4	57.4	275	464.0	369.5	300.0	86.3	-	-	OOM
30l_3_30	35	3494.1	1764.5	12	1387.4	699.7	300.0	60.3	66	257.7	161.9	300.0	90.8	-	-	OOM
30s_3_50	414	4759.2	2586.6	320	1649.5	984.7	96.5	61.9	419	506.4	462.7	300.0	82.1	-	1986.6	OOM
30m_3_50	234	8504.2	4369.1	199	2816.5	1507.7	186.2	65.5	470	412.1	441.1	300.0	89.9	-	-	OOM
30l_3_50	115	9105.0	4610.0	112	2358.9	1235.5	186.0	73.2	206	509.7	357.9	300.0	92.2	-	-	OOM
30s_3_70	1080	11147.2	6113.6	918	2047.8	1482.9	172.0	75.7	913	439.3	676.2	271.5	88.9	55.5	1939.8	301.1
30m_3_70	408	10669.2	5538.6	364	2798.3	1581.1	163.7	71.5	602	721.7	661.9	300.0	88.0	80.2	1882.7	130.8
30l_3_70	630	11675.3	6152.7	529	3183.8	1856.4	293.6	69.8	939	543.4	741.2	300.0	88.0	-	5019.0	97.4
30s_5_30	193	5631.7	2912.3	160	1857.8	1008.9	169.6	65.4	249	561.0	405.0	176.3	86.1	-	-	OOM
30m_5_30	92	8734.2	4413.1	92	2129.0	1110.5	249.6	74.8	248	600.1	424.1	300.0	90.4	-	-	OOM
30l_5_30	61	6883.5	3472.3	61	1496.6	778.8	300.0	77.6	282	642.0	462.0	300.0	86.7	-	-	OOM
30s_5_50	391	7790.2	4090.6	349	2462.8	1405.9	125.9	65.6	531	408.8	469.9	300.0	88.5	-	-	61.0
30m_5_50	275	6811.0	3543.0	166	2745.7	1455.9	252.7	58.9	665	669.1	667.0	300.0	81.2	-	-	OOM
30l_5_50	246	10671.6	5458.8	162	2531.2	1346.6	300.0	75.3	566	601.5	583.8	300.0	89.3	-	-	OOM
30s_5_70	925	8077.2	4501.1	626	2258.4	1442.2	181.2	68.0	796	758.7	777.4	300.0	82.7	79.6	2624.5	158.7
30m_5_70	679	8864.2	4771.6	537	3187.9	1862.5	212.3	61.0	1152	914.8	1033.4	300.0	78.3	-	-	OOM
30l_5_70	262	10267.7	5264.8	252	2852.5	1552.3	223.7	70.5	722	715.0	718.5	300.0	86.4	-	-	OOM
50s_1_30	0	8471.2	4235.6	0	2143.5	1071.8	203.5	74.7	119	1139.1	629.0	226.7	85.1	89.4	8343.5	66.5
50m_1_30	0	9120.2	4560.1	0	2177.5	1088.8	199.3	76.1	139	1470.3	804.6	281.0	82.4	93.0	8822.7	56.7
50l_1_30	0	23604.5	11802.2	0	1661.7	830.8	225.1	93.0	4	1536.6	770.3	219.7	93.5	-	-	OOM
50s_1_50	582	16480.0	8531.0	381	4641.6	2511.3	211.6	70.6	612	1190.7	901.4	257.3	89.4	77.9	6345.8	198.9

(continued on next page)

Table 7: Computational results of Dispatching rules, Sequential and Integrated heuristics (continued)

Instance	Dispatching rules			Sequential					Integrated						Best MIP bounds	
	E_{LA}	E_{CS}	E_{τ}	E_{LA}	E_{CS}	E_{τ}	*cpu*	G%	E_{LA}	E_{CS}	E_{τ}	*cpu*	G%	*gap*	UB	LB
50m_1_50	161	15745.8	7953.4	113	6542.7	3327.9	202.5	58.2	632	1854.1	1243.0	300.0	84.4	93.3	7059.6	83.0
50l_1_50	39	14297.6	7168.3	12	6846.5	3429.3	300.0	52.2	151	1253.0	702.0	300.0	90.2	92.7	5006.6	51.1
50s_1_70	2128	16748.6	9438.3	1739	4143.1	2941.0	300.0	68.8	1993	1059.3	1526.2	239.6	83.8	47.8	3188.3	796.8
50m_1_70	1092	28900.6	14996.3	1001	7380.2	4190.6	300.0	72.1	1747	1608.0	1677.5	286.0	88.8	69.4	10327.3	513.7
50l_1_70	677	25874.3	13275.7	587	7885.2	4236.1	300.0	68.1	1044	1152.0	1098.0	300.0	91.7	71.7	10793.5	311.1
50s_3_30	402	15748.2	8075.1	294	5025.6	2659.8	300.0	67.1	608	1131.4	869.7	300.0	89.2	-	-	OOM
50m_3_30	196	20272.9	10234.5	148	7168.1	3658.0	300.0	64.3	861	1522.7	1191.8	300.0	88.4	-	-	OOM
50l_3_30	131	18198.4	9164.7	54	10340.5	5197.3	300.0	43.3	962	1361.9	1161.9	300.0	87.3	-	-	OOM
50s_3_50	858	18143.2	9500.6	667	5016.4	2841.7	218.8	70.1	1237	1272.3	1254.7	300.0	86.8	89.4	-	133.4
50m_3_50	531	26273.5	13402.3	446	9864.5	5155.3	300.0	61.5	1804	1166.2	1485.1	300.0	88.9	-	-	OOM
50l_3_50	286	30411.5	15348.8	264	10482.6	5373.3	300.0	65.0	1785	1604.1	1694.6	300.0	89.0	-	-	OOM
50s_3_70	2236	22187.2	12211.6	1920	6035.7	3977.8	300.0	67.4	2314	1067.4	1690.7	300.0	86.2	66.6	8068.8	564.4
50m_3_70	1072	25060.5	13066.3	870	9320.0	5095.0	300.0	61.0	2608	1606.4	2107.2	300.0	83.9	-	-	OOM
50l_3_70	1289	41972.4	21630.7	1106	13399.3	7252.7	300.0	66.5	3839	2053.0	2946.0	300.0	86.4	-	-	OOM
50s_5_30	617	17668.2	9142.6	515	5204.4	2859.7	300.0	68.7	1190	1405.5	1297.7	300.0	85.8	-	-	OOM
50m_5_30	346	36127.5	18236.8	346	9294.6	4820.3	300.0	73.6	1881	1872.6	1876.8	300.0	89.7	-	-	OOM
50l_5_30	162	25689.1	12925.5	124	10134.2	5129.1	300.0	60.3	1904	1982.9	1943.4	300.0	85.0	-	-	OOM
50s_5_50	1319	19176.2	10247.6	1291	8511.1	4901.0	300.0	52.2	3012	1679.5	2345.7	300.0	77.1	-	-	OOM
50m_5_50	543	24680.5	12611.8	417	9208.6	4812.8	300.0	61.8	2722	2023.6	2372.8	300.0	81.2	-	-	OOM
50l_5_50	469	37730.4	19099.7	349	13127.9	6738.5	300.0	64.7	2327	2723.1	2525.0	300.0	86.8	-	-	OOM
50s_5_70	2129	21907.9	12018.5	1689	7782.4	4735.7	300.0	60.6	3485	1736.3	2610.6	300.0	78.3	-	-	OOM
50m_5_70	1803	29442.5	15622.8	1584	13188.9	7386.5	300.0	52.7	4126	4085.9	4105.9	300.0	73.7	-	-	OOM
50l_5_70	683	35730.4	18206.7	639	16705.5	8672.2	300.0	52.4	4363	3092.1	3727.5	300.0	79.5	-	-	OOM
70s_1_30	159	21129.6	10644.3	59	9551.3	4805.1	300.0	54.9	855	2178.6	1516.8	300.0	85.8	94.7	9834.0	80.5
70m_1_30	0	20434.8	10217.4	0	4781.7	2390.8	300.0	76.6	493	2921.1	1707.0	300.0	83.3	-	-	OOM
70l_1_30	0	40968.3	20484.2	0	4610.7	2305.4	300.0	88.7	299	2930.8	1614.9	300.0	92.1	-	-	OOM
70s_1_50	1618	36220.8	18919.4	1122	10137.3	5629.7	300.0	70.2	2174	2275.6	2224.8	300.0	88.2	79.9	10748.1	447.7
70m_1_50	527	35398.5	17962.7	421	14808.9	7614.9	300.0	57.6	2503	3011.1	2757.0	300.0	84.7	93.3	-	185.9
70l_1_50	277	40254.4	20265.7	119	19999.4	10059.2	300.0	50.4	1565	2625.1	2095.1	300.0	89.7	-	-	OOM
70s_1_70	3854	34408.4	19131.2	3126	9247.3	6186.6	300.0	67.7	3727	2134.4	2930.7	300.0	84.7	53.4	10011.1	1366.7
70m_1_70	2222	56536.3	29379.1	1917	17887.0	9902.0	300.0	66.3	3786	3344.4	3565.2	300.0	87.9	74.8	15617.5	898.4
70l_1_70	1339	58128.2	29733.6	1216	19929.1	10572.6	300.0	64.4	3041	2238.2	2639.6	300.0	91.1	78.2	21421.9	575.7
70s_3_30	925	30533.0	15729.0	692	12187.0	6439.5	300.0	59.1	2229	1611.1	1920.1	300.0	87.8	-	-	OOM
70m_3_30	485	46373.1	23429.1	485	15703.6	8094.3	300.0	65.5	2383	2675.3	2529.1	300.0	89.2	-	-	OOM
70l_3_30	263	38437.7	19350.3	175	26446.0	13310.5	300.0	31.2	2363	4207.5	3285.2	300.0	83.0	-	-	OOM
70s_3_50	1653	40538.0	21095.5	1220	12007.4	6613.7	300.0	68.6	2837	1971.9	2404.4	300.0	88.6	-	-	OOM
70m_3_50	4039	52832.0	28435.5	973	16526.0	8749.5	300.0	69.2	4608	2439.0	3523.5	300.0	87.6	-	-	OOM
70l_3_50	656	64629.5	32642.8	561	26877.7	13719.4	300.0	58.0	4183	3868.6	4025.8	300.0	87.7	-	-	OOM
70s_3_70	994	47335.1	24164.6	3387	13763.5	8575.2	300.0	64.5	4277	7306.6	5791.8	300.0	76.0	-	-	OOM
70m_3_70	2098	53474.1	27786.1	1780	21424.2	11602.1	300.0	58.2	4947	3678.0	4312.5	300.0	84.5	-	-	OOM
70l_3_70	2080	69566.2	35823.1	1898	32793.6	17345.8	300.0	51.6	5844	6006.1	5925.0	300.0	83.5	-	-	OOM
70s_5_30	1189	34016.0	17602.5	1173	11377.2	6275.1	300.0	64.4	3019	2394.4	2706.7	300.0	84.6	-	-	OOM

(continued on next page)

Table 7: Computational results of Dispatching rules, Sequential and Integrated heuristics (continued)

Instance	Dispatching rules			Sequential					Integrated						Best MIP bounds	
	E_{LA}	E_{CS}	E_{τ}	E_{LA}	E_{CS}	E_{τ}	*cpu*	G%	E_{LA}	E_{CS}	E_{τ}	*cpu*	G%	*gap*	UB	LB
70m_5_30	624	73034.1	36829.1	623	18676.8	9649.9	300.0	73.8	4391	3693.4	4042.2	300.0	89.0	-	-	OOM
70l_5_30	414	65472.1	32943.0	414	23484.7	11949.3	300.0	63.7	3336	5458.1	4397.0	300.0	86.7	-	-	OOM
70s_5_50	2189	41795.0	21992.0	2255	13117.9	7686.4	300.0	65.0	4696	3542.3	4119.2	300.0	81.3	-	-	OOM
70m_5_50	850	53458.1	27154.1	850	17077.1	8963.5	300.0	67.0	5336	4445.2	4890.6	300.0	82.0	-	-	OOM
70l_5_50	739	58429.7	29584.4	739	19174.3	9956.7	300.0	66.3	5270	5840.9	5555.4	300.0	81.2	-	-	OOM
70s_5_70	3404	40241.7	21822.9	2946	16937.8	9941.9	300.0	54.4	5891	2974.5	4432.7	300.0	79.7	-	-	OOM
70m_5_70	2901	62528.1	32714.6	2655	25866.7	14260.8	300.0	56.4	7327	8793.2	8060.1	300.0	75.4	-	-	OOM
70l_5_70	1332	72718.7	37025.4	1190	31762.4	16476.2	300.0	55.5	6893	7988.2	7440.6	300.0	79.9	-	-	OOM

Figure 9 illustrates how the integrated heuristic may improve over the constructive heuristic. The instances are grouped by their number of requests and storage sizes. The horizontal axis represents the instance categories while the vertical axis corresponds to the improvement over constructive heuristics. The results are averaged over each category and demonstrate that the improvement over the constructive heuristic is, on average, 84% over all instances. Given that the constructive heuristic reflects common operational dispatching rules, the potential performance increase is large.

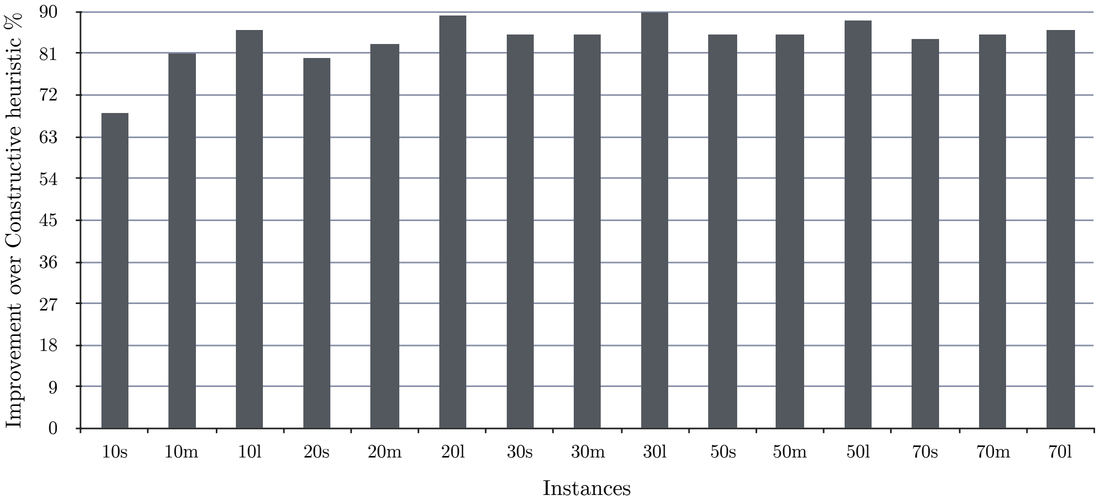


Figure 9: Comparison of the constructive and integrated heuristic's performance

5.4 Weight parameters and instance analysis

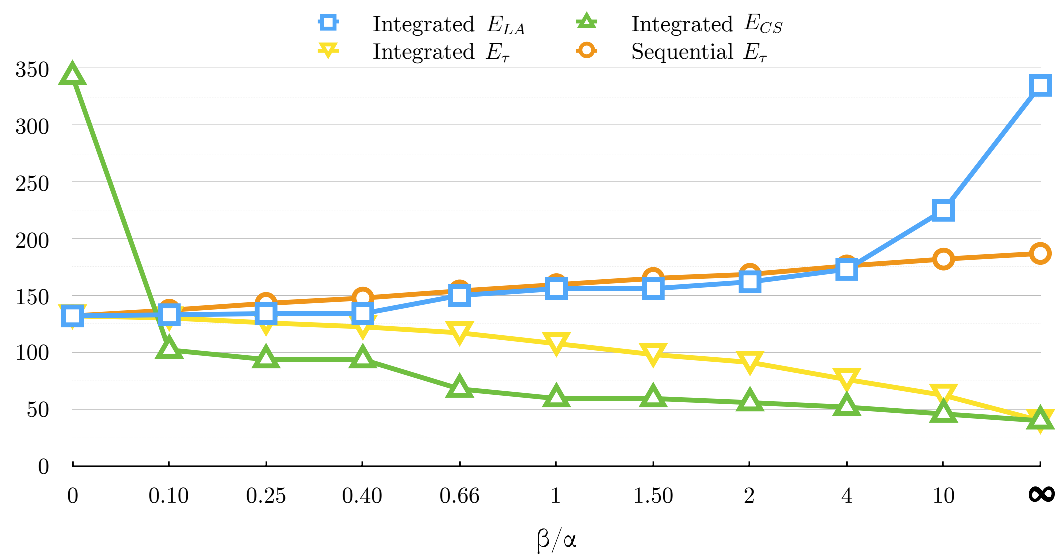
Section 5.2 already briefly touched on the impact of the weight parameters α and β. In this section, the effect of adjusting these weight parameters is further investigated. Both a small (`10s_1_70`) and large (`50m_1_50`) instance were selected to illustrate the weight parameters' effect. Figure 10 presents the impact of adjusting α and β for both the small (Figure 10(a), solved by $\mathcal{F}_{LA}+\mathcal{F}_{CS}$ and $\mathcal{F}_{CWS}$), and the large instance (Figure 10(b)), solved by the sequential and integrated heuristic. The figure illustrates the LAP (E_{LA}), CSP (E_{CS}) and weighted objective function (E_τ) values obtained by the integrated approach as well as the E_τ obtained by the sequential approach.

The weight parameters in the sequential approach do not impact upon the optimization process as the two sub-problems are solved separately. Sequential E_{LA} and E_{CS} are constant for all values of α and β. Therefore $E_\tau(\alpha,\beta)$ is a linear function.

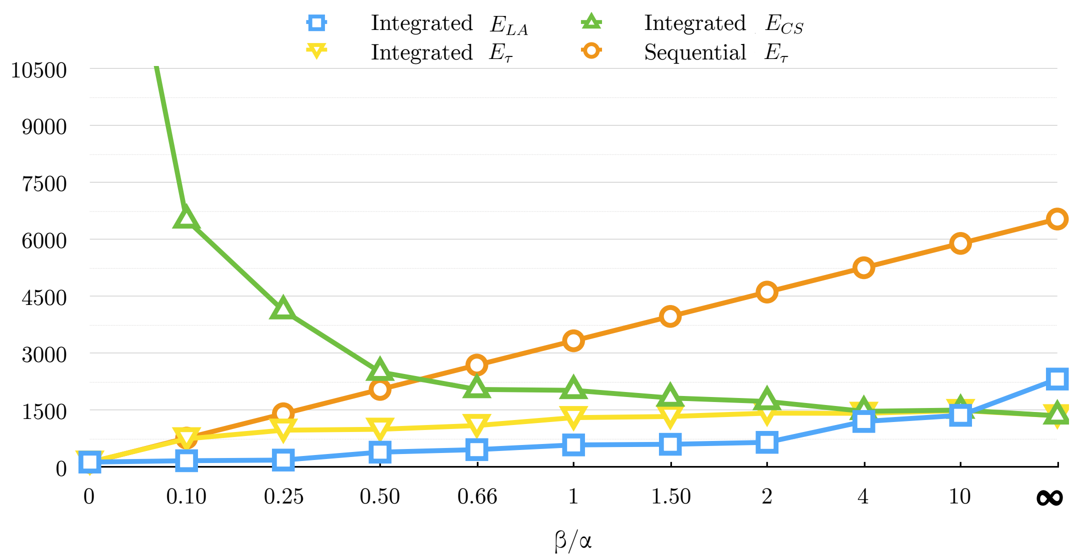
As illustrated for $\dfrac{\beta}{\alpha}=0$:

$$\text{integrated } E_\tau = \text{sequential } E_\tau = \text{integrated } E_{LA} = \text{sequential } E_{LA}$$

By increasing $\dfrac{\beta}{\alpha}$, the integrated E_{LA} increases while the integrated E_{CS} decreases. Interestingly the integrated E_τ is always lower than the sequential E_τ and by increasing $\dfrac{\beta}{\alpha}$ the gap between the integrated and sequential E_τ increases.



(a) Instance 10s_1_70 solved by the mathematical formulation



(b) Instance 50m_1_50 solved by the heuristic

Figure 10: Impact of the weight parameters' ratio.

Figure 11 demonstrates how different attributes of instances may impact the improvement of the sequential and integrated heuristic over the constructive heuristic. Figure 11(a) represents the average E_τ obtained by the constructive, sequential and integrated heuristic with respect to the stacking level. By increasing the maximum stacking level, the relative improvement over the solution obtained by the constructive heuristic decreases. By increasing the storage size from small to large (Figure 11(b)), the sequential and integrated heuristics improve the initial solution from 61% to 68% and from 80% to 87% respectively. Figure 11(d), meanwhile, illustrates how the load factor does not significantly influence the algorithms' improvement over the initial solution.

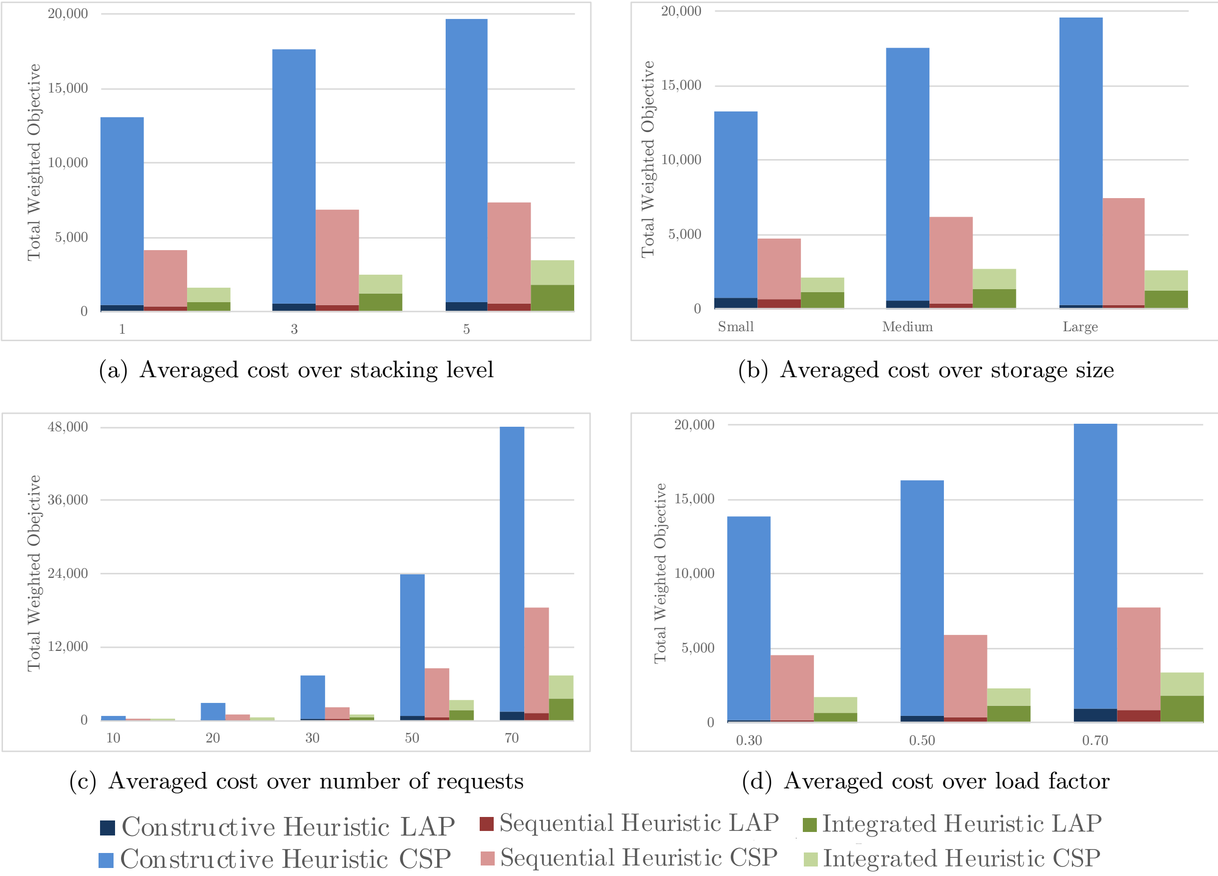


Figure 11: Instance attributes analysis by the constructive, sequential and integrated heuristic when $\alpha = \beta$.

6. Conclusion

This paper investigated the impact of integrating a location assignment problem (LAP) and a crane scheduling problem (CSP) in crane-operated warehouses by introducing the integrated Crane-operated Warehouse Scheduling Problem (CWSP). The CWSP assigns a storage location to input requests, assigns a crane to execute each request and decides how the set of requests must be sequenced per crane in such a way that the total storage cost and tardiness is minimized. Mixed Integer Programming (MIP) formulations for the LAP and CSP were presented in addition to a continuous-time MIP formulation which integrates both the LAP and CSP. This model considers realistic crane interactions in the storage area where cranes cannot pass each other and must keep a safety distance. Furthermore, a meta-heuristic based on Late Acceptance Hill Climbing (LAHC) was developed to overcome the limited scaling ability of solving the mathematical formulations by MIP solvers. In addition, 135 instances with various problem specifications were generated to enable validation and encourage future research.

A comprehensive computational study revealed how integrating the LAP and the CSP may lead to 48% improvement for the CSP while keeping the same level of quality for the LAP solutions. Subsequently, the results showed that by integrating the LAP and CSP into one problem, there is a significant reduction in the total weighted objective of 62%. Furthermore, the benefit of the integrated heuristic over the MIP formulation was shown, where better solutions for both medium and large size instances were obtained compared to solving the MIP formulation on the respective instances. Additionally, a simulation of a real-world automated warehouse shows a significant potential for minimizing the storage cost and tardiness of the requests, when comparing the new procedures with typical dispatching rules.

In conclusion, integrating location assignment and crane scheduling coordinates the resources in automated warehouses or container terminals more effectively and eventually leads to efficiently storing the products or containers in the storage areas as well as minimizing the tardiness of the input and output requests.

Several interesting avenues exist to build upon this study in future research. Further solution approaches may be considered and investigated. Particularly, exact solution approaches would be a valuable contribution which may include proposing efficient lower bounding methods. Due to the nature of operations in automated warehouses, other research directions may focus on exploring the development of robust scheduling models which consider uncertain requests' arrival times.

Appendix A. Detailed computational results

Table A.8 presents detailed computational results obtained by the mathematical formulations. $gap_{la}\%$ (location assignment gap) and $gap_{cs}\%$ (crane scheduling gap) indicate the gap obtained by $\mathcal{F}_{LA}$ and $\mathcal{F}_{CS}$, respectively, when solving the problem sequentially. $gap\%$ denotes the gap obtained by $\mathcal{F}_{AWS}$.

Table A.8: Detailed computational results obtained by mathematical formulations

Ins.	$\mathcal{F}_{LA}+\mathcal{F}_{CS}$							$\mathcal{F}_{CWS}$ ($\alpha >> \beta$)					$\mathcal{F}_{CWS}$ ($\alpha = \beta$)				
	E_{LA}	E_{CS}	$E_\tau(\alpha >> \beta)$	$E_\tau(\alpha = \beta)$	$gap_{la}\%$	$gap_{cs}\%$	cpu	E_{LA}	E_{CS}	E_τ	$gap\%$	cpu	E_{LA}	E_{CS}	E_τ	$gap\%$	cpu
10s_1_30	0	267.03	2.670E+02	133.51	0.00	0.00	107.28	0	55.13	5.513E+01	0.00	49.79	1	51.53	26.26	0.00	170.92
10m_1_30	0	478.41	4.784E+02	239.20	0.00	0.00	604.05	0	37.62	3.762E+01	0.00	60.78	0	37.62	18.81	0.00	513.46
10l_1_30	0	515.69	5.157E+02	257.84	0.00	0.00	302.12	0	76.82	7.682E+01	0.00	195.32	0	76.82	38.41	0.00	2044.59
10s_1_50	0	229.25	2.293E+02	114.63	0.00	0.00	67.69	0	123.93	1.239E+02	0.00	311.87	11	78.92	44.96	0.00	1009.35
10m_1_50	0	219.18	2.192E+02	109.59	0.00	0.00	360.00	0	56.45	5.645E+01	0.00	91.66	7	40.07	23.53	0.00	475.39
10l_1_50	0	447.85	4.479E+02	223.92	0.00	0.00	164.24	0	44.49	4.449E+01	0.00	97.42	4	32.49	18.24	0.00	187.29
10s_1_70	132	186.97	1.320E+07	159.49	0.00	0.00	115.83	132	135.97	1.320E+07	0.00	11.06	156	59.25	107.62	0.00	1227.04
10m_1_70	48	348.86	4.800E+06	198.43	0.00	0.00	121.24	48	449.55	4.800E+06	0.00	11.09	86	185.92	135.96	0.00	3290.77
10l_1_70	11	271.44	1.100E+06	141.22	0.00	0.00	155.79	11	148.17	1.100E+06	0.00	27.31	25	73.78	49.39	0.00	613.51
10s_3_30	13	352.10	1.300E+06	182.55	0.00	0.00	442.41	13	179.72	1.300E+06	0.00	119.08	49	76.07	62.53	35.51	3600.00
10m_3_30	1	354.66	1.004E+05	177.83	0.00	0.00	1254.72	1	107.51	1.001E+05	0.00	3600.00	9	70.37	39.68	80.35	3600.00
10l_3_30	0	295.28	2.953E+02	147.64	0.00	0.00	746.32	0	79.51	7.951E+01	0.00	2387.71	-	-	-	-	3600.00
10s_3_50	53	317.77	5.300E+06	185.39	0.00	0.00	879.75	53	201.77	5.300E+06	0.00	3535.55	64	109.83	86.91	49.46	3600.00
10m_3_50	31	409.77	3.100E+06	220.39	0.00	0.00	911.50	31	332.35	3.100E+06	0.00	303.92	41	126.77	83.88	80.67	3600.00
10l_3_50	9	454.43	9.005E+05	231.72	0.00	0.00	1056.12	9	145.71	9.001E+05	0.00	467.58	12	85.91	48.95	0.00	2366.16
10s_3_70	130	392.07	1.300E+07	187.73	0.00	0.00	187.72	130	279.13	1.300E+07	0.00	217.53	190	102.40	146.20	38.42	3600.00
10m_3_70	64	433.66	6.400E+06	248.83	0.00	0.00	212.35	64	492.57	6.400E+06	0.00	92.46	93	77.90	85.45	20.80	3600.00
10l_3_70	98	441.02	9.800E+06	269.51	0.00	0.00	574.97	98	566.37	9.800E+06	0.00	502.12	134	118.11	126.05	55.16	3600.00
10s_5_30	18	322.97	1.800E+06	170.48	0.00	0.00	2714.13	18	182.53	1.800E+06	0.00	3600.00	35	93.23	64.11	66.30	3600.00
10m_5_30	653	477.18	6.530E+07	565.09	100.00	0.00	3224.10	9	275.46	9.002E+05	0.01	3600.00	24	76.36	50.18	78.51	3600.00
10l_5_30	464	457.64	4.640E+07	460.82	100.00	0.00	2845.26	6	244.62	6.002E+05	0.00	3600.00	16	206.88	111.44	85.09	3600.00
10s_5_50	66	435.57	6.600E+06	250.79	0.00	0.00	2300.19	66	292.55	6.600E+06	6.82	3600.00	-	-	-	-	3600.00
10m_5_50	25	-	-	-	0.00	-	3600.00	25	294.46	2.500E+06	42.16	3600.00	42	196.59	119.30	86.90	3600.00
10l_5_50	13	602.07	1.301E+06	307.54	0.00	0.00	2306.55	13	159.11	1.300E+06	0.00	3391.92	21	82.93	51.96	56.24	3600.00
10s_5_70	83	247.93	8.300E+06	165.64	0.00	0.00	197.83	83	247.93	8.300E+06	0.00	176.25	100	91.85	95.92	32.34	3600.00
10m_5_70	59	419.26	5.900E+06	239.13	0.00	0.00	1313.89	59	263.06	5.900E+06	0.00	3600.00	75	211.27	143.13	78.93	3600.00
10l_5_70	43	541.38	4.301E+06	292.19	0.00	0.00	1755.06	43	418.37	4.300E+06	0.00	1223.19	51	87.02	69.01	29.27	3600.00
20s_1_30	0	1221.04	1.221E+03	610.52	0.00	0.00	1970.28	0	160.38	1.603E+02	35.40	3600.00	-	-	-	-	3600.00
20m_1_30	0	1821.80	1.822E+03	910.90	0.00	0.00	2098.25	0	225.29	2.252E+02	37.60	3600.00	1	235.90	118.45	84.91	3600.00
20l_1_30	0	2171.42	2.171E+03	1085.71	0.00	0.00	2425.32	0	198.40	1.984E+02	0.00	3420.09	0	261.18	130.59	75.71	3600.00
20s_1_50	4	1412.59	4.014E+05	708.30	0.00	0.00	1898.00	4	425.09	4.004E+02	0.09	3600.00	-	-	-	-	3600.00
20m_1_50	0	1373.24	1.373E+03	686.21	0.00	0.00	1967.59	0	445.37	4.453E+02	85.75	3600.00	9	351.42	180.21	90.47	3600.00
20l_1_50	0	1851.12	1.851E+03	925.56	0.00	0.00	2205.82	0	155.85	1.558E+02	0.00	3600.00	8	118.23	63.11	93.63	3600.00
20s_1_70	448	801.34	4.480E+07	624.67	0.00	0.00	2062.93	448	692.26	4.480E+07	0.00	2151.22	480	196.16	338.08	30.03	3600.00
20m_1_70	241	2396.80	2.410E+07	1318.90	0.00	0.00	1948.97	241	1287.71	2.410E+07	0.00	1295.57	333	640.11	486.55	69.77	3600.00
20l_1_70	77	1210.96	7.701E+06	643.98	0.00	0.00	2020.08	77	758.37	7.700E+06	0.00	358.51	88	337.18	212.59	69.57	3600.00
20s_3_30	54	1328.04	5.401E+06	691.02	77.77	80.41	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
20m_3_30	995	2031.35	9.950E+07	1513.18	100.00	60.03	3600.00	15	740.32	1.500E+06	0.04	3600.00	64	676.63	370.31	94.94	3600.00
20l_3_30	648	-	-	-	100.00	-	3600.00	0	277.50	2.775E+02	47.91	3600.00	37	417.42	227.21	96.77	3600.00
20s_3_50	168	1434.26	1.680E+07	799.13	58.53	73.01	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
20m_3_50	99	2203.80	9.902E+06	1151.40	82.82	87.74	3600.00	-	-	-	-	3600.00	181	366.17	273.58	88.77	3600.00
20l_3_50	1344	1852.22	1.344E+08	1598.11	99.62	49.66	3600.00	45	1123.48	4.501E+06	66.67	3600.00	49	578.48	313.74	90.35	3600.00

(continued on next page)

Table A.8: Detailed computational results obtained by mathematical formulations (continued)

Ins.	$\mathcal{F}_{LA} + \mathcal{F}_{CS}$							$\mathcal{F}_{CWS}$ ($\alpha >> \beta$)					$\mathcal{F}_{CWS}$ ($\alpha = \beta$)				
	E_{LA}	E_{CS}	$E_\tau(\alpha >> \beta)$	$E_\tau(\alpha = \beta)$	$gap_{la}\%$	$gap_{cs}\%$	cpu	E_{LA}	E_{CS}	E_τ	$gap\%$	cpu	E_{LA}	E_{CS}	E_τ	$gap\%$	cpu
20s_3_70	423	1552.35	4.230E+07	987.67	14.42	59.01	3600.00	424	2902.26	4.240E+07	14.62	3600.00	528	390.82	459.41	55.85	3600.00
20m_3_70	168	1491.69	1.680E+07	829.84	0.00	76.80	3600.00	168	1407.24	1.680E+07	11.31	3600.00	-	-	-	-	3600.00
20l_3_70	274	2989.96	2.740E+07	1631.98	44.16	81.51	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
20s_5_30	923	939.96	9.230E+07	931.48	99.89	42.39	3600.00	68	1630.26	6.801E+06	95.45	3600.00	-	-	-	-	3600.00
20m_5_30	886	-	-	-	100.00	-	3600.00	43	2017.09	4.302E+06	98.66	3600.00	-	-	-	-	3600.00
20s_5_50	1527	1051.42	1.527E+08	1289.21	99.21	34.04	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
20m_5_50	1450	-	-	-	99.72	-	3600.00	87	2062.80	8.702E+06	88.24	3600.00	-	-	-	-	3600.00
20s_5_70	330	1499.62	3.300E+07	914.81	59.09	70.46	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
20m_5_70	249	1435.31	2.490E+07	842.15	83.93	76.39	3600.00	385	1738.69	3.850E+07	78.95	3600.00	-	-	-	-	3600.00
20l_5_70	129	-	-	-	82.17	-	3600.00	-	-	-	-	3600.00	158	812.23	485.11	89.34	3600.00
30s_1_30	0	3495.86	3.496E+03	1747.93	0.00	0.00	2214.11	0	668.80	6.688E+02	91.89	3600.00	-	-	-	-	3600.00
30m_1_30	0	4501.77	4.502E+03	2250.88	0.00	0.00	2554.77	0	921.33	9.213E+02	90.25	3600.00	-	-	-	-	3600.00
30l_1_30	0	4911.02	4.911E+03	2455.51	0.00	94.73	3600.00	0	634.49	6.344E+02	80.19	3600.00	-	-	-	-	3600.00
30s_1_50	98	3299.06	9.803E+06	1698.53	0.00	0.00	3401.45	98	1303.66	9.801E+06	11.74	3600.00	210	729.63	469.81	85.70	3600.00
30m_1_50	19	3742.17	1.904E+06	1880.58	100.00	95.28	3600.00	19	2616.36	1.902E+06	99.99	3600.00	75	1880.72	977.86	97.79	3600.00
30l_1_50	0	4683.93	4.684E+03	2737.80	0.00	0.00	2737.80	0	724.51	7.245E+02	94.12	3600.00	-	-	-	-	3600.00
30s_1_70	934	2416.98	9.340E+07	1675.49	10.38	66.61	3600.00	937	2667.16	9.370E+07	11.20	3600.00	-	-	-	-	3600.00
30m_1_70	544	5919.16	5.441E+07	3231.58	8.63	87.56	3600.00	554	3078.50	5.540E+07	10.29	3600.00	631	2046.16	1338.58	78.02	3600.00
30l_1_70	249	3310.89	2.490E+07	1779.94	0.00	0.00	2641.01	-	-	-	-	3600.00	268	1623.10	945.55	85.39	3600.00
30s_3_30	1585	2858.33	1.585E+08	2221.66	99.74	59.80	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
30s_3_50	330	3643.15	3.300E+07	1986.57	70.00	84.45	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
30s_3_70	821	3058.59	8.210E+07	1939.79	18.27	69.32	3600.00	840	3613.93	8.400E+07	19.35	3600.00	-	-	-	-	3600.00
30m_3_70	371	3394.37	3.710E+07	1882.68	39.35	84.73	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
30l_3_70	2589	-	-	-	97.29	-	3600.00	-	-	-	-	3600.00	684	9339.08	5019.04	98.06	3600.00
30s_5_70	756	4492.94	7.560E+07	2624.47	85.58	80.18	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
50s_1_30	0	16686.92	1.669E+04	8343.46	0.00	0.00	2892.64	-	-	-	-	3600.00	-	-	-	-	3600.00
50m_1_30	0	17645.35	1.765E+04	8822.67	0.00	97.98	3600.00	0	8264.20	8.264E+03	98.38	3600.00	-	-	-	-	3600.00
50s_1_50	401	12290.55	4.011E+07	6345.77	34.41	93.74	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
50m_1_50	250	13869.28	2.501E+07	7059.64	84.40	95.04	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
50l_1_50	11	15907.70	1.116E+06	7959.35	90.90	97.80	3600.00	-	-	-	-	3600.00	146	9867.24	5006.62	98.98	3600.00
50s_1_70	1743	11544.01	1.743E+08	6643.50	16.40	83.19	3600.00	1772	6898.15	1.772E+08	17.77	3600.00	2370	4006.59	3188.29	75.01	3600.00
50m_1_70	1017	19637.64	1.017E+08	10327.32	14.06	92.16	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
50l_1_70	599	20988.08	5.992E+07	10793.54	12.18	95.38	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
50s_3_70	3904	12233.50	3.904E+08	8068.75	78.32	72.48	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
70s_1_30	67	19600.96	6.720E+06	9833.98	100.00	96.20	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
70s_1_50	1436	20060.26	1.436E+08	10748.13	48.11	90.22	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
70s_1_70	3225	16797.21	3.225E+08	10011.10	18.63	80.97	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
70m_1_70	2023	29211.91	2.023E+08	15617.45	19.37	90.92	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00
70l_1_70	1332	41511.74	1.332E+08	21421.87	20.79	95.56	3600.00	-	-	-	-	3600.00	-	-	-	-	3600.00

References

Boysen, N., Emde, S., 2016. The parallel stack loading problem to minimize blockages. European Journal of Operational Research 249, 618–627.

Boysen, N., Stephan, K., 2016. A survey on single crane scheduling in automated storage/retrieval systems. European Journal of Operational Research 254, 691–704.

Burke, E.K., Bykov, Y., 2017. The late acceptance hill-climbing heuristic. European Journal of Operational Research 258, 70–78.

Chen, L., Lu, Z., 2012. The storage location assignment problem for outbound containers in a maritime terminal. International Journal of Production Economics 135, 73–80.

Darvish, M., Coelho, L.C., 2018. Sequential versus integrated optimization: Production, location, inventory control, and distribution. European Journal of Operational Research 268, 203 – 214.

Dorndorf, U., Schneider, F., 2010. Scheduling automated triple cross-over stacking cranes in a container yard. OR Spectrum 32, 617–632.

Escudero, L., 1988. An inexact algorithm for the sequential ordering problem. European Journal of Operational Research 37, 236–249.

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., Vernooij, F.G., Zaerpour, N., 2017. A simulation study of the performance of twin automated stacking cranes at a seaport container terminal. European Journal of Operational Research 261, 108–128.

Gharehgozli, A.H., Yu, Y., de Koster, R., Udding, J.T., 2014. A decision-tree stacking heuristic minimising the expected number of reshuffles at a container terminal. International Journal of Production Research 52, 2592–2611.

Jovanovic, R., Voß, S., 2014. A chain heuristic for the blocks relocation problem. Computers & Industrial Engineering 75, 79–86.

Ku, D., Arthanari, T.S., 2016. On the abstraction method for the container relocation problem. Computers & Operations Research 68, 110–122.

Li, W., Goh, M., Wu, Y., Petering, M., 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., Goh, M., Souza, R.D., 2009. Discrete time model and algorithms for container yard crane scheduling. European Journal of Operational Research 198, 165–172.

López-Ibáñez, M., Dubois-Lacoste, J., Cáceres, L.P., Birattari, M., Stützle, T., 2016. The irace package: Iterated racing for automatic algorithm configuration. Operations Research Perspectives 3, 43–58.

Montemanni, R., Smith, D., Rizzoli, A., Gambardella, L., 2009. Sequential ordering problems for crane scheduling in port terminals. International Journal of Simulation and Process Modelling 5, 348–361. doi:10.1504/IJSPM.2009.032597.

Park, T., Choe, R., Hun Kim, Y., Ryel Ryu, K., 2011. Dynamic adjustment of container stacking policy in an automated container terminal. International Journal of Production Economics 133, 385–392.

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.

Zhang, C., Wu, T., Zhong, M., Zheng, L., Miao, L., 2014. Location assignment for outbound containers with adjusted weight proportion. Computers & Operations Research 52, 84–93.