JOURNAL OF MARITIME RESEARCH

Vol XXI. No. I (2024) pp 23–29

**ISSN: 1697-4840,** **www.jmr.unican.es**

# Dynamic container relocation problem

Khaled Mili[1,*]

*ARTICLE INFO*

***Article history:***
Received 14 Jun 2023;
in revised from 24 Aug 2023;
accepted 30 Sep 2023.

***Keywords:***
Containerized transportation;
Optimization; relocation strategies;
Heuristic.


*ABSTRACT*

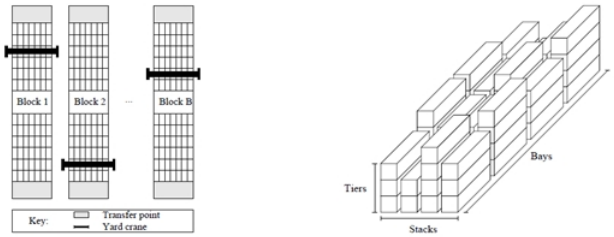
The container relocation problem supposes that the whole retrieval chain is identified in advance. This is practical for vessels where the shipment plan is known ahead in time. But, exact truck arrivals can barely be predicted and are exposed over time. As a result, the retrieval order is not known in advance. This paper deals with a dynamic and more practical version of the container relocation problem, where information about container retrievals becomes revealed over time.

## 1. Introduction.

### *1.1. Problem description.*

Container terminals have limited information on exact arrival times and on the arrival order of trucks. It is not uncommon, that terminals obtain this information only when trucks check in at the terminal gate. When processing the truck at the terminal gate, a container request is issued to retrieve the corresponding container from the storage area.

Figure 1: Container relocation problem.



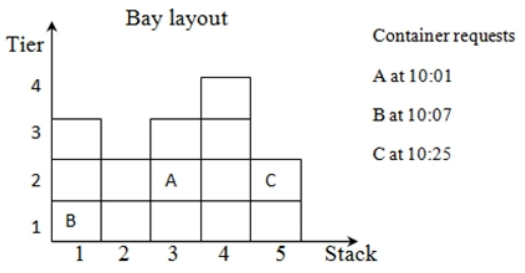

(a) A container yard (b) A single block

Source: Author.

The terminal operator has to decide in which order to serve the current container requests and where to relocate blocking containers. The yard of such a terminal is illustrated in Figure 1. The yard is divided into different blocks. Each block consists of several bays, each bay of several stacks and each stack of several tiers. Thanks to new technologies, the terminal knows exactly at which position (block, bay, stack, tier) each container is stored and which positions are empty.

The decision is based on known requests, since the terminal operators has no information on future retrievals. Figure 2 illustrates the dynamic container relocation problem. The number of relocations increases with the stacking height of containers and is therefore a bigger issue at terminals using stacking cranes for storage operations.

Figure 2: Dynamic Container relocation problem.




Source: Author.

[1]King Faisal University, Saudi Arabia.
[*]Corresponding author: Khaled Mili. E-mail Address: kmili@kfu.edu.sa.

The objective is to minimize truck service times. The order in which requests are served impacts truck service times and the number of relocations. Our main objective is to evaluate the benefit of knowing the retrieval sequence ahead in time, rather than evaluating different service policies. We suppose that trucks are served with a first-come, first-served policy. In this case, truck service times depend mainly on the number of relocations. Our objective is to minimize the number of relocations. The problem definition relies on assumptions A1 to A9.

A1: No new containers arrive during the retrieval process.

A2: Only the topmost container of a stack can be picked up. A relocated container can only be put on the top of another stack or on the ground.

A3: Containers are only relocated within the bay since relocations between bays are very time consuming.

A4: The bay size is limited by the maximum numbers of stacks and tiers.

A5: Containers in the same bay have the same size and can be piled up in any order.

A6: The distance traveled within one bay (horizontally and vertically) has little impact on the time to relocate or to retrieve containers.

A7: Only containers located above the current target container may be relocated.

A8: Container requests become known when trucks are processed at the terminal gate.

A9: Trucks are served with a first-come, first-served policy.

Like most other studies, we address the dynamic container relocation problem with precedence constraints among single containers and relocate only containers above the target container (A7). We call these containers blocking containers. We use the notation introduced by Caserta et al. (2012) to represent the container relocation problem. A bay consists of $W$ stacks and $H$ tiers. Each slot within the bay is addressed with coordinates $(i, j)$ where $i \in \{1, \ldots, W\}$ and $j \in \{1, \ldots, H\}$. The initial configuration contains $N$ containers, labeled $1, \ldots, N$. Containers have to be retrieved in ascending order, e.g. container 1 is the first one to be retrieved and container $N$ the last one. At each time period $t$ $(t = 1, \ldots, T)$, container

$n = t$ is retrieved and any blocking containers are relocated. The container labels are not known from the beginning, but revealed over time. To represent partly knowledge about the future retrieval sequences, we introduce a look-ahead horizon $D$ $(D \geq 1)$. It indicates that at each period t the exact retrieval sequence for the next $D$ containers is known: at period $t$, $Da$

$t = t$ is the first known retrieval container and $Db$ $t = t + D - 1$ the last known retrieval container.

### 1.2. Related literature.

To the best of our knowledge no scientific literature exists on the dynamic container relocation problem. But, several articles deal with the related stacking problem. The aim is to find good storage positions for incoming containers based on partial knowledge about their destinations, weights and departure times. The main objectives are to use the storage space efficiently, to reduce traveling times within the terminal and to reduce the number of relocations. Here, we only present studies aiming to minimize the number of relocations.

Dekker et al. (2006) and Borgman et al. (2010) evaluate different stacking strategies and the impact of available information via simulation. The performance of each strategy is measured via the number of relocations, the yard crane workload and the level of occupancy of the yard. They show that stacking containers on ground positions reduces the number of relocations. They compare scenarios with no information on future retrievals with scenarios with imprecise information on future departure times. Results show that using imprecise information increases the efficiency of the terminal. Park et al. (2011) present an online search algorithm to decide where to stack incoming containers. The algorithm tries variants of the best-so-far policy and can easily adapt to changes at the terminal. Results show that this algorithm can reduce quay crane delays, but does not obtain the best results for average truck waiting times.

Zhao and Goodchild (2010) use simulation to evaluate the use of information on truck arrivals to reduce relocations during the retrieval process. They run experiments for different levels of information and different bay configurations. Results show that already limited information on future arrivals can reduce the number of relocations. They also show that updating information in real time lowers information requirements. Jang et al. (2013) consider the problem with groups of homogeneous containers. They present a genetic algorithm for the case where the retrieval order of groups is known. They also present a statistical model to estimate the expected number of relocations when no information on future retrievals is available.

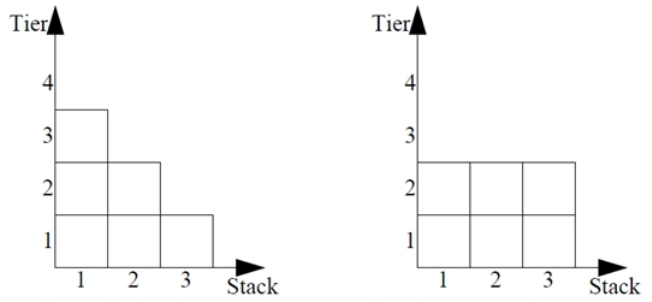
Yang and Kim (2006) consider the problem of stacking incoming containers in a way that minimizes the expected number of relocations. They relocate each container at most once. They address a static and a dynamic version of the problem. For the static problem, arrival and due dates of all containers are known in advance; for the dynamic version, arrival and due dates become known when containers arrive at the terminal. They use dynamic programming and a genetic algorithm to solve the static problem and use heuristics based on known departure times to solve the dynamic problem. Preston and Kozan (2001) present a container location model that minimizes the time needed to transfer containers from the storage area to vessels. This model includes traveling and relocation times. Khaled and Faissal (2012) formulate a mixed integer linear programming model and solve the problem via a genetic algorithm. Khaled (2014) develop a formulation and subsequent development of a Six Sigma approach solution for the problem. His work aims to develop a novel method based on a combined ANP and DEMATEL techniques to help container terminals determine critical Six Sigma transportation plans.

## 2. Expected value of relocations.

This section introduces a criterion to indicate the quality of a given bay layout if we do not have any information on future retrievals. In this case, all containers in the bay are equally

likely to be retrieved next. We determine the expected value of relocations, *EVR*, necessary to retrieve one container from the given bay.

Figure 3: Expected value of relocations EVR for two different layouts with 6 containers.




(a) Layout A, $EVR = 0.67$ (b) Layout B with minimal $EVR$, $EVR = 0.5$

Source: Author.

To do so, we compute the average number of blocking containers. Equation (1) defines the expected value *EVR*. It depends on the number of containers in the bay, $N\prime$, and on the number of containers per stack, $s(i)$ for all $i = 1, \ldots, W$. Figure 3 illustrates the computation on two examples:

$EVR_A = 1/6 \cdot ((0+1+2)+(0+1)+(0)) = 0.67$

$EVR_B = 1/6((0+1)+(0+1)+(0+1)) = 0.5.$

$$EVR = \frac{1}{N'} \cdot \sum_{i=1}^{W} \sum_{j=0}^{s(i)-1} j \tag{1}$$

***Lemma 1.*** *The minimum difference between the lowest stack and the highest stack equals 0*

*if $N\prime \bmod W = 0$ and 1 if $N\prime \bmod W \neq 0$.*

***Proof***. The minimum difference between the lowest stack and the highest stack is obtained if containers are evenly distributed among stacks. If $N\prime \bmod W = 0$, each stack has a height of $N\prime/W$; if $N\prime \bmod W \neq 0$ some stacks have height $\lceil N'/W \rceil$and others$\lfloor N'/W \rfloor$.

***Lemma 2***. *The expected value of relocations EVR is minimal if the difference between the lowest stack and the highest stack is minimal.*

***Proof***. We assume that for layout 1 the difference between the lowest stack and the highest stack is not minimal. Let $a$ be the lowest stack in layout 1 and $b$ the highest stack. We obtain layout 2 by moving the topmost container from stack $b$ to stack $a$. Let $EVR_{ab}$ be the expected value of layouts 1 and 2 without stacks $a$ and $b$. We compare the expected value of layouts 1 and 2.

$$EVR_2 - EVR_1 = \left(EVR_{ab} + \textstyle\sum_{j=0}^{s(a)} j + \sum_{j=0}^{s(b)-2} j\right) - \left(EVR_{ab} + \sum_{j=0}^{s(a)-1} j + \sum_{j=0}^{s(b)-1} j\right) = s(a) - s(b-1)$$

It is hence possible to reduce *EVR* by moving one container from the highest stack $b$ to the lowest stack $a$ as long as $s(a) < s(b) - 1$. Consequently, *EVR* is minimal if the difference between stacks $a$ and $b$ is minimal.

## 3. Different relocation strategies.

This section presents different relocation strategies that may be applied to the dynamic container relocation problem for a partial known retrieval order of length $D$.

*Strategy S1: Random heuristic*

For each container to be relocated, the heuristic randomly chooses a stack that is not full.

*Strategy S2: Leveling heuristics for D = 1 and D = 2*

The objective of the leveling heuristic is to relocate containers in a way that minimizes the expected value of relocations EVR. Lemma 1 and 2 show that containers should be distributed equally over stacks to minimize EVR.

For $D = 1$, only the current retrieval container is known. For each container to be relocated, the heuristic determines current stack heights and relocates the container to the lowest stack. If several stacks have the same height, the leftmost stack among them is chosen.

For $D = 2$, the heuristic uses information about the second retrieval container to keep it accessible. Like before it balances stack heights by relocating containers to the lowest stack.

But containers are only relocated on top of the second retrieval container if no other positions are free. If the second retrieval container itself has to be relocated, it is relocated to the highest stack.

The subsequent strategies S3 to S8 determine a partial solution to retrieve the next $D$ containers with a minimum number of relocations.

*Strategy S3: Relocations are updated every time new information becomes available*

The problem is solved repeatedly for each period $t = 1, \ldots, T$ with information on containers

$n = D_t^a, \ldots, D_t^b$. We initialize the model for period $t = 1$ with variables and constraints corresponding to periods $t = D_1^a, \ldots, D_1^b$and solve it. We then adapt the model to the next period $t = t + 1$ by adding variables and constraints corresponding to period $D_t^b$ (since variables and constraints corresponding to periods $D_t^a$ to $D_t^b - 1$are already in the model).

Since it is not possible to revoke decisions taken at earlier periods we fix variables representing container positions at the beginning of period $t$ according to the solution obtained in the previous iteration. We solve the updated model. The process ends when the time horizon is reached. At each iteration, the objective function (2) minimizes the number of relocations necessary to retrieve all $D$ containers for the given initial layout.

$$\min \sum_{i=1}^{W} \sum_{j=2}^{H} \sum_{k=1}^{W} \sum_{t=1}^{H} \sum_{t'=D_t^a}^{D_t^b} \sum_{n=t'+1}^{N} x_{ijklnt'} \tag{2}$$

*Strategy S4: Relocations are determined for the next D containers and are not updated*

The complete relocation sequence to retrieve the next $D$ containers is determined with the information on these $D$ containers. The solution is not updated if new information becomes

available. The problem is solved repeatedly at periods *1, 1+D, 1+2D,* . . . with information on the next $D$ retrieval containers. We initialize the model for period $t = 1$ with variables and constraints corresponding to periods $t = D_1^a, ...., D_1^b$. and solve it. We then adapt the model to the next iteration at period $t = t + D$ by adding variables and constraints for periods $D_t^a$ to$D_t^b$. To prevent revoking decisions taken at earlier periods, we set variables representing container positions at the beginning of periods $t - D + 1$ to $t$ to the values obtained in the previous iteration. We solve the updated model. The process ends when the time horizon is reached.

*Strategy S5: No detailed information about far-away retrievals*

We suppose that we know the exact retrieval sequence of the next $D$ containers. In addition, we know the subsequent $D\prime$ containers to be retrieved, but not their exact retrieval order.

Keeping subsequent retrieval containers $D\prime$ on top of stacks should reduce the number of relocations necessary at the next iteration to retrieve these containers. We introduce integer variables $a_{ijnt}$ that count the number of containers located above the next retrieval containers$n \in D'$:

$$a_{ijnt} = \begin{cases} 0 \; if \; container \; n \; is \; not \; located \; at \; position \; (i, j) \; at \\ \quad the \; beginning \; of \; period \; t, \\ \\ R \end{cases}$$

The objective function (3) penalizes the number of relocations for the current iteration (periods $D_t^a$ to $D_t^b$ ). In addition, it penalizes the number of containers above subsequent retrieval containers at the beginning of the next iteration at period$D_t^b + 1$. Constraint (4) defines variables $a_{ijnt}$ for period $D_t^b$ + 1for the next $D\prime$ retrieval containers.

$$\min \sum_{i=1}^{W} \sum_{j=2}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{t'=D_t^a}^{D_t^b} \sum_{n=t'+1}^{N} x_{ijklnt'} + w_1 . \sum_{i=1}^{W} \sum_{j=1}^{H-1} \sum_{n \in D'} a_{ijnD_t^b+1} \tag{3}$$

$$\sum_{j'=j+1}^{H} \sum_{n' \in N \setminus \{n\}} b_{ij'n'D_t^b+1} \leq a_{ijnD_t^b+1} + (H-1).(1 - b_{ijnD_t^b+1}) \tag{4}$$

$$\forall i = 1, ..., W, j = 1, ...., H-1, n \in D'$$

With strategies S6 and S7, we want to analyze the impact of different intermediate bay layouts on the total number of relocations. The objective is to be able to determine layouts that are advantageous with regard to unknown future retrievals. We compare two cases: distribute containers evenly among stacks and keep one stack empty. Again, only the initial layout for the next iteration (the layout at period$D_t^b + 1$) is of interest.

*Strategy S6: Distribute containers evenly among stacks*

The objective is to distribute containers evenly among all stacks to reduce the expected value EVR. We use integer variables $a_{ijnt}$ (introduced above) to count the number of containers located above each container. The expected value EVR is identical to the total of all$a_{ijnt}$.

The objective function (5) penalizes the number of relocations and to minimize the expected value at period$D_t^b + 1$. Constraint (**??**) defines variables $a_{ijnt}$ for period $D_t^b + 1$ for all containers.

$$\min \sum_{i=1}^{W} \sum_{j=2}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{t'=D_t^a}^{D_t^b} \sum_{n=t'+1}^{N} x_{ijklnt'} + w_1 . \sum_{i=1}^{W} \sum_{j=1}^{H-1} \sum_{n \in D'} a_{ijnD_t^b+1} \tag{5}$$

$$\sum_{j'=j+1}^{H} \sum_{n' \in N \setminus \{n\}} b_{ij'n'D_t^b+1} \leq a_{ijnD_t^b+1} + (H-1).(1 - b_{ijnD_t^b+1}) \tag{6}$$

$$\forall i = 1, ..., W, j = 1, ...., H-1, n = D_t^b + 1, ..., N$$

*Strategy S7: Keep one stack free*

The objective is to obtain a layout with at least one empty stack. This increases the expected value *EVR*. It might nevertheless be beneficial to have an empty stack to place containers in the next iteration. We add integer variables $f_{it}$and $e_t$to determine if at least one stack is empty.

$$f_{it} = \begin{cases} 0 \; if \; stack \; i \; is \; empty \; at \; the \; beginning \; of \; period \; t, \\ \\ 1 \; Otherwise; \end{cases}$$

$$e_t = \begin{cases} 1 \; if \; at \; least \; one \; stacki \; is \; empty \; at \; the \; beginning \; of \\ \quad period \; t, \\ \\ 0 \; Otherwise; \end{cases}$$

The objective function (7) penalizes the number of relocations and rewards an empty stack at period$D^b - t + 1$. Constraint (8) makes sure that $f_{it}$ equals 0 only if stack i is empty. Constraint (10) determines if at least one empty stack exists.

$$\min \sum_{i=1}^{W} \sum_{j=2}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{t'=D_t^a}^{D_t^b} \sum_{n=t'+1}^{N} x_{ijklnt'} - w_3 . e_{D_t^b+1} \tag{7}$$

$$\sum_{j'=j+1}^{H} \sum_{n'=D_t^b+1} b_{ijnD_t^b+1} \leq H . f_{i,D_t^b+1} \quad \forall i = 1, ..., W \tag{8}$$

$$\sum_{i=1}^{W} f_{i,D_t^b+1} + e_{D_t^b+1} \leq W \tag{9}$$

*Strategy S8: First-come, first served policy not necessary*

We want to analyze the impact of being able to serve the next D trucks in any order. This should decrease the number of relocations. A container blocking a retrieval container may itself be a retrieval container. In this case, it can be retrieved directly, rather than being relocated. Until now we imposed, that container n is retrieved at period n. Now, container $n$ may be retrieved at any period $n - D + 1, \ldots, n + D - 1$.

The objective function (10) minimizes the number of relocations. It takes into account that some containers $n < t$ may

be relocated at period $t$. Constraints (11) imposes that each container is retrieved within its time window. Constraint (12) makes sure that each container is retrieved exactly once. Variables $x_{ijklnt}$and $b_{ijnt}$for $n \leq t$ have to be added to existing constraints. A part from this, constraints remain identical and are not repeated here.

$$\min \sum_{i=1}^{W} \sum_{j=2}^{H} \sum_{k=1}^{W} \sum_{l=1}^{H} \sum_{t'=D_t^a}^{D_t^b} \sum_{n=t'-D+2}^{N} x_{ijklnt'} \tag{10}$$

$$\sum_{i=1}^{W} \sum_{j=1}^{H} \sum_{t'=t-D+1}^{t+D-1} y_{ijnt'} = 1 \qquad \forall n = 1, ..., N \tag{11}$$

$$\sum_{n=t-D+1}^{t+D-1} y_{ijnt} = 1 \qquad \forall t = 1, ..., T \tag{12}$$

## 4. Computational results.

We test strategies S1 to S8 on the instances sets 3-3, 3-4, 3-5, 3-6, 3-7, 3-8, 4-4, 4-5, 4-6, 4-7, 5-4 and 5-5 introduced by Caserta et al. (2012) and hence on $12 \cdot 40 = 480$ instances. All experiments are carried out on a computer with Inter(R) Xeon(R) CPU clocked at 2.67GHz (dual core), 3.48GB RAM and operating with Windows XP Professional. We limit the run time to 60 minutes per instance. Cplex 12.1 is used to solve the mixed integer programming models for S3 to S8.

Table1 summarizes the experimental settings. We test these strategies with different look-ahead horizons $D = 1, 2, 3, 5$ and $7$. For S5, we set $D\prime = D$. We impose a strict hierarchy to *i)* minimize the number of relocations per iteration and *ii)* optimize the initial bay layout for the next iteration. Weights $w_1$, $w_2$ and $w_3$ are defined based on the following observations. A relocation has a cost of 1. For S5, the maximum layout cost is obtained if all $D\prime$ containers are located at height 1 and $H-1$ containers are located above. For S6, the maximum layout cost is obtained if containers are stacked as high as possible. In this case, $\left\lfloor \frac{N}{H} \right\rfloor$stacks contain $H$ containers and one stack contains $N - H \cdot \left\lfloor \frac{N}{H} \right\rfloor$containers. For S7, the maximum benefit from one empty stack should be lower than the cost of one relocation.

Table 2 presents experimental results. It displays the numbers of solved instances (out of 480), the average numbers of relocations and the average run times of service strategies S1 to S8 for different look-ahead horizons $D$. Strategy SX-Y refers to strategy X with look-ahead horizon Y.

Table 1: Experimental setting for evaluating relocation strategies.

| Strategy | Look-ahead horizon D | Weights $w$ |
|---|---|---|
| S1 | n.a. | |
| S2 | 1,2 | |
| S3 | 3,5,7 | |
| S4 | 3,5,7 | |
| S5 | 3,5,7 | $w_1 = (D'.(H-1)+1)^{-1}$ |
| S6 | 3,5,7 | $w_2 = (\left\lfloor \frac{N}{H} \right\rfloor . \sum_{i=0}^{H-1} i + \sum_{i=0}^{N-H\left\lfloor \frac{N}{H} \right\rfloor - 1} i + 1)^{-1}$ |
| S7 | 3,5,7 | $w_3 = 0.5$ |
| S8 | 3 | |

Source: Author.

Table 2: Performance of relocation strategies S1 to S8 for different look-ahead horizons.

| Strategy | S1 | S2-1 | S2-2 |
|---|---|---|---|
| Solved inst. | 480 | 480 | 480 |
| Avg. Relocations | 18.9 | 15.0 | 14.0 |
| Avg. CPU time [s] | <0.1 | <0.1 | <0.1 |

| Strategy | S3-3 | S4-3 | S5-3 | S6-3 | S7-3 | S8-3 |
|---|---|---|---|---|---|---|
| Solved inst. | 480 | 480 | 480 | 480 | 480 | 460 |
| Avg. Relocations | 14.9 | 16.1 | 13.2 | 13.8 | 16.8 | 13.6 |
| Avg. CPU time [s] | 24.8 | 11.4 | 11.4 | 12.1 | 12.7 | 66.4 |
| Avg. time per iter. | 1.2 | 1.4 | 1.4 | 1.5 | 1.6 | 8.0 |

| Strategy | S3-5 | S4-5 | S5-5 | S6-5 | S7-5 |
|---|---|---|---|---|---|
| Solved inst. | 480 | 480 | 480 | 480 | 480 |
| Avg. Relocations | 13.1 | 14.6 | 12.5 | 13.2 | 15.0 |
| Avg. CPU time [s] | 28.1 | 10.6 | 10.8 | 12.9 | 11.4 |
| Avg. time per iter. | 1.5 | 2.2 | 2.2 | 2.6 | 2.3 |

| Strategy | S3-7 | S4-7 | S5-7 | S6-7 | S7-7 |
|---|---|---|---|---|---|
| Solved inst. | 480 | 480 | 480 | 480 | 480 |
| Avg. Relocations | 12.4 | 13.6 | 12.2 | 12.7 | 13.7 |
| Avg. CPU time [s] | 38.3 | 15.9 | 22.6 | 35.5 | 18.4 |
| Avg. time per iter. | 2.2 | 4.4 | 6.3 | 9.7 | 5.0 |

Source: Author.

*SX-Y represents service strategy X with look-ahead horizon Y.*

With strategies S1, S2, S3, S4, S5 and S7 all instances can be solved. One instance cannot be solved with S6-7 and 20 instances cannot be solved with S8-3.

Run times for strategies S1, S2-1 and S2-2 are fast enough to be applied in real-time at the terminal. For strategies S3 to S7, run times per iteration increase for bigger look-ahead horizons since the underlying models get bigger. But, for bigger look-ahead horizons less iteration is necessary and the total run time may decrease. If the truck travel time between the gate and the loading and/or unloading area may be used to determine relocation moves, run times per iteration for strategies S3 to S7 are also sufficient.

The numbers of relocations for all service strategies for different look-ahead horizons are presented in more detail in Figure 4. It also compares the dynamic results to the offline solution (Off) where the entire retrieval sequence is known in advance. The x-axis states the service strategy with the associated look-ahead horizon. The y-axis indicates the number of relocations. The box plots represent the number of relocations obtained for 459 instances (those solved by all strategies). Every

box plot indicates the median, the upper and lower quartiles and outliers for one relocation strategy.

Relocation strategies S2 to S7 with limited knowledge on future retrievals perform well.

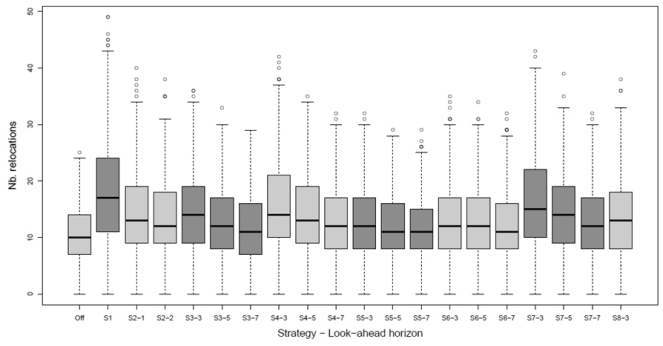
They outperform the random relocation strategy S1, but cannot reach the solution quality of the offline solution with complete knowledge. For each strategy, the number of relocations and their variances decrease when the look-ahead horizon increases.

Comparing strategies S3 and S4 suggests that it is beneficial to update relocation decisions every time new information becomes available. However, results of S1, S2, S3 and S6 show that for little information ($D \leq 3$) seeking a leveled bay layout may be more beneficial than updating relocation moves; for more information ($D = 5$, $D = 7$) results are similar.

Results for S1, S2 and S6, also show that the benefit of knowing more than the next 3 retrieval containers is limited. Comparing results S5-3 ($D + D\prime = 6$) with S3-5 and S3-7

($D = 5$ and $D = 7$) shows that knowing the exact retrieval order of far-away containers is of little benefit. Results of S7 show that keeping one stack empty decreases the solution quality since containers have to be stacked higher in the remaining stacks. Results of S4-3 and S8-3 show that serving trucks in any order rather than in FIFO order reduces the number of relocations.

Figure 4: Comparison of different relocation strategies for different look-ahead horizons.




Source: Author.

## Conclusions.

This paper presented the dynamic container relocation problem that has not been addressed in literature yet. We introduced the expected value of relocations EVR as an indicator to determine the quality of a bay layout with no information on future retrievals.

We proved that EVR is minimal for balanced stack heights.

We presented different relocation strategies for partial knowledge of the retrieval sequence.

We compared their solution qualities - indicated via the number of relocations for different look-ahead horizons. Results were also compared to a random relocation strategy and to the optimal offline solution obtained if the entire retrieval sequence is known in advance. It appeared that relocation strategies perform well and outperform the random strategy, but cannot reach the solution quality of the offline solution. Especially, strategies trying to balance stack heights perform well. Run times seem to be short enough to be applied at a terminal in real time.

To continue this work a more generic heuristic for $D > 2$ could be designed. This heuristic could try to balance stack heights and to relocate known retrieval containers using relocation rules from heuristic HC. It would also be interesting to evaluate the competitiveness ratio of the leveling heuristic to obtain more information on the worst case performance of heuristic S2.

For strategies S5 and S6, the layout obtained among those with the same number of relocations depends on cost parameters $w_1$ and $w_2$. The obtained layout strongly influences the number of relocations in the subsequent periods since the solution obtained at one iteration fixes the starting layout for the next iteration. It would be interesting to test how the cost parameters influence the solution quality.

It would also be interesting to evaluate the impacts of information on future retrievals and of the point in time when information becomes available (e.g., Wasesa et al.; 2011). This would make it possible to evaluate the potential benefit of new technologies providing the terminal with more details on truck arrivals.

Another approach to tackle the dynamic version would be stochastic programming to include uncertainty directly into the model. The problem can also be extended to deal with dynamic storage and retrieval requests simultaneously. In this case, the problem is to decide in which order to serve trucks, where to locate incoming containers and where to relocate blocking containers in order to minimize truck service times.

## References.

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

Dekker, R., Voogd, P. and van Asperen, E. (2006). Advanced methods for container stacking, *OR Spectrum 28*: 563–586.

Borgman, B., van Asperen, E. and Dekker, R. (2010). Online rules for container stacking, *OR Spectrum 32*: 687–716.

Park, T., Choe, R., Kim, Y. H. and Ryu, K. R. (2011). Dynamic adjustment of container stacking policy in an automated container terminal, *International Journal of Production Economics* 133: 385–392.

Zhao, W. and Goodchild, A. V. (2010). The impact of truck arrival information on container terminal rehandling, *Transportation Research Part E 46*: 327–343.

Jang, D.-W., Kim, S. W. and Kim, K. H. (2013). The optimization of mixed block stacking requiring relocations, *International Journal of Production Economics 143*: 256–262.

M. Khaled. (2014). "Six Sigma Approach for the Straddle Carrier Routing Problem", *Procedia - Social and Behavioral Sciences* Volume 111, (2014), Pages 1195–1205.

Mili, K.; Gassara, M. Multiple Straddle Carrier Routing Problem. *Journal of Maritime Research*, [S.l.], v. 12, n. 2, p. 63-70, july 2017. ISSN 1697-9133. Available at: <https://www-.jmr.unican.es/index.php/jmr/article/view/303>.

M. Khaled, M. Faissal. (2012), "Genetic procedure for the Single Straddle Carrier Routing Problem" *International Journal of Advanced Computer Science and Applications*, Vol. 3, No. 11, (2012).

Yang, J. H. and Kim, K. H. (2006). A grouped storage method for minimizing relocations in block stacking systems, *Journal of Intelligent Manufacturing 17*: 453–463.

Preston, P. and Kozan, E. (2001). An approach to determine storage locations of containers at seaport terminals, *Computers & Operations Research 28*: 983–995.

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

Wasesa, M., Muhammad, I. H. and van Heck, E. (2011). Improving the container terminal performance by incorporating location synchronization module to the pre-notification protocol, Unpublished paper presented at the *International Conference on Computational Logistics, Hamburg, Germany*, September 19-22, 2011.

Kim, K. H., Lee, K. M. and Hwang, H. (2003). Sequencing delivery and receiving operations for yard cranes in port container terminals, *International Journal of Production Economics 84*: 283–292.

Casey, B. and Kozan, E. (2012). Optimising container storage processes at multimodal terminals, *Journal of the Operational Research Society 63*: 1126–1142.