Rehandling Strategies for Container Retrieval

Tonguç Ünlüyurt and Cenk Aydin
Sabanci University, Faculty of Engineering and Natural Sciences
`e-mail: tonguc@sabanciuniv.edu`

1 Introduction

In this work, we consider the problem of optimizing the retrieval of containers from a bay in a fixed sequence. When there are other containers on top of the container to be retrieved, there may be alternative locations for those containers to be relocated. So it is essential to develop algorithms that find "good" locations in order to minimize the total time of the retrieval. In the literature, there are not many papers on this subject. The problem is discussed in some papers but only a few of them actually propose algorithms whose performance is demonstrated through extensive testing.

In Chen (1999), factors causing unproductive moves during storage from operational to strategic levels are discussed in a top to bottom perspective. Kim et al. (2000) derive a methodology to locate export containers within a bay. An optimization model based on containers' weight groups is formulated to find the location within the bay, minimizing the expected number of rehandles for an arriving export container. The container positioning problem is defined by Tranberg (2005). The problem is minimizing the total handling time of a block in an automated terminal. A linear mixed-integer model with a non-polynomial number of variables is formulated. In Lee & Hsu (2007), a multi-commodity network problem with side constraints is constructed for the pre-marshalling problem

Kim & Hong (2006) deals with the problem of the retrieval of import containers from a single bay and is closet to our study. The problem is to find exact locations of relocated containers while retrieving all the containers from a bay according to a predetermined order. Another version of the problem is also considered. In this version, the retrieval sequence is not given as a permutation of all individual containers, but the sequence is among groups of containers. The model is formulated and first solved via a branch and bound search with the objective of minimizing the number of relocations.

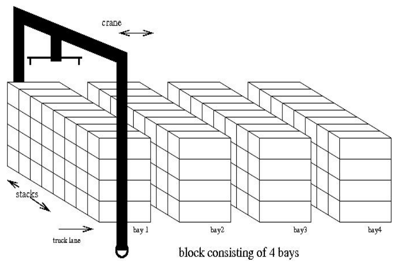

Figure 1: A typical block

Then a heuristic based on the expected additional rehandles is proposed. This heuristic determines the next move by minimizing the expected number of future relocations assuming that the containers will move randomly. The heuristic runs fast and gives results with about 3-16% optimality gap depending on the problem size.

In this study, we develop a branch and bound algorithm and heuristic algorithms for two objective functions and test the performance of these algorithms on randomly generated instances. The details of the algorithms can be found in Aydin (2006).

2 Problem Definition

Each block consists of a number of bays and each bay consists of a number of rows/stacks. Except for temporary storage all the containers are stacked in blocks. Mainly yard cranes are assigned to blocks for stacking operations. An illustration of how the blocks look like and how the yard cranes operate is given in Figure 1. In the figure, the block consists of 4 bays and each bay consists of 8 stacks of containers. There is a truck lane that the trucks use to bring the incoming containers and take away the outgoing containers. Typically a bay has between 2-10 rows and each row contains 3-7 tiers. There is no limitation on the number of bays but usually there are up to 20 bays and yard cranes can move between bays on their wheels.

Arrival and retrieval of the containers are performed by trucks. When a container arrives on a truck, as in Figure 2, the operator moves the crane to the right or left so that it would get on top of the truck. Then, the crane is lowered to pick up the incoming container. Once the container is picked up, the crane is levered up, to move on top of the available position that is allocated for the container. Then, the crane lowers down to release the container and is levered up to its usual position. In order for the crane to

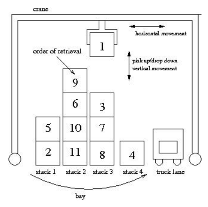

Figure 2: A single bay and crane movements

move horizontally, it has to be levered up.

The retrieval of a container is almost like an arrival. The only difference is that, the container to be retrieved should be accessible by the crane, meaning that there should not be any other containers on top of the container that is being retrieved. If a container is not accessible, then the containers above it should be rehandled/relocated to other available positions in the bay. Obviously rehandling causes operational inefficiency.

In this context, the related decision problem that we study can be described as follows: Given an initial configuration of a bay along with the sequence in which the containers will be retrieved, we would like to decide how to relocate the containers (when necessary) to minimize an appropriate objective function, that is related to the total time of the retrieval operation. We allow a container to be relocated to another position in the bay only when another container beneath that container is to be retrieved. In other words, we will try to devise a strategy that will determine how to retrieve the containers in a bay one by one in a predetermined sequence. When there are other containers on top of the container to be retrieved, the strategy will indicate where to relocate those containers.

In this work, two distinct objectives are considered. The first objective function is the number of relocations as in Kim & Hong (2006) (this will be referred as *Objective 1*). Secondly the number of relocations and horizontal movement of the crane are considered together. In this case, both quantities need to be multiplied by the associated constants such that we obtain the total time spent for relocations (pick up/drop down) and the horizontal movement of the crane. (this objective function will be referred as *Objective 2*)

3 A Branch and Bound Algorithm

We first propose a branch and bound procedure in order to solve the problem exactly. Our main goal is to compare our proposed heuristics' results with the optimal solutions when possible and to have an idea on the computation times of optimal solutions.

The state of the bay corresponds to the configuration of the bay at a certain time, i.e. the state of the bay describes which container is in which location at that particular time. An action consists of a series of movements of the containers. We will use the following notation introduced mainly in Kim & Hong (2006).

a) S^k: The state of the bay after k containers are retrieved from the bay.

b) a^k: The action taken for the removal of the k^{th} container.

c) $h(a^k)$: The number of relocations that occurred during action a^k, i.e. the number of relocations that occurred while the state of the systems changed from S^{k-1} to S^k as the k^{th} container is retrieved.

d) $F(S^k)$: The minimum total number of relocations to retrieve the remaining containers from the bay at state S^k.

The problem with *Objective 1* is formulated in Kim & Hong (2006) as in equation 1:

$$F(S^0) = \min_{a^1,a^2,...,a^k} \{\sum_{c=1}^{k} h(a^c) + F(S^k)\} \text{ where } S^{c-1} \rightarrow^{a^c} S^c \text{ for } c = 1, ..., k. \tag{1}$$

For formulating the problem with respect to *Objective 2*, we introduce the following notation:

a) $handle(a^k)$: The number of crane pickups experienced during action a^k when the bay is in state S^k.

b) $horizontal(a^k)$: The horizontal distance traveled by the crane during action a^k.

c) A and B are the appropriate coefficients for the pickups and horizontal distance such that total time can be computed as A $\times$ number of pickups and B $\times$ horizontal distance traveled.

d) $V(S^k)$: The optimal objective function value to retrieve up the remaining containers from the bay that is in state S^k.

The problem with *Objective 2* can be formulated as in equation 2:

$$V(S^0) = \min_{a^1,a^2,\dots,a^k} \{\sum_{c=1}^{k}(A \times handle(a^c) + B \times horizontal(a^c) + V(S^k)\} \quad (2)$$

where $S^{c-1} \rightarrow^{a^c} S^c$ for $c = 1, ..., k$. Essentially, for equation (2) we update the objective function with respect to *Objective 2*. Both for *Objective 1* and *Objective 2*, equations 1 and 2 indicate that after every movement of a container we have a new problem of the same kind to solve. For solving the problem for *Objective 1* and *2* optimally, a branch and bound search, with depth-first and backtracking strategies is implemented in C++.

In the branch and bound tree, each node corresponds to the current state of the bay. The children are produced from the parent such that the state of the bay corresponding to a child is obtained from the state of the bay corresponding to the parent by applying a specified action, i.e. a series of container movements. The effectiveness of our branch and bound scheme will be demonstrated in Section 5.

4 Heuristic Algorithms

4.1 Expected additional relocation heuristic (EAR1/EAR2)

To the best of our knowledge, the only proposed heuristic for the solution of the problem other than the branch and bound search, is suggested in Kim & Hong (2006), for minimizing the number of relocations (i.e. *objective 1*). In this algorithm, basically, the idea is to check each alternative location for the container that is being rehandled and estimate the expected additional relocations that would result by placing that container to the alternative location, assuming that further container movements will be random. Then the decision would be to relocate the container to the location with the minimum expected number of additional relocations. We have implemented this algorithm (EAR1) and adapted it also for *Objective 2*, EAR2.

4.2 Greedy Heuristic

The idea of expected additional rehandles relies on the assumption of random container movements. As a matter of fact, this will not typically be true since any methodology applied for such a problem will try to make moves to minimize the total time of retrieval. So assuming random future movements is rather pessimistic. Algorithms *EAR1* and *EAR2* choose the locations of containers based on calculations that are built on this assumption. Therefore we propose another algorithm which does not rely on such assumptions.

In the branch and bound search, we use an effective branching strategy and generate well structured trees for the optimal solution. In the greedy algorithm, we branch on the child that has the minimum lower bound for *Objective 1* and *Objective 2* respectively. We will refer to these algorithms as *Grdy1* and *Grdy2*.

4.3 Difference Heuristic

As an alternative to the heuristics proposed above, we propose another heuristic that requires fewer calculations. The idea behind the algorithm is to avoid further relocations as much as possible for each relocation. When container X is to be relocated, the stack it will be relocated to is chosen depending on the other containers of the candidate stack. The following rules are applied.

a) If there exists a stack such that the container that will be retrieved first in that stack will be retrieved later than container X, then that stack is chosen for relocating container X. If this is possible, a recognized rehandle becomes a realized rehandle. So we do not cause any additional cost as a result of this relocation. If there are multiple stacks that satisfy this condition, then the stack that has the container that will be retrieved earliest is chosen. Let us denote this container by Y. By minimizing the difference in the orders of retrieval, we minimize the number of containers ordered between X and Y, which will be rehandled again in the case of being relocated on X.

b) If a stack satisfying the above condition is not found then a stack with a container Z that is accessible (meaning at the top of the stack) by the crane and with an order number smaller than X is searched. In this way we stack containers which will be relocated in a reverse order so that they may become ordered when they are relocated. Again the difference between X and Z is minimized due to the same reasoning in the case of multiple stacks satisfying the condition.

c) If there is no stack satisfying conditions a) or b) , we simply minimize the difference between the order numbers of container X and the container which X will be relocated on.

So in each case we try to minimize the difference in the orders of containers to minimize the number of containers that would potentially be rehandled in the future. While doing this, the empty stacks are assumed to contain highest ordered container. We also modify this algorithm for *Objective 2* and refer to these algorithms as *Diff1* and *Diff2*.

5 Computational Results

A total of 8000 random instances are generated for different initial configurations, different width and height of bays and with different initial intensity values. The parameters that we use in our experiments are as follows:

a) 2 types of layouts: Balanced, meaning equal initial stack heights and unbalanced, meaning random initial stack heights.

b) 5 bay widths: 3 to 7 stacks.

c) 4 bay heights: 4 to 7 which is maximum number of containers in a single stack.

d) 5 initial intensity values: 55%-60%-65%-70%-75% of the bay is initially occupied,

We try to solve 40 instances for each combination of parameters optimally and by the heuristics described above (*EAR1/2*, *Greedy1/2*, *Difference1/2*) for *Objective 1* and *Objective 2*. All the algorithms are implemented on a Celeron 2.8 Ghz processor with 256 MB RAM.

Since the problems are generated randomly, in some cases out of the 40 problems for a certain set of parameters, some of the instances in a group cannot be solved to optimality. If this is the case, for all the 40 instances in that group, we consider the best solution obtained in 5 minutes to compare with the solutions obtained by the heuristics. With this distinction, we use optimal solutions for 7280 cases for *Objective 1* and 6400 cases for *Objective 2* out of 8000 cases for the comparisons.

For *Objective 2*, the parameters are assumed as A=5 and B=1. It turns out that the average computation time for the branch and bound algorithm for *Objective 2* is considerably higher than those for *Objective 1*. Also the average computation time for unbalanced cases is larger than those for balanced cases. In general, the computation times for the branch and bound algorithm very much depend on the size and intensity of the problems.

Next we compare the heuristic algorithms proposed in Section 4 among themselves and with the optimal solution, when possible, for *Objective 1* and *Objective 2*. Let us recall that *EAR1* is proposed in Kim & Hong (2006), for

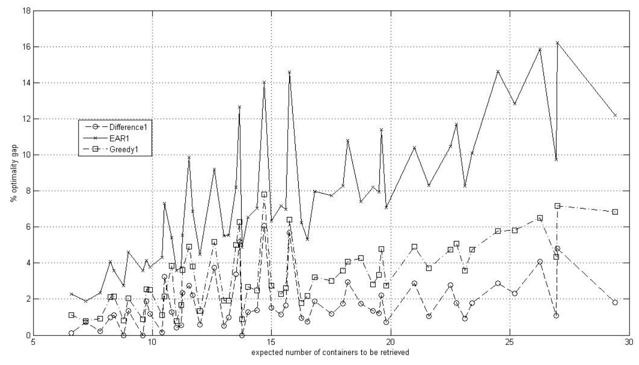

Figure 3: % optimality gap plotted against expected number of containers to be retrieved (7280 instances)

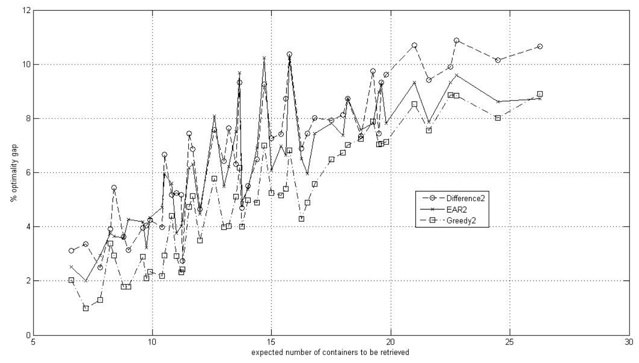

Figure 4: % optimality gap plotted against expected number of containers to be retrieved (6400 instances)

Objective 1. We implemented *EAR1* in addition to *EAR2* which we obtained by adaptation of *EAR1* for *Objective 2*. In addition we propose the heuristic algorithms *Grdy1/2* and *Diff1/2* for *Objectives 1 and 2* respectively.

We first plot the average percent optimality gaps versus the expected number of containers to be retrieved for all algorithms with respect to *Objective 1* and *Objective 2* in Figures 3 and 4. We observe that the optimality gaps increase in general as the expected number of containers increases for both objective functions. The *Diff1* algorithm outperforms the other algorithms for *Objective 1* while the *Grdy2* algorithm outperforms the other two heuristics for *Objective 2*. For *Objective 1* the average optimality gap for the Difference algorithm for any group of instances is below 6% for *Objective 1* whereas for *Objective 2* the average optimality gap for the *Grdy2* algorithm

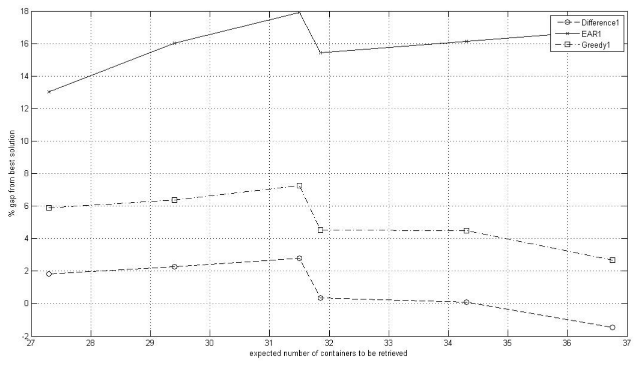

Figure 5: % gap from best solution plotted against expected number of containers to be retrieved (720 instances)

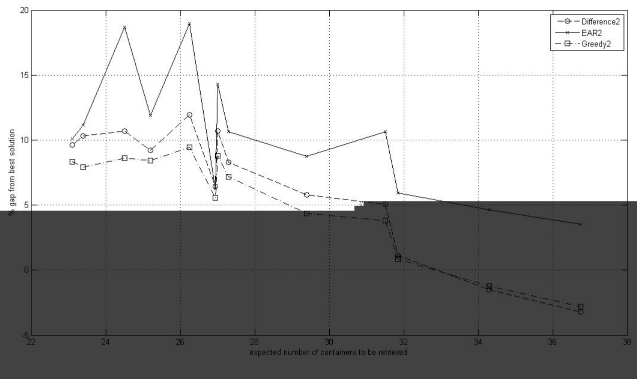

Figure 6: % gap from best solution plotted against expected number of containers to be retrieved (1600 instances)

is below 10% again for any group of instances.

For the groups of instances that contain large problems for which the optimal solution cannot be found in 5 minutes by branch and bound, we plot the gap of the heuristics as compared with the best solution that is found by the branch and bound algorithm in 5 minutes against the expected number of containers to be retrieved. We observe that as the expected number of containers to be retrieved increases, the quality of the solutions obtained by the heuristic algorithms and the best solutions found by the exact algorithm in 5 minutes becomes closer to each other. Again in these plots we observe that for *Objective 1*, the *Diff1* heuristic outperforms the other algorithms. As can be seen in the plot, for the largest instances the *Diff1* algorithm outperforms the best solutions found in 5 minutes. For *Objective2*, the *Grdy2* and *Diff2* perform similar to each other and they outperform *EAR2*.

6 Conclusions

In this paper, we study the problem of optimizing the retrieval of containers from their stacks. As a matter of fact, one could consider this problem in a more general setting where standard size boxes are retrieved from their locations in a certain order. This could be in the context of a warehouse, temporary storage area etc.

Future research directions in this area may be as follows:

a) For both objective functions heuristic algorithms that find near optimal cleaning moves can be developed.

b) One can consider the same problem when different container groups have different retrieval times. This situation could occur due to the weight or content of the containers.

c) One could develop similar heuristic algorithms that can handle precedence among groups of containers rather than a fixed complete sequence.

References

Aydin, C. (2006), Improved rehandling strategies for retrieveing containers from a bay, Master's thesis, Sabanci University.

Chen, T. (1999), 'Yard operations in the container terminal: A study in the unproductive moves', *Maritime Policy and Management* **26**(1), 27–38.

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

Kim, K., Park, Y. & Ryu, K. (2000), 'Deriving decision rules to locate export containers in container yards', *European Journal of Operational Research* **124**, 89–101.

Lee, Y. & Hsu, N. (2007), 'An optimization model for the container premarshalling problem', *Computers and Operations Research* **34**, 3295–3313.

Tranberg, L. (2005), Optimizing yard operations in port container terminals, *in* 'Proceedings of the 10th EWGT Meeting and 16. Mini Euro Conference', pp. 386–391.