Bachelor Thesis

July 19, 2011

Strategies for minimization of container relocations

A search for good stacking strategies for container retrieval by minimizing the number of container relocations. The strategies are tested for emptying bays and for keeping bays optimal in the long run.

Monique Gelten

Student number: 323861

Supervisor: Prof. Dr. Ir. Rommert Dekker

Second reader: Dr. Eelco van Asperen

Econometrics and Operational Research - Erasmus School of Economics

Erasmus University Rotterdam

Index

1. Introduction 2

1.1 Problem description 3

2. Literature 7

3. Main settings 8

3.1 Used parameters 8

4. Bay optimality 10

4.1 Imperfect containers 10

4.1.1 Lower bound 11

4.1.2 Important factors 11

4.2 Good containers 12

4.2.1 Important factors 12

5. Experimental setup 14

5.1 Case generation 14

5.2 Problem variants 15

5.3 Solving variants 16

6. Strategies 20

6.1 Hypothesis 22

7. Experimental outcomes 24

7.1 Influences of filling rate and tier/stack rates 24

7.2 Case 1 24

7.2.1 Different heuristic variants 24

7.2.2 Different solving variants 26

7.3 Case 2 30

8. Overall conclusion and notes 32

9. Appendix 33

10. References 35

1. Introduction

While the amount of container traffic is increasing, this cannot be said about the space at various container terminals. At these terminals containers are brought in and stored, waiting for their departure.

Because of the limited amount of space, these containers need to be stored in an efficient way which means stacking them is unavoidable.

Yard layout

A basic container yard layout consists of several blocks. At these blocks containers are piled as shown in figure 1.

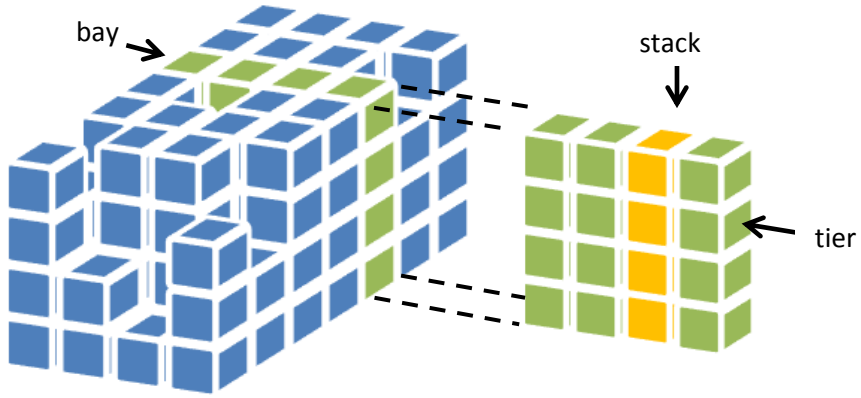

Figure 1: Container block: 7 bays, 4 stacks and 4 tiers.

This block has a total of seven bays in which containers can be stored, each bay having four stacks. The number of bays and stacks determine the space required for the block.

In this block, containers can be stacked up to a maximum of four high. Each height level is called a tier, which gives this block a total of four tiers. The more tiers, the more containers can be stacked in a certain area but also the harder it might get to retrieve containers.

Container handling

Containers are often moved from and to the right block and bay by transportation vehicles. A yard can have plenty of them. Moving containers from these vehicles onto the block or the other way around can be a problem. In some terminals this is done by cranes that have to move along the block to the right bay.

Once arrived they pick up and drop down containers, since this cannot be done by the transportation vehicles. This takes time and cranes can often just handle one container at a time. Another problem is that they can only move containers that are at the top of a stack.

Unlike the transportation vehicles, the number of cranes may be limited to just one or two per block, which makes it only more important to use them in an efficient way.

1.1 Problem description

Because of the limited number of cranes at the terminals, their efficient usage is very important. Travel time and handling time will have to be reduced as much as possible to make sure that these cranes can handle the work.

When it comes to travel time, the movement from bay to bay can be limited, as well as the vertical movement that the crane needs to adjust the level for picking up or dropping containers and the horizontal movement for moving above the different stacks. An example of such crane movements is given in figure 2. The handling time however is assumed equal for each container moved. It is the time to attach or de-attach a container.

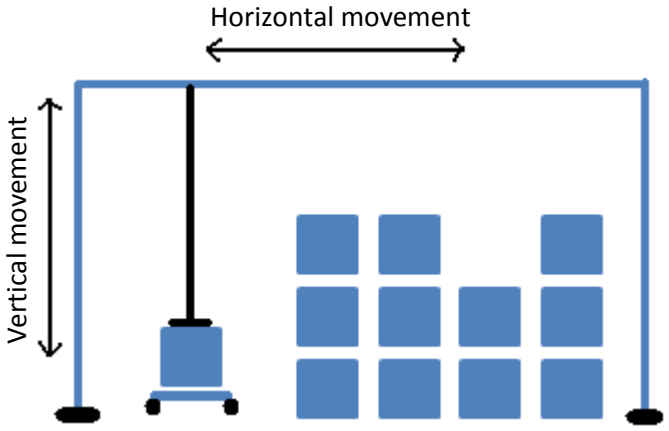

Figure 2: yard crane movements

Large problems arise when a container at the bottom of a stack is needed. If this happens, all containers above this container will have to be moved by the crane first. These extra container movements are called reshuffles or unproductive moves and can be very time consuming because of large amount of handling time.

Example 1: How reshuffles occur

In figure 3, the green squared container is needed, but it can only be retrieved by moving containers A and B first. This gives us two reshuffles.

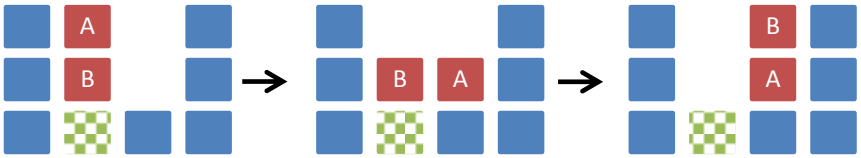

Figure 3: Movements to get to the green squared container.

It would have been more efficient, had this container been on the top of a stack.

The travel time will be left beyond the scope of this thesis, leaving the handling time as the main goal. Since handling time adds up for each container moved,

The reduction of the number of container movements will be the main goal.

Stacking strategies

To make everything run as smooth as possible, stacking strategies are used for determining where to place which containers. A stacking strategy is a set of rules to be followed while adapting the bay. The strategies determine which moves to make in which circumstances.

They will be compared based on the number of unproductive moves they bring.

In this thesis, containers are stored and/or retrieved in one bay with as less moves as possible by testing different stacking strategies.

The strategies can however also be used for an entire block, as travel times are being ignored. Distances between bays do not matter, so the three dimensional block can be sliced into layers giving us one large bay instead. This is shown in figure 4, where the block is sliced in the length.

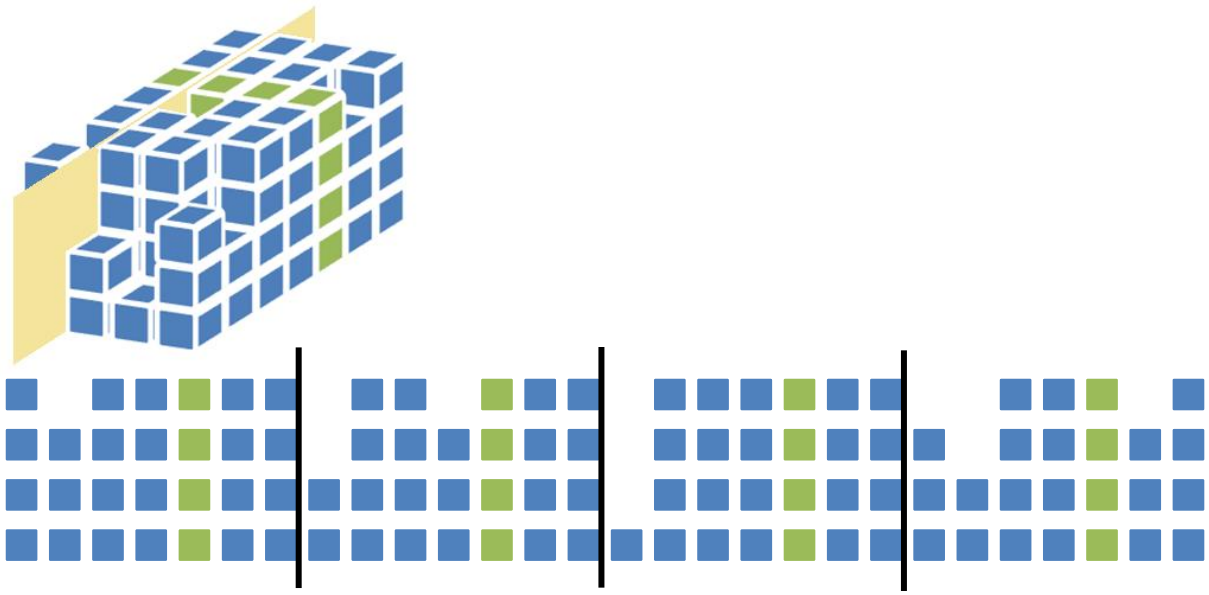

Figure 4: A sliced block.

In much existing literature, a distinction is made by seperating the problem in a restricted and unrestricted variant. In the restricted variant of the problem, the solution space is limited by only allowing relocating containers on top of the target container. In the unrestricted variant all top containers can be moved at all times. This unrestricted variant's solution space encaptures the restricted one's and its solution thus may be better. For large problems the solution space grows rapidly and therefor computation times of methods based on branch and bound might grow out of hand.

Research questions

Case 1: Blocks relocation problem

The main question to be answered is how to empty a partly filled bay of containers in sequence with as less moves as possible by using stacking strategies.With perfect information about the sequence this is called the blocks relocation problem.

It is to be find out what strategies work best and in which setups. Setups can vary by bay sizes or by the amount of space available.

Comparing is done against random solving and it is also checked what effects different setups have on this random solving.

The goal is also to see if a heuristic rule based on the unrestricted variant can lead to less reshuffles then heuristic rules based on the restricted variant. This should be possible because in the unrestricted variant an equally good or better solution then the restricted variant exists.

It is important to note that we assume perfect information on the retrieval times of all containers in the bay, so that the retrieval order is known and does not change. It is also assumed that there is enough space to retrieve all containers in sequence.

Case 2

In the second case, containers in a bay need to be retrieved in sequence. The difference is the addition of new containers. It is again assumed that exact retrieval times of all containers in the bay are known. The departure information on arriving containers is known as soon as they enter the bay. It is unknown when a container will enter.

The question is whether this different setup changes priorities between different strategies.

Thesis structure

Chapter two gives an overview of some literature on the container relocation subject. Chapter three gives some general information on the different cases and used parameters in this thesis. In chapter four the optimality of a bay will be discusses and chapter five holds information regarding the experiment. In chapter six the tested strategies are given, with some hypothesis on them. Chapter seven contains the outcomes and chapter eight a conclusion.

The appendix can be found under chapter nine, followed by the references in chapter ten.

2. Literature

For the container retrieval problem, exact solutions were found by Kim and Hong (2006) using a branch and bound algorithm for bays with 5 stacks by 5 tiers. For larger problems heuristics were used by estimating the expected number of reshuffles. The solutions found were all solutions for the restricted problem, because using branch and bound algorithms for the unrestricted problem leads to excessive calculation times.

Kim et al (1997) uses formulas to determine the expected number of reshuffles and Kim et al. (2000) proposes a decision tree to support real time decisions. The performance of this decision tree was evaluated by comparison with a slower dynamic programming method.

Kefi et al (2010) uses simulation and assigns slots with minimum costs, with costs depending on the expected unproductive moves.

Zhang et al (2010) uses iterative deepening A* algorithms combined with heuristics to find near optimal solutions for the restricted variant, finds new lower bounds for this problem and takes a look at the unrestricted variant of the problem.

Wu and Ting (2009/2010) used a beam search algorithm with look-ahead heuristic to find solutions for the restricted variant of the problem.

Caserta et al (2010) uses a corridor method inspired algorithm for the blocks relocation problem, where an exact method is applied over restricted portions of the solution space.

When it comes to realistic views on container stacking, Dekker et al. (2006) consider several variants of category stacking where categories are defined by export modality and in case of ships, the place of the container on the ship. They conclude that detailed simulation experiments could improve stacking performance.

Borgman et al. (2010) evaluated the trade-off between stacking further away in the terminal versus stacking close to exit points and accepting more reshuffles. They also concluded that departure time information, even imperfect, can lead to improvements.

Caserta et al. (2010) gives an overview of recent contributions to container handling. The relocation problem is explained and its relevance and connections with other issues are offered.

Part of the existing literature focuses on exact solutions and part focuses on heuristics. Simulation is also often used to find realistic results. When it comes to the heuristics, the focus often lies on the restricted variant of the problem.

This thesis will not concentrate on exact solutions but will include heuristics on both the restricted and the unrestricted variant of the problem.

3. Main settings

The search for good strategies will be split into two different cases. In both cases strategies will be compared and conclusions will be drawn. This chapter explains the cases and introduces the parameters that are used throughout the thesis. The first case is the blocks relocation problem. The second case is used to see if heuristics still perform well in the long run by constantly adding containers. Though information is assumed perfect, the second case comes a little closer to the realistic problems in container terminals.

Case 1: The container retrieval problem

In the container retrieval problem, a partly filled bay is given. The goal is to retrieve all containers in this bay in sequence with a minimum number of moves.

The total number of moves equals the number of containers in the bay (which have to be retrieved) plus the number of reshuffles. Since the number of containers in the bay is fixed, minimizing the number of moves comes down to minimizing the number of reshuffles.

Different stacking strategies will be used for emptying bays of variant sizes.

Case 2: The container retrieval and storage problem

In the second case, a partly filled bay is given as a starting point. The bay will continuously be changing as containers arrive and leave. Findings from case 1 will be put to the test to see how they do in the long run in this everlasting bay.

3.1 Used parameters

In this thesis bays are shown in figures as stacked blocks with printed numbers on them. These numbers indicate the retrieval order of the container. The higher the number, the later the container will be retrieved. The bays have varying sizes, indicated below the figures.

The container that has to be retrieved first will be called the target container and will be colored green.

Example 3: Bay with retrieval order

Figure 5 shows a bay with 4 tiers and 5 stacks. The container that will need to be retrieved first is container number 1, followed by number 2.

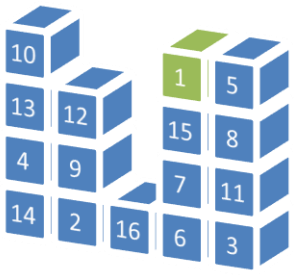

Figure 5: 4x5 bay

Throughout the thesis some parameters are used, which will be explained in this section.

T: The total number of tiers in the bay. T gives the maximum stacking height.

S: The total number of stacks in the bay. S gives the maximum number of locations on which a container can be stored.

A: $\in \{0,1, \dots, TS\}$. Number of containers in the bay. A can never exceed TS and can never be negative. In this thesis, A will never exceed TS + 1 - T (as explained in section 5.2)

The retrieval time is declared as the time in which a container will be retrieved. In this thesis' figures, the retrieval time is given as a number which can only be used to determine the sequence in which the containers leave. It thus does not mean that container 4 stays twice as long as container 2, only that it stays longer than container 2.

R_c : The retrieval time of container c

R_t: The retrieval time of the target container.

R_{max}: The retrieval time of the container in the bay that has the latest retrieval time.

H_i: Height of stack i, or the number of containers in stack i, $0 \leq H_i \leq T$ $i \in \{1,2,..S\}$

L_i: Minimum retrieval time from the containers in stack i. If stack i is empty, L_i is set to R_{max} +1

Example 4: Parameter example

For the bay of figure 6 the following statements hold:

S=5	$H_1 = H_4 = H_5 = 4$	$L_1 = 4$
T=4	$H_2 = 3$	$L_2 = 2$
A=16	$H_3 = 1$	$L_3 = 16$
		$L_4 = 1$
		$L_5 = 3$

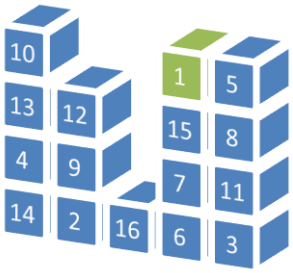

Figure 6: A 4x5 bay.

If container 16 leaves in 16 periods, then $R_{max}=16$

4. Bay optimality

When moving or placing a container, in most cases there are different options for doing this. The goal is to choose these movements that keep the number of relocations as small as possible. One has to make a tradeoff between the number of relocations the move will surely bring and the number of relocations it might bring in the future.

When analyzing the bay the containers are separated into two types:

-imperfect containers

-good containers

For both types ideas are shown on how to determine which places are to be preferred and which are not.

4.1 Imperfect containers

Imperfect containers are containers that have to be moved at least one time before their own retrieval. This is due to a container with an earlier retrieval time being positioned underneath them. In figure 7 all red colored containers are in an imperfect position.

Figure 7: A 4x4 bay with 6 imperfect red containers.

4.1.1 Lower bound

These imperfect containers will have to be moved in the future for other containers to be retrieved and thus give us a lower bound of the minimum number of reshuffles needed to empty the stack. This simple lower bound can be computed by summing up over all containers that have a container with smaller retrieval time underneath them.

In the bay of figure 7 there is no solution for emptying the stack in less than 19 movements: 13 retrieval moves and at least 6 reshuffle moves.

4.1.2 Important factors

One would like to have as few imperfect containers as possible, as this gives a smaller lower bound. Lower bounds are not guaranteed to be possible solutions, so not only the number of these containers is important.

There is a possibility that when moving an imperfect container, a good place cannot be found and this container has to go to an imperfect place once again. This process continues until a good location can be found with the option of leaving the container there until its retrieval.

This good location exists when the retrieval time of our container R_c is smaller than or equal to that of the lowest container of stack i for at least one i for which the stack height is smaller than the number of tiers.

This holds if R_c is smaller than or equal to the highest lowest container of all stacks with stacking space.

$$R_c \leq \max_{i|H_i<T} L_i$$

Moment of retrieval

The retrieval time of these containers may influence the optimality of the bay. Container 12 in figure 7 has a high number which corresponds with a retrieval time laying relatively far in the future. The higher the number of this container, the more difficult to stack it on a good location. As a matter of fact, for retrieving container 1, container 12 will have to be moved to an imperfect location once more, giving another extra reshuffle move. If this container had number 8 or lower, there would have been no problems moving it.

Given a stack, and a container c that has to be placed, the shorter the retrieval time of this container, the higher the probability that it is smaller than L_i, and can be placed at a good location.

$$The\ lower\ R_c, the\ better$$

Moment of reshuffle

Imperfect containers can be moved whenever one wants, but there is a moment at which a movement must be made: when a container underneath it is being retrieved.
It is questionable whether the imperfect container can be replaced to a good location. This depends on the containers with smallest retrieval times in each stack: L_i.

If the number of containers in the bay with an earlier retrieval time than our imperfect container decrease over time, then L_i might increase and so may the probability of finding a good location.

Also, the further away in time this latest reshuffle possibility, the more moments on which moving the container is possible. The stack is constantly changing so if more turns can be waited, more stacks come by and the probability raises that there might come a stack in which there is a good place to move it.

Given a number of imperfect containers, it is preferred to have as less of them on top of containers with early retrieval times.

4.2 Good containers

Containers that are stacked properly will only have to be moved for their departure. We will call these containers, that are possible to keep without reshuffles, “good containers” and their places “good positions”.

Because of future arriving containers or imperfectly placed containers that have to be removed, we want these good containers to be placed as optimal as possible. That means: place them so that the amount of reshuffles during retrieval are minimized.

4.2.1 Important factors

As long as containers can be placed to a good position, the amount of reshuffles does not have to increase. To prevent containers having to go to imperfect positions, we want as many future containers to be able to move to a good position as possible. A container can get a good place on stack i if:

(1) there is stacking space in this stack: $H_i<T$

(2) it has an earlier retrieval time than all other containers in the stack: $R_c<L_i$

For (1) holds that this is the case when the stack is not at full height. An idea would be to try to keep the top tier available when possible.

For (2) it holds that one should try to keep L_i as large as possible. Here one could think of keeping all L_i as high as possible by maximizing their sum or to try and keep just one or two L_i extra high, possibly by keeping a stack empty.

Example 5 gives an example of a bay where the sum of L_i is kept as high as possible.

Example 5: Stack preference

Figure 8a shows the target container 2, which can only be reached by moving container 4.

Container 4 has an earlier retrieval time than all containers on the second and fourth stacks. Since there is also stacking space available on these two stacks, this gives us two good locations for container 4:

Container 4 → stack 2:

L_1	L_2	L_3	L_4
3	4	2	9

$\sum_{i=1:4} L_i = 18$

Container 4 → stack 4:

L_1	L_2	L_3	L_4
3	6	2	4

$\sum_{i=1:4} L_i = 15$

Based on maximizing the sum of Li, the second option is preferred.

Figure 8b shows the movements made when moving container 4 to stack 2 and figure 8c when moving container 4 to stack 4. This shows that in this example the first option was indeed better.

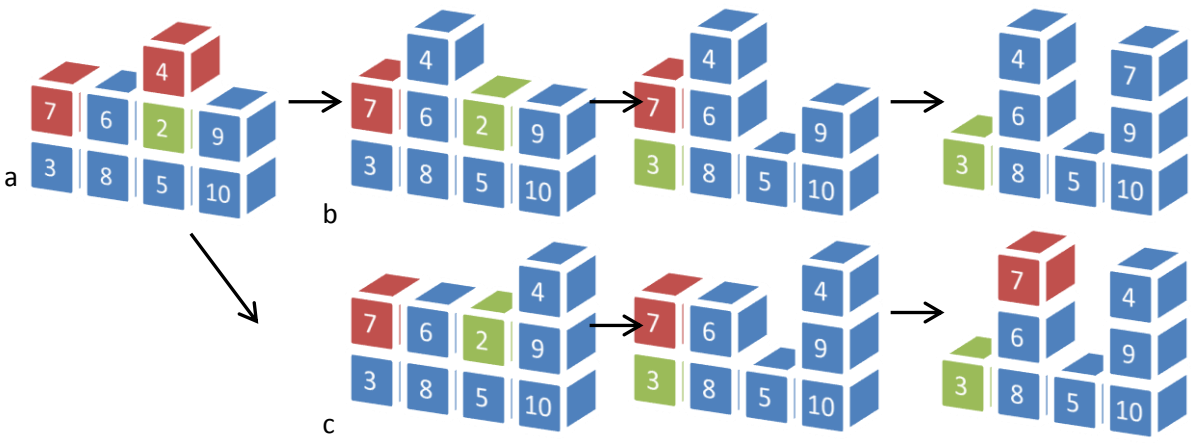

Figure 8: Movements for reaching container 2

If a stack is at full height, the value of L_i is less important because no containers can be stacked on top anyways. For stacks with only a few containers L_i might be more important because lots of containers can be stacked on top.

5. Experimental setup

5.1 Case generation

In both case 1 and 2 a bay has to be generated beforehand. Case 2 also requires arriving containers. This section shows how the bays and arrivals were generated.

Generating the bay

Data will be created by generating a stream of containers with random leaving order. The order is given by numbers, where lower numbers leave first. Bays can then be filled by taking these containers and storing them on randomly chosen stacks. If a stack is at full height, a different stack will be chosen.

An example of such a stream and bay are given in figure 9. The number of generated containers, as well as the sizes of the bays can easily be adapted.

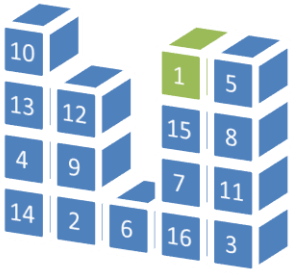

Figure 9: A randomly filled bay of 5 stacks and 4 tiers, 80% filled.

Generating the arrivals

For the second problem the bay is initialized as above, however now arriving containers also have to be generated.

After each container departure, an arrival will or will not be generated depending on the current number of containers: the more containers, the lower the probability of a new container arriving.

This means that the number of containers will lay around a certain filling rate. If the number deviates too much from this rate, arrivals will occur with a probability of zero or one. This is to prevent bays from running empty or overly filled and to be able to analyze results based on filling rates.

Arriving containers will also need a departure time. This time is assumed uniformly distributed between the minimum and the maximum departure time of the stacked containers. It could thus be that the arriving containers leaves very soon, but it might as well be that the container will be stacked for a very long time.

Getting the results

After generating the bays, different heuristics will be followed to empty them. For each bay and each heuristic, the amount of reshuffles is counted.

To compare with random retrieval, bays are also emptied in a random way. This is dome 1 or 100 times, in the last case for each bay the best solution will be stored.

5.2 Problem variants

Tier and stack sizes

For a given filling rate, the larger the number of tiers relative to the number of stacks, the more containers stacked on top of each other. The percentage of top containers is lower and the probability of reshuffles occurring raises. For a lower tier number stacking rules might have less influence on the number of reshuffles than for higher tier numbers.
The number of tiers relative to the number of stacks will thus be varied. This is called the tier-stack rate:

$$Tier/stack\ rate = T/S$$

The larger the number of stacks and tiers, the larger the bay, the more stacking possibilities and the bigger the problem:

$$Stacking\ space = T \cdot S$$

Different sizes will be tested to make sure that heuristics can be used for bigger instances too and to check whether some heuristics perform better than others under specific tier-stack rates.

Filling rates

The cases will be tested for different filling rates. The higher the filling rate, the less choices for stacking a container and more reshuffles will be needed on average.
The filling rate is given by the number of containers divided by the amount of stacking space:

$$Filling\ rate = A/T \cdot S$$

If the number of containers equals or is smaller than the number of stacks, containers can always be moved to an empty stack, giving a zero-reshuffle solution. Because of the easiness of this problem the minimum number of containers is set to one container more than the number of stacks:

$$A_{minimum} = S + 1$$

It must always be possible to get to the target container. When stacking T containers high, there can at maximum be T-1 containers on top of the target. If during the retrieval process there are at least T-1 spaces available, the target container can always be reached.

We make sure emptying the bay is possible for all random fillings by setting the maximum number of containers equal to:

$$A_{maximum} = T \cdot S - (T - 1)$$

where TS, the total stacking space, equals the number of tiers multiplied by the number of stacks and T – 1 the needed space.

5.3 Solving variants

Base vs. Extended: One container vs. multiple containers at a time

In both case 1 and 2, containers have to be retrieved. If multiple containers are blocking the target container, then multiple containers will eventually have to be moved. One can determine the best destination for one container at a time. This is done in the base version. If there are multiple containers on top of the target, a destination is chosen for each container separately.

In a bay of T tiers and S stacks, there are a maximum of T-1 containers on top of the target which can be moved to a maximum of S-1 stacks (all stacks except for the stack where the target is at). This gives a maximum number of (T-1)*(S-1) possibilities and T-1 decisions; one for each container.

Example 6: Base version

Figure 10 gives a bay with 3 tiers and 4 stacks.

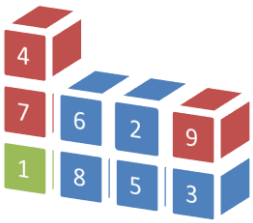

Figure 10: A 3x4 bay

The target container 1 is blocked by both containers 4 and 7.

In the base version, all possible movements of container 4 are examined first as in figure 11.

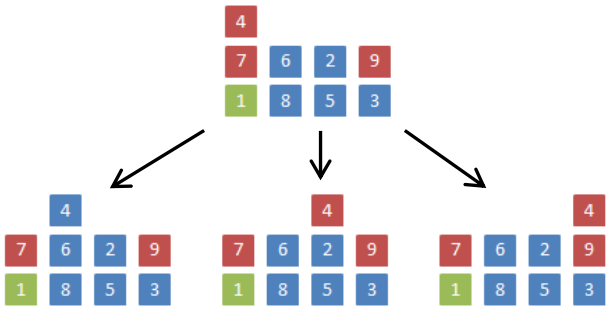

Figure 11: Possible movements for container 4 in a 3x4 bay

One solution is chosen and now container 7 is up next. Figure 12 shows all possible movements of container 7. A solution is chosen again and container 1 can be retrieved.

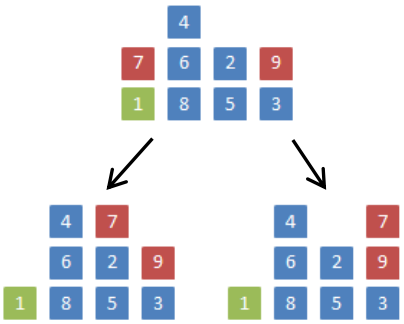

Figure 12: Possible movements for container 7 in a 3x4 bay

This process needed two decisions and evaluated a total of 5 possible movements.

In the extended version solutions may also be found by solving the problem by evaluating multiple containers at a time. This might prevent containers from being moved to locations that would be better for containers having to be moved in the next few steps.

All possible combined movements for the containers blocking the target are examined. This gives us a larger set of possibilities to choose from, since all possibilities of the base version form a subset of the extended version. This extended version might prevent suboptimal moves from being made so it can give better solutions then the base version.

In a bay of T tiers and S stacks, there are a maximum of T-1 containers on top of the target which can be moved to a maximum of S-1 stacks (all stacks except for the stack where the target is at). This now gives a maximum of $S\text{-}1^{T-1}$ different ways to stack these containers with only one decision to make per target retrieval.

Example 7: Extended version

Figure 13 gives a bay of 3 tiers and 4 stacks. This gives:

T=3

S=4

$S\text{-}1^{T-1} = 3^2 = 9$

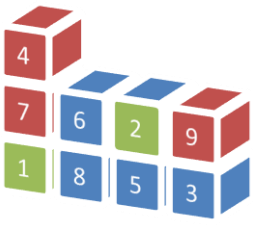

Figure 13: A 3x4 bay

All 9 possible movements are given in figure 14. Of these 9, only 6 are feasible given the maximum height of the stacks. $S\text{-}1^{T-1}$ thus gives a maximum on the number of different ways of stacking.
The best bay out of these 6 bays is chosen.

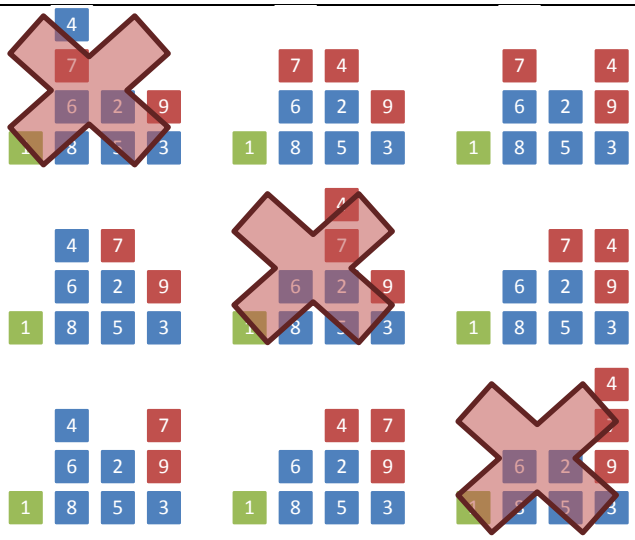

Figure 14: Possible movements of container 4 and 7

Limiting the extended variant

In every sub step of the problem a container has to be retrieved and all containers above it will have to be moved.

Although in practice there might be limited space so that not all $S\text{-}1^{T-1}$ distributions are possible, when T and S increase, the number of possibilities to check increases as well.

Table 1 shows that an increase of tiers has a huge impact on $S\text{-}1^{T-1}$ especially when the number of stacks is large.

To limit computational time, the number of containers for which all possible movements are being checked can be limited to a small number.

Number of tiers	Number of stacks								
	2	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
2	1	2	3	4	5	6	7	8	9
3	1	4	9	16	25	36	49	64	81
4	1	8	27	64	125	216	343	512	729
5	1	16	81	256	625	1296	2401	4096	6561
6	1	32	243	1024	3125	7776	16807	32768	59049
7	1	64	729	4096	15625	46656	117649	262144	531441
8	1	128	2187	16384	78125	279936	823543	2097152	4782969
9	1	256	6561	65536	390625	1679616	5764801	16777216	43046721
10	1	512	19683	262144	1953125	10077696	40353607	1,34E+08	3,87E+08

Table 1: $S\text{-}1^{T-1}$ for different stack and tier numbers.

Restricted vs. unrestricted

When the target container cannot be retrieved because one or more other containers are blocking it, other movements have to be made first. One can choose to only move these blocking containers as in the previous example. This gives the restricted problem variant where only containers on top of the target can be moved. This seems a logical assumption because this is the fastest way to get to the target. The possibility of moving all top-containers however, might give better solutions in the long run as the restricted variant may not give an optimal solution.

It is therefore important to look at both the restricted and unrestricted forms of all problems.

Example 8: Restricted vs. unrestricted

In figure 16 the restricted variant of the problem would move container 7 to the good spot straight on top of container 9. After the retrieval of container 2, container 8 has to be moved to an imperfect position.

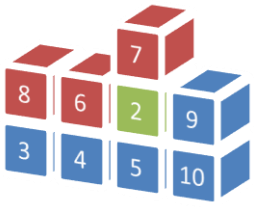

Figure 15: A 4x4 bay

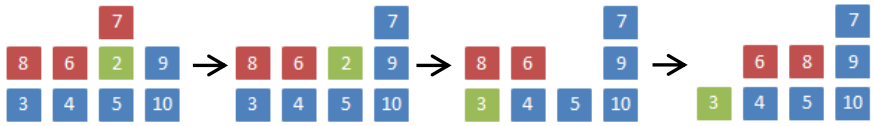

Figure 16: Movements in restricted version

Container 8 has to be reshuffled anyways so in the unrestricted variant of the problem, this container can be moved on top of container 9 before container 7 is moved as can be seen in figure 17. This will eventually lead to a solution with less reshuffles.

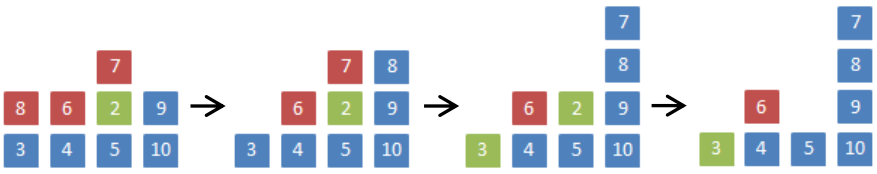

Figure 17: Movements in unrestricted version

6. Strategies

In the first setting a partly filled bay is given and the objective is to empty this bay in as few moves as possible. In the second setting containers will arrive as well. Here the bay will not be emptied and the problem goes on. Different strategies will be compared for both settings.

Random solving

To compare the strategies, random solving is applied. In this method only containers on top of the target container are moved and they will be moved to randomly chosen stacks. This gives a guaranteed solution because there can only be limited number of containers on top of each target and a target is removed from the stack whenever possible.

Comparison will be done against outcomes of this random solving method but also against the best outcomes out of 1000 or even 10000 trials. This is done because this solving does not take much computational time and it would be unfair to conclude a better strategy has been found if it cannot even standup against 1000 random trials.

This random solving method will also be used to analyze the influence of filling rates and tier/stack rates.

Restricted variant

In this sub-case only containers above the target can be moved. The main goal for these containers is to be placed at a good location.

All possible container replacings will be considered, for either one container at a time or multiple containers. A solution with minimum number of imperfect containers is chosen.

Decision rules

If there are multiple solutions which give the same number of imperfect containers, a decision from these solutions has to be made. The following decision rules will be tested:

A Pick a random solution.

B Place as less containers on the next target as possible.

Each container placed on the next target is going to have to be moved again in the next turn, which gives us another reshuffle in a very short time period.

C Pick the solution in which the imperfect containers on average have the containers underneath them with later departure times. For all imperfect containers in all i stacks, L_i will be summed up.

D Maximize the sum of L_i, the smallest departure times in each stack.

E Maximize the weighted sum of L_i, with weights depending on the height of the stack.

Once a decision has been made, containers will be moved, the target will be retrieved and a next target container is chosen.

Unrestricted variant

In the unrestricted variant it is possible to move all top containers.

Just as in the restricted case, solutions for retrieving the target container with minimum number of imperfect containers are found.

If we call this number of imperfect containers M, than it might be possible by moving one other container to lower this M.

For all top containers movements will be tested and lower bounds of the next sub solutions will be compared. If M decreased by one, this makes up for our extra move. If M decreases by more than one, it is even more worth the try.

Example 9: Unrestricted variant strategy

In figure 18a there are 11 imperfect containers. Container 14 has to be moved in order to get to container 3. Doing this, M will be equal to 10 since container 14 can be moved to a good place.

Top containers 18, 19 and 11 however might be moved as well. For all of these containers it holds that they have to be reshuffled anyways.

Moving them to a perfect location will guaranty a smaller lower bound but when moving container 11, in the next step container 14 cannot be moved to a perfect location. This means that when moving container 11 to the empty stack, M will not decrease because container 14 will become an imperfect container.

For container 18 and 19 holds that moving one of these to the third stack will decrease the number

of imperfect containers by one. It will also decrease M by one. Container 19 has our preference while this container is harder to stack on a good location and more containers can be stacked on top of it so it will have a slighter decrease on L_3.
Of course, now container 18, followed by 17 and 16 might also be moved to the third stack, followed by container 14 when no other moves are available that give a smaller lower bound.

In this example all moved containers were imperfectly stacked ones. This does however not always have to be the case.

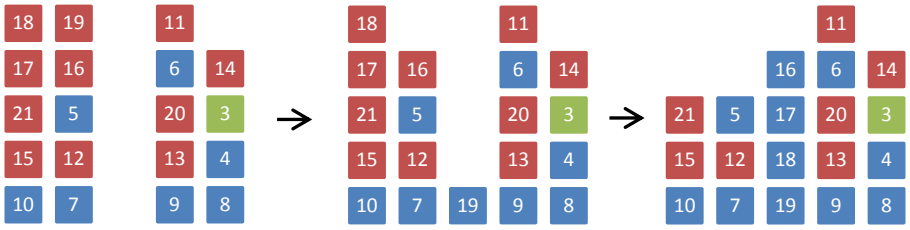

Figure 18a: 11 imperfect containers

b: 10 imperfect containers, 1 move later

c: 7 imperfect containers, 4 moves later

6.1 Hypothesis

Random solving

Random solving is expected to give a solution on which improvement should be easy. Using the best solutions out of a 1000 trials should overall give better results, but in general no optimal results. Using even more trials should improve results further but the gain will get smaller, the more trials will be done.

The more containers in a bay or the more destinations they can be moved to (in general: the more stacks) the larger the amount of steps to come to a solution and the larger the amount of different possibilities in each step. The larger the amount of containers and/or stacks, the larger the solution space and thus the smaller the percentage of this space visited for a given amount of trials. It is to be expected that random solving will start to deviate more from the optimal solution more rapidly than the heuristics.

Unrestricted vs. restricted

If the unrestricted variant was solved to optimality, it would give a better (or equally good) solution

than the restricted variant, since the solution space of the restricted variant is a subset of the unrestricted one. Knowing that this unrestricted variant can give us better solutions, this is exactly what is to be expected. While branch and bound methods can give problems for this unrestricted variant because of this much bigger solution space, the stacking strategies in this thesis only look one sub solution ahead, hopefully keeping the calculation time low enough.

Base vs. extended variant

The extended variant takes more computing time, especially as more containers are stacked on the target container. The expectation is that this extra computing time will be worth it and better solutions can be gained. It might be needed to set the maximum number of containers (for which all move combinations and their solutions are checked) to a limited number to improve the computing time.

Decision rules

It is of course to be expected that rules B, C, D and E perform better than rule A, since rule A picks a random solution with minimum number of imperfect containers. It is also to be expected that rule E will perform better than rule D since this is an extension of it.

7. Experimental outcomes

7.1 Influences of filling rate and tier/stack rates

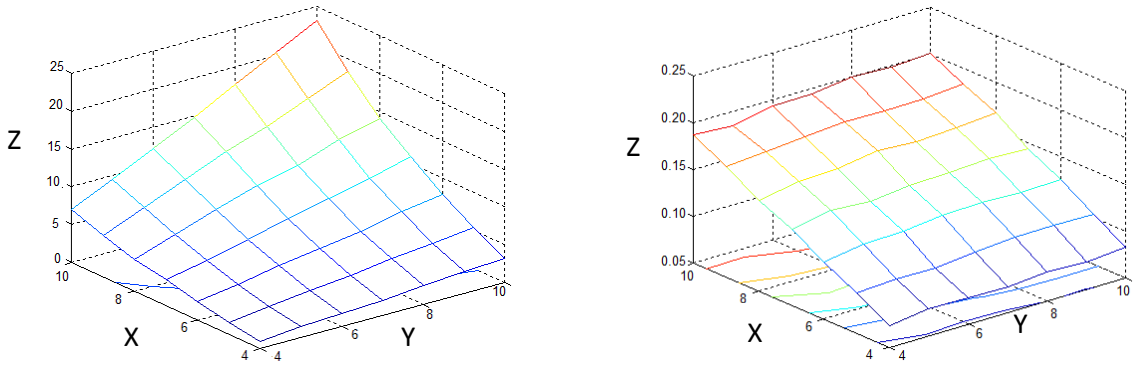

Figure 19: A: Total number of reshuffles *B: Fraction of reshuffles*

The outcomes in graphs 19 A and B were gained by 1000 random trials at a filling rate of 40 percent. The x-axis gives the number of tiers, the y-axis the number of stacks. On the z-axis in graph 1 the number of reshuffles is shown while graph 2 gives the fraction of reshuffles.

For a given filling rate we see that the number of reshuffles increases with both the number of tiers and the number of stacks. If storage space increases, when the filling rate is given, so will the number of containers. It thus makes sense that when retrieving more containers, the number of moves raise, and so will the number of unproductive moves.
What we can see is that the number of reshuffles does increase more with the number of tiers than with the number of stacks.
When we look at the fraction of reshuffle moves (given by the number of reshuffle moves divided by the number of total moves) the number of stacks does not really seem to have much influence on this fraction. The height of the stack given by the number of tiers however does: The more tiers, the higher the percentage of reshuffle moves. This pattern can be found for different filling rates as can be seen in Appendix 1.

7.2 Case 1

7.2.1 Different heuristic variants

To compare the different heuristic variants, bays of 6 tiers and 6 stacks were created with different filling rates. The number of imperfect containers in these initial bays are given in table 2. They give a

lower bound on the number of reshuffles needed to empty the bays. The numbers of reshuffles needed by the different heuristics are shown in table 3. These results were gained by solving for the unrestricted variant with multiple container movements at a time, averaging over 200 bays.

All variants were able to beat thousand random trials at all filling rates. The higher the filling rate the larger the difference between the heuristics and the random trials.

Method A is beaten by C, B, D and E. Since C outperformed method B and D outperformed E, a combination of C and D was also put to the test. C and D loses to a thousand random trials in only 1% of all cases and wins in 25%. In the other cases C and D scores just as good as the best out of a thousand trials.

This combination not only gave the best results in the unrestricted variant but also in the restricted variant for both one, and multiple containers at a time. Tables 8 and 9 in Appendix 2 show their results. Similar results were also found for other bay sizes.

Filling rate	Imperfect containers
Max (86%)	16,82
60%	10,12
40%	5,64

Table 2: Average number of imperfect containers of the 6x6 bays

Computational times of the heuristics differed only slightly from each other and all bays were emptied in well under one second.

Filling rate	Random	Random 1000 x	A	C	B	D	E	C & D combined
Max(86%)	42,66	31,35	26,24	25,29	25,71	26,06	26,12	**25,11**
60%	22,93	15,14	12,63	12,49	12,56	12,59	12,65	**12,44**
40%	9,95	6,37	6,12	6,11	6,11	6,10	6,11	**6,09**

Table 3: The number of reshuffles, unrestricted variant, multiple containers at a time, 6x6 bay.

Conclusion

The heuristic approach beats random trials for all different filling rates. For small filling rates however, the differences are only small.

For a given bay size, the lower the filling rate, the less containers and on average the less reshuffles needed in the optimal solution. These problems require fewer steps to solve and have less solutions

(in the restricted case). When trying to solve with random steps for a large number of times, the probability to find a (near) optimal solution increases.

For higher filling rates the usefulness of the heuristics becomes clearer. Even 1000 random trials only give a relatively small number of solutions to the problem.

We also see that the differences between the heuristics are smaller for low filling rates. They are all close to optimal.

The determination of the best found heuristic seems clear: as there is only a minimal increase in computation time, the combination of heuristics C and D is preferred because it gives the lowest number of reshuffles.

7.2.2 Different solving variants

All four solving variants were tested in different bay settings. Their results can be found in table 4. Table 5 shows the random trial results for the same bays.

The combination of heuristic C and D was used, since this gave the best results for all solving variants in earlier tests. Tests were run for a total of 200 bays per test. The bay sizes and filling rates are given in the tables.

The bays had a minimum filling rate equal to this rate. For example: 4x6 bays with filling rates of 60% had at least 0.6*4*6 = 14.4 containers, which gives a total of 15 containers in each bay.

The outcomes of the table are grouped. Each group of three in table 5 gives the outcomes for 1, 100 and 1000 random trials per bay for rows one two and three resp. Each group of four in table 4 gives the outcomes for the four different variants with:

- r: restricted variant
- u: unrestricted variant
- b: base variant
- e: extended variant

The stars in the table indicate that a limit was set on the number of containers used in the extended variant to decrease computation time.

For the low filling rates all four variants score more or less the same. As the filling rates increase however, we see that the extended variant performs better than the base variant and the unrestricted variant outperforms the restricted variant.

These differences start small for a bay with only 4 tiers and 6 stacks but become somewhat larger for larger bays.

All four variants outperform the best out of thousand random trials, at all bay sizes for the higher filling rates. For the lower filling rates they perform equally good or better.

The heuristics were performed on a Acer Aspire 7720g with 4 GB Intel Core 2 Duo T7500 processor running on Microsoft Vista. When it comes to computing time the unrestricted variant takes a little more time than the restricted one. The time to run the extended version grew rapidly when the number of tiers increased. In most cases the base variant was solved in well under one second. For the extended variant, computation times can go up to a few minutes for the larger bays. The results of table 4 were all gained in a few seconds for each bay by limiting the extended variant.

Filling rate →		Max	80%	60%	40%	20%
r,e	**4x6 Bay (Max= 21)**	12.32	11.04	5.82	2.67	**0.84**
u,e		**12.17**	**10.88**	**5.65**	**2.66**	**0.84**
r,b		12.52	11.17	5.82	2.67	**0.84**
u,b		12.39	11.02	5.73	2.68	**0.84**
r,e	**4x8 Bay (Max = 29)**	17.01	13.22	7.79	**3.80**	**1.04**
u,e		**16.90**	**13.01**	**7.65**	**3.80**	**1.04**
r,b		17.15	13.31	7.81	**3.80**	**1.04**
u,b		17.13	13.26	7.76	**3.80**	**1.04**
r,e	**4x10 Bay (Max = 37)**	21.35	15.17	8.64	**4.29**	**1.17**
u,e		**21.15**	**15.02**	**8.55**	**4.29**	**1.17**
r,b		21.63	15.20	8.64	**4.29**	**1.17**
u,b		21.66	15.23	8.63	4.30	**1.17**
r,e	**6x6 Bay (Max = 31)**	27.02	23.13	12.98	6.21	**1.77**
u,e		**26.96**	**22.73**	**12.76**	**6.11**	**1.77**
r,b		27.79	23.68	13.09	6.22	**1.77**
u,b		27.57	23.35	12.96	6.17	**1.77**
r,e	**6x8 Bay (Max = 43)**	34.90	28.83	15.81	8.22*	**2.30***
u,e		**34.62**	**28.65**	**15.37**	**8.16***	**2.30***
r,b		36.36	29.49	15.95	8.24	**2.30**
u,b		35.57	29.45	15.62	8.28	**2.30**
r,e	**6x10 Bay (Max = 55)**	44.89**	32.31**	19.00**	9.04**	**2.85****
u,e		**44.64****	**32.20****	**18.74****	**9.03****	**2.85****
r,b		45.66	32.51	19.03	9.05	**2.85**
u,b		45.32	32.37	18.98	9.17	**2.85**
r,e	**10x10 Bay (Max = 91)**	118.38**	86.87**	**48.33****	23.45**	**6.66****
u,e		**116.78 ****	**86.67****	48.40**	**23.01****	6.68**
r,b		120.41	88.67	48.86	23.53	**6.66**
u,b		118.45	88.07	48.96	23.27	6.71

Table 4: Heuristics results: Average amount of reshuffles for given bay sizes and filling-rates.

** number of containers at a time set to a maximum of 4*

*** number of containers at a time set to a maximum of 2*

Filling rate →	Max	80%	60%	40%	20%
4x6 Bay	17.86	15.84	9.05	3.79	0.92
	13.52	11.96	6.13	2.67	**0.84**
	12.73	11.29	5.93	2.67	**0.84**
4x8 Bay	25.12	21.23	12.71	5.70	1.31
	19.83	15.55	8.78	3.88	**1.04**
	18.61	14.46	8.15	3.82	**1.04**
4x10 Bay	32.63	25.44	14.54	6.47	1.39
	26.00	19.18	9.97	4.43	**1.17**
	24.39	17.57	9.16	4.31	**1.17**
6x6 Bay	43.05	37.63	22.10	10.15	2.48
	33.55	28.88	16.24	6.78	**1.77**
	31.20	26.90	14.93	6.35	**1.77**
6x8 Bay	59.90	52.09	28.61	13.80	3.17
	47.58	40.62	21.41	9.64	**2.30**
	44.45	37.97	19.78	8.85	**2.30**
6x10 Bay	75.95	61.13	35.87	15.81	4.18
	62.69	49.40	27.42	11.10	2.86
	58.96	45.78	25.28	9.99	**2.85**
10x10 Bay	231.32	180.70	105.97	47.75	10.96
	194.90	151.45	86.04	36.62	7.59
	185.41	143.83	81.47	34.35	6.87

Table 5: Random trial result: Average amount of reshuffles for given bay sizes and filling-rates, for 1, 100 and 1000 trials per bay.

Adaption of the extended variants

Because computation time might increase rapidly for the extended variant, the number of containers for which all possible moves are examined can be limited. The extended variant was tested using different numbers to see if this influenced the solutions. Table 6 gives the number of reshuffles for needed to empty a maximal filled bay of 10 tiers and 10 stacks for both extended variants.

# containers →	1	2	3	4
r,e	120.58	118.38	116.10	115.91
u,e	119.70	116.78	115.52	**115.13**

Table 6: Number of reshuffles needed to retrieve 91 containers in a 10 by 10 bay

Computational times increase largely for more than four containers, so we only have results for a maximum of four containers. When we look at the results, the higher the number of containers, the lower the number of reshuffles. For a 10 by 10 bay this difference is 4 to 5 moves on average.

Conclusion

The extended unrestricted variant gives the best results. One should keep in mind that for large bays computation can become a problem in the extended case.

A solution is that we can set the number of containers for which all possible moves are checked to a limited number. This was already done in some of the test cases and still gave better results than the base variant. This way the biggest part of the influence of the number of tiers on the computational time can be blocked out.

It seems worth the extra effort of allowing unrestricted moves when emptying the bay. The unrestricted variants thus have preference.

The fill rates and sizes of the bays do influence the differences, but overall the same method flourishes.

7.3 Case 2

For the long term results around 5000 containers were added and retrieved in each testing bay. Bay sizes varied as well as the filling rates. The filling rates were determined by the average number of containers in the bay.

In the long run it became clear that the unrestricted extended variant outperformed the restricted and base variants clearly on all filling rates.

For a 6x6 bay with different filling rates the outcomes of the unrestricted base variant can be found in table 7. This table gives the percentage of reshuffles by dividing them by the total number of moves. The computation time for this simulations lies around 14 seconds for the 40% filling rate but get higher for higher filling rates (around 60 seconds at 60%). At a filling rate of 86% method A still kept the percentage of reshuffles under 40% but the other methods could not improve A anymore.

Filling rate	A	C	B	D	E	C & D combined
60%	39,7%	38,3%	37,3%	37,7%	38,0%	37,1%
42%	15,0%	15,2%	14,7%	14,7%	15,3%	14,6%

Table 7: Unrestricted, one at a time

Conclusion

Although sometimes the simulations take a bit more time, this should not have to be a problem in practice because you don't want to decide where to put the upcoming 5000 containers in a few seconds.

The differences between the restricted and the unrestricted variant get huge, so one would want to use the unrestricted.

Strategy B now gives better results than strategy C, this is the opposite of what happened in case 1. This might be due to the fact that in case 2 stacks don't turn empty. The combination of strategy C and D still performs best on different filling rates.

On very high filling rates strategy A cannot be outperformed. This might be due to the fact that on these high filling rates there are only a few sub solutions with the least number of imperfect containers to choose from.

8. Overall conclusion and notes

The unrestricted variant was not only helpful in the second case but also in the first. It gave a clear improvement so that the benefits weigh up to the increase in computation time.

The addition of moving multiple containers in the extended variant also has a positive effect on the results but one should keep in mind that the computation time increases quickly with the number of containers. Using this variant with a maximum on the number of containers might be a solution that ways the benefits against the costs.

One should note that all results depend on filling rates and in case 2 also on the distribution of the arriving containers, which in this case had a randomly chosen storage time on entering.

In case 1 the results were not yet compared to the optimal solutions. This could be done to give a clearer picture of the optimality of the solutions. Especially for high filling rates the solutions deviate largely from the lower bounds. It is important to keep in mind that for these “larger” problems, the optimal solutions will probably also deviate more from these lower bounds.

Overall it can be concluded that the helpfulness of stacking strategies increases with the filling rate and size of the bay. The best found stacking strategy was found by keeping the number of imperfect containers as small as possible, preferably on top of late departure containers and by keeping the departure times of the earliest leaving containers on each stack as high as possible.

9. Appendix

Appendix 1

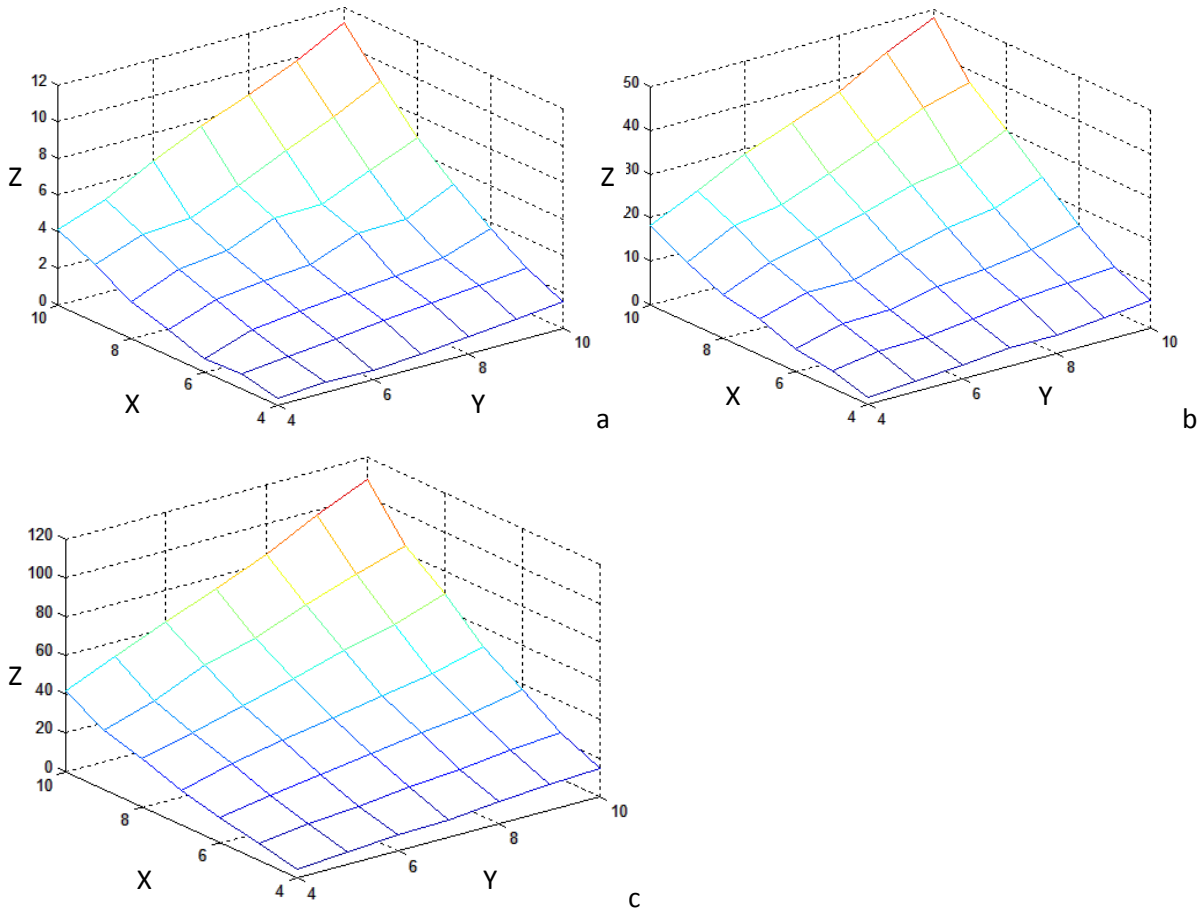

Figure 20: The average amount of reshuffles needed to empty the bay in a random way. X equals the amount of tiers, Y the amount of blocks and Z the amount of reshuffles on a 20, 40 and 60 percent filling rate.

The average amount of reshuffles needed to empty a bay using random trials on the 20%, 40% and 60% filling level for figure A, B and C respectively. Figure A shows a less smooth graph. This can be explained by the filling rate. Because of the amount of containers having to be a natural number, the filling rate is higher or equal to 20% and on some sizes it is higher than on others.

Appendix 1

Filling rate	Random	Random 1000 x	A	C	B	D	E	C & D combined
Max (86%)	42,66	31,35	30,87	27,90	29,68	29,00	29,94	**27,82**
60%	22,93	15,14	14,430	13,400	14,080	13,800	14,01	**13,12**
40%	9,95	6,37	6,42	6,28	6,35	6,31	6,42	**6,25**

Table 8: The number of reshuffles, restricted variant, multiple containers at a time

Filling rate	Random	Random 1000 x	A	C	B	D	E	C & D combined
Max (86%)	42,66	31,35	33,26	29,33	32,23	30,54	32,05	**28,23**
60%	22,93	15,14	15,27	13,81	14,89	14,28	14,93	**13,62**
40%	9,95	6,37	6,64	6,41	6,51	6,38	6,68	**6,27**

Table 9: The number of reshuffles: restricted variant, one container at a time

10. References

Kefi, K. G. (2010). Heuristic-based model for container stacking problem. *19th International Conference on Production Research.*

Kim, K. H. (1997). Evaluation of the number of rehandles in container yards. *Comput. Ind. Eng.*, 32(4) 701-711.

Kim, K. H.-P. (2006). A heuristic rule for relocating blocks. *Comput. Oper. Res.*, 33(4) 940-954.

Kim, K. H.-R. (2000). Deriving decision rules to locate export containers in container yards. *European Journal of operational Research* , 124(i) 89-101.

Wu, T. (2009/2010). A Beam Search Algorithm For Minimizing Reshuffle Operations At Container Yards.

Zhang, G. Z. (2010). An Investigation of IDA* Algorithms for the Container Relocation Problem. *IEA/AIE, Part I, Springer, LNAI 6096*, 31-40.

Caserta M, Voß S, Sniedovich M. (2009). Applying the corridor method to a blocks relocation problem. *OR Spectrum,* doi:10.1007/s00291-009-0176-5.

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

Borgman B, van Asperen E, Dekker R (2010). Online rules for container stacking. *OR Spectrum* 32 (3): 687–716

Caserta M, Schwarze S, Vos S (2010). Container Rehandling at Maritime Container Terminals. *Operations Research/Computer Science Interfaces Series*, 2011, Volume 49, Part 4, 247-269, DOI: 10.1007/978-1-4419-8408-1_13