Container Relocation Problem:
Approximation, Asymptotic, and Incomplete Information

Setareh Borjian ∗  Virgile Galle †  Vahideh H. Manshadi ‡  Cynthia Barnhart §  Patrick Jaillet ¶

Abstract

The Container Relocation Problem (CRP) is concerned with finding a sequence of moves of containers that minimizes the number of relocations needed to retrieve all containers respecting a given order of retrieval. While the problem is known to be NP-hard, there is much evidence that certain algorithms (such as the A* search [15]) and heuristics perform reasonably well on many instances of the problem. In this paper, we first focus on the A* search algorithm, and analyze lower and upper bounds that are easy to compute and can be used to prune nodes and to determine the gap between the solution found and the optimum. Our analysis sheds light on which bounds result in fast computation within a given approximation gap. We also present extensive simulation results that improve upon our theoretical analysis, and further show that our method finds the optimum solution on most instances of medium-size bays. On the “hard” instances, our method finds an approximate solution with a small gap and within a time frame that is fast for practical applications. We also study the average-case asymptotic behavior of the CRP where the number of columns grows. We calculate the expected number of relocations in the limit, and show that the optimum number of relocations converges to a simple and intuitive lower-bound. This gives strong evidence that the CRP is “easier” in large instances, and heuristics such as [1] finds near optimal solution.

We further study the CRP with incomplete information by relaxing the assumption that the order of retrieval of all containers are initially known. This assumption is particularly unrealistic in ports without an appointment system. In our model, we assume that the retrieval order of a subset of containers is known initially and the retrieval order of the remaining containers is observed later at a given revelation time. Before this time, we assume a probabilistic distribution on the retrieval order of unknown containers. We extend the A* algorithm and combine it with sampling technique to solve this two-stage stochastic optimization problem. We show that our algorithm is fast and the error due to sampling is reasonably small. Using our framework, we study the value of information and the effect of revelation time on the average number of relocations.

∗ORC and CEE, MIT. Email: sborjian@mit.edu
†ORC, MIT. Email: vgalle@mit.edu
‡School of Management, Yale. Email: vahideh.manshadi@yale.edu
§CEE, MIT. Email: cbarnhar@mit.edu
¶EECS and ORC, MIT. Email: jaillet@mit.edu
1 Introduction

With the growth in international container shipping in maritime ports, there has been an increasing interest in improving the operations efficiency in container terminals on the sea side and land side. The operations in the sea side include loading export containers on the vessels or discharging import containers from vessels and loading them onto internal trucks. The import containers are then transferred to the land side and are stacked in the storage area. These containers are later retrieved and delivered to external trucks to be distributed in the city.

Due to limited space in the storage area, containers are stacked in tiers on top of each other. As shown in Figure 1, several columns of containers in a row create a bay of containers. If a container that needs to be retrieved (target container) is not located at a top most tier and is covered by other containers, the blocking containers must be relocated to another slot. As a result, during the retrieval process, one or more relocation moves are performed by the yard cranes. Such relocations are costly for the port operators and result in delay in the retrieval process. Thus, reducing the number of relocations is one of the main challenges faced by the port operators. Finding the sequence of moves that minimizes the number of relocations while retrieving containers from a bay in a pre-defined order is referred to as the Container Relocation Problem (CRP) or the Block Relocation Problem (BRP). A common assumption of the CRP is that only the containers that are blocking the target container can be relocated. We refer to the CRP with this setting as the restricted CRP.

The CRP has been widely studied in the literature and has been shown to be NP-hard [1]. A few papers have developed mathematical models for the CRP and have solved small-sized instances of the problem exactly ([1], [11], [5]). In other papers, different heuristics are designed and tested on real-sized instances ([6], [2], [7], [3]). Many of these heuristics are fast and shown to be on average close to optimal for small-sized instances. However, none of them have provable guarantees and their performance on large-sized instances cannot be well assessed. Among the optimization methods that provide exact solution for the CRP, the $A^*$ algorithm (a branch and bound framework) has shown to be very effective and able to solve large instances of the problem in a time frame that
is suitable for practical application [13], [17], [13].

In this paper, we apply the $A^*$ algorithm to the CRP and extensively study its properties including different lower and upper bounds, both theoretically and through simulation. We provide theoretical optimality gaps as a function of the depth of the tree in the $A^*$ algorithm. Further, we show through experiments that the optimality gap drops to zero very quickly and the optimal solution is obtained on average after constructing only a few levels of the tree (retrieving a few containers). We also use the idea of branching from the $A^*$ algorithm to design a new heuristic called the Tree Heuristic ($TH - L$). We use randomly generated instances to benchmark several existing heuristics and the Tree Heuristic against the optimal solution given by the $A^*$ algorithm and show that $TH - L$ has the best performance (smallest optimality gap) on average.

The CRP has mostly been studied in a two dimensional setting (relocating containers within one bay), and for small bay sizes. However in reality, containers are grouped together in a block that consists of several bays, and a container can be relocated within a bay or from one bay to another. To model the relocations among bays, one may consider several bays in a block as a single large bay (with many columns), and solve the problem for this large bay. In this paper, we investigate the CRP for large-sized instances and we show how the number of relocations scale as the number of columns grows. We also show through experiments the effectiveness of the Tree Heuristic as well as the one presented in [1].

A critical assumption of the CRP is that the retrieval order of all containers in the bay are considered to be known in advance. However, in practice, such information is not available far in advance and the exact departure order of containers becomes known only as their retrieval times approach. We refer to this problem as the CRP with incomplete information and we study it in a 2-stage setting, where the retrieval order of a subset of containers is known initially and the retrieval order of the remaining containers is revealed later but all at once. Before the entire information becomes available, we assume a probabilistic distribution on the retrieval order of unknown containers. We develop a 2-stage approximate stochastic optimization algorithm, called $ASA^*$, and use it to solve medium-sized instances of the CRP with incomplete information. We also give theoretical bounds on the approximation error. Furthermore, we introduce a myopic heuristic which is fast and performs well on average compared to the $ASA^*$ algorithm. We use the heuristic to solve large instances of the problem and show through experiments that the number of relocations for the 2-stage setting converges to a constant as the bay gets larger. Moreover, we use the $ASA^*$ algorithm and the myopic heuristic to assess the value of information. Our experiments show that when the retrieval order of 50%-75% of containers is known at the beginning, the loss from the missing information is negligible and the average number of relocations is very close to that of the CRP with full information.
1.1 Literature Review

For a general survey on port operations literature, see Stahlbock and Voss [12]. For a recent general review and classification survey of the existing literature on the CRP as well as the stacking problem, see Lehnfeld and Knust [8].

Many heuristics have been developed for the CRP. Kim and Hong suggest a decision rule that uses an estimate of expected number of additional relocations for a column [6]. Caserta et al. [2] propose a binary description of configurations and use it in a four steps heuristic. Lee and Lee [7] consider an extended objective incorporating working time of cranes in a three phases heuristic. Caserta et al. [3] derive a meta-heuristic based on the corridor method, which again optimizes over a restricted set of moves and Caserta et al. in [1] present a heuristic that is defined later in this paper. Ünlüyurt and Aydın [14] propose a branch and bound approach with several heuristics based on this idea. Foster and Bortfeld [4] use a tree search procedure to improve a greedy initial solution. Finally, Petering and Hussein [11] describe a look-ahead heuristic that does not make the common assumption of the restricted CRP.

In the mathematical programming formulations that insure optimality on the entire feasible set of moves, the first binary linear formulation was suggested by Kim and Hong [6]. Petering and Hussein [11] propose another formulation which is more tractable but cannot solve real-sized instances efficiently. Caserta et al. [1] derive a different Integer Program (IP). However they notice that the number of variables and constraints dramatically increases as the size of the bay grows and the IP cannot be solved. A way to bypass this problem has been to look at a new type of algorithm called $A^*$. It was first introduced to this problem by Zhang et al. [15], studied by Zhu et al. [17], and Tanaka and Takii [13] proposed a new lower bound for the algorithm.

Olsen and Gross [10] also provide an average case analysis of the asymptotic CRP when not only the number of columns grows to infinity but also the maximum height of each tier. They show that there exists a polynomial time algorithm that solves this problem close to optimality with high probability.

Finally, we mention that another major challenge in port optimization is uncertainty and the value of information, which have not been considered in the literature until very recently. Zhao and Goodchild [16] model uncertainty in truck arrivals at the terminal. They assume that trucks arrive in groups and that the information is revealed for the whole group at the same time. Using this assumption of rolling information, they use simulation to get insights on the trade-off between information and optimality.

This issue of lack of information is critical as it captures port operations more realistically. Understanding value of information helps port operators design efficient appointment system for trucks.

This paper is structured as follows: Section 2 presents the $A^*$ Algorithm in the full information
case. It describes the method, proposes lower and upper bounds and derives theoretical results. Section 3 presents the asymptotic analysis of the CRP. Section 4 provides a stochastic scheme for the case of incomplete information and studies the value of information about container departure times. In Section 5, we present several experimental results to make an average case analysis of the CRP in both the complete and incomplete information cases. Finally, Section 6 concludes and suggests future research topics. The proofs for all propositions are given in the Appendix.

2 Optimal solution for the CRP with full information

In this section and for the sake of completeness of the paper, we present the $A^*$ algorithm applied to the CRP, previously introduce by [17]. We also introduce bounds developed in the literature ([1] and [17]) and we derive new theoretical results on them.

2.1 Description of the algorithm

To find the optimal solution, we use the $A^*$ algorithm, which is basically a move-based approach using a special decision tree.

This approach takes 4 entries: $B$ is the bay to be solved optimally. $R(D)$ and $S(D)$ are functions giving respectively upper and lower bounds on $z_{opt}(D)$ for any configuration $D$. In this paper, upper bound functions that are considered also give us feasible solutions. Finally, $N$ is the maximum number of nodes in the decision tree.

The algorithm returns two results: The best incumbent (in objective function) found by $A^*$, denoted $z_A$, to which we associate the sequence of moves needed to find this incumbent called $\sigma_A$ and the gap guaranteed between $z_A$ and $z_{opt}(B)$ denoted $\gamma$. Notice that if the algorithm did not reach $N$ nodes, then the algorithm found the optimal solution (i.e. $\gamma = 0$ and $z_A = z_{opt}(B)$).

Let us define the following notations: $l$ is the level of the tree. A node is at level $l$, if $l$ relocations have been done from the initial bay to get to this node. This is the reason why this method is called a move-based approach. The algorithm is described in Algorithm [1]

The problem is defined as a sequence of decisions. Suppose $n$ is the target container. If $n$ is not blocked, it can be retrieved since a retrieval is optimal. In that case, no decision has to be made, therefore no new node is created. If $n$ is blocked, the topmost blocking container needs to be relocated to another column. This column has to be chosen in order to minimize all future relocations. In order to do so, we generate all possible feasible moves for the blocking container by creating children of a node in a decision tree procedure. Considering all possible nodes gives the optimal solution.

Building the whole tree of feasible moves would not be tractable since the number of nodes increases exponentially with the number of relocations. Consequently, we need some kind of decision rule to prune the tree with the assurance of keeping the optimal path in the tree.
Algorithm 1 $A^*$ Algorithm

1: procedure $[z_A, \gamma] = A^*(B, R, S, N)$
2: Pre-processing:
3: while target container $n$ is on top do
4: retrieve target container $n$ of $B$ and $n \leftarrow n + 1$
5: end while
6: Initialize:
7: $z_A \leftarrow \infty$, $l \leftarrow 0$, $B^l \leftarrow B$, $m \leftarrow 0$ (number of nodes)
8: Loop:
9: while all nodes at level $l$ are not pruned do
10: for all nodes $B^l$ at level $l$ do
11: $U(B^l) = R(B^l) + l$ and $L(B^l) = S(B^l) + l$
12: Updating the incumbent:
13: if $U(B^l) < z_A$ then
14: $z_A \leftarrow U(B^l)$
15: end if
16: Pruning:
17: if $U(B^l) > L(B^l)$ and $L(B^l) < z_A$ then
18: for Every “Child” of $B^l$ do
19: if $m \geq N$ then
20: Stop
21: else
22: add “Child” to the tree at level $l + 1$ and $m = m + 1$
23: end if
24: end for
25: end if
26: end for
27: $l \leftarrow l + 1$
28: end while
29: Gap:
30: $L_{\min} = \min_{D \text{ leaf not pruned}} (L(D))$
31: $\gamma \leftarrow z_A - L_{\min}$
32: end procedure

Let $B^l$ be any configuration that appears at level $l$ in the tree. By definition,

\[ S(B^l) \leq z_{opt}(B^l) \leq R(B^l) \]  \hspace{1cm} (1)

Let us define the cumulative upper and lower bounds as the sum of the bound evaluated at the
current state and the number of relocations already performed (i.e., \( l \) relocations have been done at level \( l \) of the tree) \( U(B^l) = R(B^l) + l \) and \( L(B^l) = S(B^l) + l \) (line 10 of Algorithm \( \mathbb{I} \)). Notice that in that case

\[
L(B^l) \leq z_{opt}(B) \leq U(B^l), \text{ for all } B^l \text{ in the } A^* \text{ tree}
\]  

(2)

At any level \( l \), the algorithm prunes paths using the following two rules (line 15 of Algorithm \( \mathbb{I} \)):

(i) If \( L(B^l) = U(B^l) \), then \( z_{opt}(B^l) = U(B^l) \) and we can stop branching on this path. From this point, we can simply follow the feasible solution given by the upper bound.

(ii) If \( L(B^l) \geq z_A \) (i.e. the lower bound \( L(B^l) \) is larger than the best solution found so far \( z_A \)) then this node can be pruned because the optimal solution for this node is going to be greater than its lower bound, hence greater than \( z_A \). The best incumbent is updated if a \( B^l \) such that \( U(B^l) < z_A \) is found (lines 12, 13 of Algorithm \( \mathbb{I} \)).

By construction, if the algorithm terminates, those two rules maintain the optimal path in the \( A^* \) tree.

![Figure 2: Bay with 3 tiers, 3 columns and 6 containers](image)

**Example 1** Figure \( \mathbb{3} \) presents an example of \( A^* \) where the initial bay is the one in Figure \( \mathbb{2} \). It uses upper and lower bounds that are presented in Section \( \mathbb{2.2} \). Based on this example, we focus on the 4 rightmost configurations in Figure \( \mathbb{3} \). The third one uses rule (i) to stop branching from this node since lower and upper bounds are both equal to 4 at this node. At this level, we can notice that \( z_A = 4 \). Therefore we can also prune the first, second and fourth nodes using rule (ii). Therefore, the tree is complete at level 2. We follow the path until the third node at level 2 and then follow the feasible solution given by the upper bound from this point.

We know from [1] that the restricted CRP is NP hard. Therefore, there exists some instances for which the number of nodes in the tree is too large to find the optimal solution with this method. Using the fact that upper bounds provide feasible solutions, we can set a maximum number of nodes \( N \) allowed in the \( A^* \) tree and return our best incumbent if the number of nodes reaches \( N \). In this case, we cannot insure optimality, but we can provide a guaranteed gap with optimality. Let \( L_{min} \) be the minimum of the cumulative lower bounds of every leaf of the tree that is not pruned by \( A^* \). Notice that \( z_{opt}(B) \geq L_{min} \). Thus \( z_A - z_{opt}(B) \leq z_A - L_{min} = \gamma \) (line 24 of Algorithm \( \mathbb{I} \)).
We show later that given certain properties on the lower bounds, as we increase $N$, then $\gamma$ is not increasing. In any case, there exists $N^*$ such that if $N > N^*$ then $\gamma = 0$. In Section 5, we show in an average case analysis how $N$ and $\gamma$ are related.

The size of the $A^*$ tree varies according to the definition of the local lower and upper bounds that are used. The closer they are to the optimal solution at each node, the faster the tree is pruned, hence the faster it solves. In this paper, we define some of the possible candidates, explain this choice and provide details about the efficiency of the chosen bounds. Note that the algorithm can also be implemented using depth first search instead of breadth first search; in this paper, we implement the latter as it appears to be the most efficient one.

2.2 The Candidate Bounds

2.2.1 The Lower Bounds

The counting lower bound. This bound was introduced by Kim and Hong in [6] and it is based on the following simple observation. If a container is blocking, then it must be relocated at least once. Thus we count the number of blocking containers in $B$ and we denote it $S_0(B)$. Note that if a container blocks more than one container (for instance, container 6 blocks containers 1 and 4 in Figure 2), we count this container only once.
Fact 2.1. For any configuration $B$, 
\[ S_0(B) \leq z_{opt}(B) \]

We define $L_0$ as the cumulative bound computed with $S_0$.

As mentioned in section 2.1, we want our algorithm to have the following property: As we increase $N$, $\gamma$ should not increase. One way to enforce this is to insure that given any path of the tree, the cumulative lower bound is not decreasing on this path. We will indeed show that $L_0$ follows this property. Before proceeding let us define a few notations: Let $c_i$ be the $i^{th}$ column of the configuration and $\min(c_i)$ be the index of the container with the smallest retrieval time in $c_i$. Now suppose we want to move the blocking container $r$. If we relocate $r$ to column $c_i$ and $r < \min(c_i)$, then we call such a move a “good” move and column $c_i$ is a “good” column for $r$. On the other hand, if $r > \min(c_i)$, we call this move a “bad” move and $c_i$ a “bad” column for $r$.

Proposition 2.2. For any configuration $B$, any level $l \geq 0$, any configuration $B^l$ in level $l$ of $A^*$ and any child $B^{l+1}$ of $B^l$, we have $L_0(B^{l+1}) \geq L_0(B^l)$.

The look-ahead lower bounds. Note that the counting lower bound ($S_0$) is only taking into account the initial configuration of the bay. By definition, a “bad” move implies a future relocation for the relocated container. Therefore we can construct lower bounds that anticipate “bad” moves and hence are closer to the optimal solution. This idea has been used by Zhu et al. (17) to define the following family of lower bounds. For the sake of completeness, we redefine the bounds formally. In this article we prove in Proposition 2.3 that these lower bounds have the non-decreasing property as we branch in the $A^*$ tree.

The basic idea is the following: We want to count unavoidable “bad” moves in any feasible sequence of moves. In order to do so, we need to keep track of the “maximum of minimums” of each column of $B$ denoted by $\text{MOM}(B) = \max_{i \in \{1, \ldots, C\}} (\min(c_i))$. Suppose $r$ is the first blocking container; a “bad” move for $r$ is unavoidable if $\text{MOM}(B) \leq r$. As we empty the bay, the MOM is changing. More precisely the MOM depends on the sequence of moves that we use. We can, however, compute an upper bound on the MOM for every feasible sequence of moves. One way is to assume that every container that is relocated is discarded from the bay. At any time, the MOM in the “discarded” bay will be not smaller than if we computed it given any feasible sequence of moves. Before introducing formal notations, we give an example.

Example 2 Consider the leftmost bay $B$ in Figure 4. First, one can see that $S_0(B) = 2$ since 6 is blocking 1, and 5 is blocking 2. Now in $B_1$, 6 is blocking 1 and $\text{MOM}(B_1)$ is 3; thus 6 has to be relocated at least twice. Therefore the optimal is at least 1 more than $S_0$. Now consider bay $B_2$ where 1 and 6 have been discarded. 5 is blocking 2 and $\text{MOM}(B_2)$ is 4; thus 5 has to be relocated.
Figure 4: Bay with 3 tiers, 3 columns and 6 containers

at least twice. Therefore the optimal is at least 2 more than \( S_0 \).

Now we formalize this idea: Let \( n_1 \) be the smallest container in the bay (for simplicity of proofs we assume that \( n_1 = 1 \)). Let \( k \) be a container \((k \in \{n_1, \ldots, N\})\) and let \( R_k(B) \) be the set of containers blocking \( k \) and not blocking any container \( k' \) such that \( k' < k \).

Let \( B_n = B \) and let \( B_{n+1} \) be the bay where container \( n_1 \) and containers in \( R_{n_1}(B) \) have all been discarded from \( B_n \). By recursion, we can define a sequence of bays \( B_k \) for \( k \in \{n_1 + 1, \ldots, N\} \).

For \( n \in \mathbb{N} \), we now define the \( n^{th} \) look-ahead lower bound (denoted by \( S_n(B) \)):

\[
S_n(B) = S_0(B) + \min_{n+1} \sum_{k=n_1}^{n} \sum_{r \in R_k(B)} \chi(r > \text{MOM}(B_k)),
\]

where \( \chi \) is the indicator function. The corresponding cumulative lower bound is denoted by \( L_n(B) \).

**Fact 2.3.** For every configuration \( B \) and \( n \in 1, \ldots, N \),

\[
S_n(B) \leq z_{opt}(B)
\]

**Proposition 2.4.** For \( n \in 1, \ldots, N \) and any bay \( B \), we have \( L_n(B) \geq L_{n-1}(B) \).

This first result shows that as we increase \( n \), \( S_n \) gets closer to the optimal solution. In Section 5, we numerically study the effect of the choice of lower bounds in the \( A^* \) algorithm in terms of the number of nodes needed to solve a random set of instances. We observe that on average, most of the gain comes from the first look ahead. Another observation is that \( S_N \) is really close to the optimal solution.

We remark that \( S_N(B) = S_{N-C}(B) \) simply because when we reach \( N-C \), there is always at least one empty column. Note that if \( k \) is the smallest integer such that \( B_k \) has an empty column, then the lower bound does not change, i.e., \( S_N(B) = S_{k-1}(B) \). Thus, in practice, the process of computing the lower bounds terminates much before \( N - C \).

In the next proposition, we show that similar to \( L_0 \), the cumulative upper bound \( L_n \) has the desirable property, i.e., they do not decrease as we branch in the \( A^* \) tree.

**Proposition 2.5.** For any \( n \in 1, \ldots, N \), configuration \( B \), level \( l \geq 0 \), configuration \( B^l \) in level \( l \) and any child \( B^{l+1} \) of \( B^l \) in the \( A^* \) tree, we have \( L_n(B^{l+1}) \geq L_n(B^l) \).

In general it is hard to find the rate at which the lower bound increases on a given path. In Section 5 we show how the lower bound \( S_N \) increases on the optimal path of random instances.
2.2.2 The Upper Bound

Any feasible solution can be used as an upper bound. Thus, we use heuristics to construct feasible solutions that serve as upper bounds.

Ideally we would want a heuristic that is close to the optimal solution and at the same time easy to compute. Note that we need to compute the upper bound at every node. The heuristic proposed by Casserta et al. ([1]) seems to meet those criteria. For completeness of the paper, we redefine this heuristic. In this article we prove several properties that we will use in later parts (see Propositions 2.7, 2.8 and 2.9).

The Heuristic H ([1]) Suppose \( n \) is the target container located in column \( c \), and \( r \) is the topmost blocking container in \( c \). We use the following rule to determine \( c^* \), the column where \( r \) should be relocated to. Recall that \( \min(c_i) \) is the minimum of column \( c_i \). We set \( \min(c_i) = N + 1 \) if \( c_i \) is empty. We have:

\[
 c^* = \begin{cases} 
 \argmin \{\min(c_i) : \min(c_i) > r\} & \text{if } \exists \ c_i \text{ such that } \min(c_i) > r \\
 \argmax \{\min(c_i)\} & \text{otherwise}
\end{cases}
\]

The rule says: if there is a column where \( \min(c_i) \) is greater than \( r \) (\( r \) can do a “good” move), then choose such a column where \( \min(c_i) \) is minimized, since columns with larger minimums can be useful for larger blocking containers. If there is no column satisfying \( \min(c_i) > r \) (\( r \) can only do “bad” moves), then choose the column where \( \min(c_i) \) is maximized in order to delay the next unavoidable relocation of \( r \) as much as possible. We will refer to this heuristic as heuristic H and denote its number of relocations by \( z_H(B) \).

Example 2 Consider the bay \( B \) in Figure 2. Using heuristic H, the sequence is going to be: relocation of 6 to column \( c_3 \), retrieval of 1, relocation of 5 to column \( c_1 \), retrieval of 2, relocation of 6 to column \( c_2 \), retrieval of 3, relocation of 5 to column \( c_2 \), retrievals of 4,5 and 6. Thus \( z_H(B) = 4 \). Notice that we had \( S_2(B) = 4 \), so heuristic H is optimal for this initial configuration.

By definition, H is a feasible solution, so it is an upper bound on the optimal solution.

Fact 2.6. For any configuration \( B \), we have \( z_{opt}(B) \leq z_H(B) \).

The heuristic H has a certain number of useful properties stated below.

Proposition 2.7. In a bay with \( C \) columns, for any configuration \( B \) with at most \( C \) containers, we have

\[
 S_0(B) = z_{opt}(B) = z_H(B).
\]
Proposition 2.8. In a bay with $C$ columns, for any configuration $B$ with at most $C+1$ containers, we have $S_1(B) = z_{\text{opt}}(B) = z_H(B)$.

This result implies that if we use heuristic $H$ as an upper bound and the $S_1$ lower bound in the $A^*$ approach, the tree will stop at most after $N-C-1$ retrievals with the guarantee of an optimal solution.

We provide two similar bounding results on $H$.

Proposition 2.9. In a bay with $C$ columns, for any configuration $B$ with at most:

- $C+2$ containers, we have $z_H(B) \leq z_{\text{opt}} + 2$
- $C+k$ containers where $k \geq 3$, we have $z_H(B) \leq z_{\text{opt}}(B) + \frac{k(k+1)}{2}$

Remark 2.10. The following example shows that this upper bound can increase as we branch in the $A^*$ tree. in Figure 5, bay (a) shows the initial configuration and bay (b) is one of its two “children”. It is easy to check that heuristic $H$ needs 7 relocations for bay (a) and 8 for bay (b).

![Figure 5: Heuristic H can increase on a path of $A^*$ tree](image)

3 An asymptotic analysis of CRP

As motivated in the introduction, operators might want to solve bays with large numbers of columns. In this section, we study CRP for random large bays and we show that as the number of columns in the bay gets larger, the problem gets “easier” in the sense that the gap between the optimal solution and our simplest lower bound ($S_0$), does not increase on average and in fact is bounded by a constant. The basic intuition is that as the number of columns grows, then for any blocking container, there is a high probability that we can find a “good” column which implies that each blocking container is only relocated once with high probability.

Throughout this section, we assume that initial configurations have $P$ tiers, $C$ columns, $N$ containers and that there are exactly $h$ containers in each column, where $h \leq P-1$. We assume that the initial configuration is a uniform sample among all possible such configurations. Notice that in that case, when $C$ grows to infinity, $N = h \times C$ also grows to infinity.
First let us explain how a uniformly random bay is generated. We view a bay as an array of $P \times C$ columns. The slots are numbered from bottom to top, and left to right from 1 to $PC$. For example, let $P = 4$ and $C = 7$, the second topmost slot of the third column is 11. The goal is to get a bay $B$ with uniform probability, meaning each container is equally likely to be anywhere in the configuration, with the restriction that there are $h$ containers per column. We first generate uniformly at random a permutation of $\{1, \ldots, N\}$ called $\pi$. Then we assign a slot for each container with the following relation:

$$B(j, i) = \pi(h \times (i - 1) + j) \text{ for } j \leq h \text{ and } B(j, i) = 0 \text{ for } j \geq h + 1.$$  

One can see that each bay is generated with probability $\frac{1}{N!}$. There is a one to one mapping between configurations with $C$ columns and permutations of $\{1, \ldots, hC\}$, denoted by $S_{hC}$. Finally, we denote the expectation of random variable $X$ over this uniform distribution by $\mathbb{E}_C[X]$.

Now we compute the expected counting lower bound.

**Proposition 3.1.** Let $S_0$ be the counting lower bound (defined in Section 2), we have

$$\mathbb{E}_C [S_0(B_C)] = \alpha_h \times C,$$

where $\alpha_h$ is the expected number of blocking containers in one column and can be computed as

$$\alpha_h = \frac{1}{h} \sum_{k=0}^{h-1} \sum_{j=1}^{k} \frac{1}{h} p_{k-(h-j), j-1},$$

and $p_{k,h} = \mathbb{P}[\text{there are } k \text{ blocking containers in a column with } h \text{ containers}].$

**Remark 3.2.** Note that $\alpha_h$ is independent from the set of numbers from which the $h$ numbers are drawn uniformly.

The major result of this part is the following. In the asymptotic case where the number of columns increases to infinity, the expected optimal number of relocations is “equivalent” to the expected number of blocking containers (the counting lower bound).

**Theorem 3.3.** Let $S_0$ be the counting lower bound (defined in Section 2) and $z_{opt}$ be the optimal number of relocations. Then for $C \geq h + 1$, we have

$$1 \leq \frac{\mathbb{E}_C [z_{opt}(B_C)]}{\mathbb{E}_C [S_0(B_C)]} \leq f(C)$$

where

$$f(C) = 1 + \frac{K}{C} \rightarrow \infty 1$$

where $K$ is a constant defined by equation (18).
The proof of Theorem 3.3 is given in the Appendix. Here we just give an intuition of the proof. Under some event that happens with high probability when $C \to \infty$, $E_C[z_{opt}(BC)] - E_C[z_{opt}(BC)+1]$ is exactly $\alpha_h$. Therefore, when $C \to \infty$, $E_C[z_{opt}(BC)]$ essentially behaves like $\alpha_h \times C$, which is equal to $E_C[S_0(BC)]$ (according to Proposition 3.1).

In the next corollary, we show that the optimal solution of the unrestricted CRP has a similar asymptotic behavior.

**Corollary 3.4.** Let $z_{gen}(BC)$ be the optimal number of relocations for the general CRP. We have for $C \geq h + 1$,

$$1 \leq \frac{E_C[z_{gen}(BC)]}{\alpha_h C} \leq f(C)$$

where $f$ is the function defined in Theorem 3.3.

In this Section, we proved a new theorem which gives insights on how the expected optimal solution of the CRP behaves asymptotically on random bays.

To conclude this section we show experimentally that the same result holds for heuristic $H$, i.e., the ratio of $E[z_H(BC)]$ and $E[S_0(BC)]$ converges to 1 as $C$ goes to infinity. For each size $C$, we compute this ratio and take expectations over a million instances and plot the result in Figure 6a. Notice that we have $1 \leq \frac{E[z_H(BC)]}{E[S_0(BC)]} \leq \frac{E[z_H(BC)]}{E[z_{opt}(BC)]}$, so Figure 6a also shows experimentally Theorem 3.3.

![Figure 6: Simulations for the Asymptotic Theorem](image)

First, note that Figure 6a implies that the relative gap between heuristic $H$ and $S_0$ shrinks to 0 as $C$ increases. Moreover we have $\frac{E[z_H(BC)] - E[z_{opt}(BC)]}{E[z_{opt}(BC)]} \leq \frac{E[z_H(BC)] - E[S_0(BC)]}{E[S_0(BC)]}$ and thus the relative gap of $H$ with optimality also converges to 0 as $C$ grows to infinity.

In the proof of Theorem 3.3, we also study the function $g(C) = E_C[z_{opt}(BC)] - E_C[S_0(BC)]$. Note that $g(C) \leq E_C[z_H(BC)] - E_C[S_0(BC)]$ where the right-hand side of the inequality is the function plotted in Figure 6b. The plot shows that $g(C) \leq 1.25$ for all $C$, meaning that $g(C)$ is
bounded as we proved in Theorem 3.3. Moreover, the plot implies that heuristic H is on average at most 1.25 away from the optimal solution, so heuristic H is relatively more efficient in the case of large bays. Intuitively, the problem should become harder as C increases. However, the probability of having a good column converges to 1, as we increase the number of columns; hence the problem tends to become easier as C grows.

Finally, in the proof, we note that the rate of convergence for the optimal solution to $S_0$ is at least $\frac{1}{C}$. Interestingly, we can infer from Figure 6a that the rate of convergence of the ratio for heuristic H is also proportional to $\frac{1}{C}$.

4 CRP with Incomplete Information

The $A^*$ algorithm as explained in Section 2 relies on the assumption that the departure order of containers is known in advance. However, in practice, we often only know the departure order of some of the containers (those that are going to depart in the near future).

In this section we explain how the $A^*$ algorithm can be adapted in a Dynamic Programming (DP) framework for solving the CRP in the case that some of the decisions should be made with incomplete information. We use the $A^*$ algorithm to obtain an approximate solution for this problem. In this section, we describe the setting of the problem, introduce some notations, and explain the algorithm for the CRP with incomplete information. Moreover, we show how to bound the approximation error. In Section 5 we present the results of computational experiments.

The CRP with incomplete information involves retrieving $N$ containers from a bay with $C$ columns and $P$ tiers, where partial information about the departure order of the containers is initially available. Usually, very little information is available about the containers that are going to be retrieved far in the future. Thus, it is reasonable to assume that at any given time step $t$, we only know the departure order of a certain number of containers in the bay (the containers that are going to be retrieved within a short time horizon after $t$). We refer to such containers in the bay as known containers. Similarly, we refer to the remaining containers in the bay as unknown containers. Note that all unknown containers have larger indices (i.e., later departure times) than known containers.

As time passes, some of the known containers are retrieved and as more information becomes available, some of the unknown containers become known. In general, the information might be updated multiple times during the retrieval process (every time that the arrival time of a truck is provided by the truck driver, the information gets updated and some unknown containers become known). In the most general case, the information can be updated every time step. Alternatively, we can consolidate several small pieces of information into one or a few pieces, and assume that the information is revealed at $\Gamma$ different times during the retrieval process (i.e., $\Gamma$ is the number of time that information is updated). In this case, we have a multi-stage problem and $\Gamma$ sets of decisions need to be made.
In this paper, we focus on a 2-stage setting; we assume that a subset of containers is initially known and that the departure order of all the remaining ones becomes known at time \( t^* \). We denote the set of known containers by \( K \) and assume that containers \( \{1, 2, \ldots, |K|\} \) are those known at time zero. Similarly, the set of unknown containers is denoted by \( U \), and containers \( \{|K|, |K|+1, \ldots, N\} \) become known at time step \( t^* > 0 \) (recall that a time step is the minimum amount of time required for completing one retrieval or one relocation move). We refer to this setting as the 2-stage CRP since there are two types of decisions that need to be made: first-stage decisions (retrievals and relocations before time step \( t^* \)) and second-stage decisions (retrievals and relocations after time step \( t^* \)). We assume that before \( t^* \), we have probabilistic information about the containers in \( U \), meaning that we know the probability of realization of each possible departure order. Such information can be obtained from historical data or from an appointment system that provides some estimate of departure times of the containers (for example a time window for retrieving each container). We denote the set of possible departure orders of the containers in \( U \) (possible scenarios) by \( Q \). From now on, we assume that all scenarios are equally likely, i.e., the probability of each scenario is \( \frac{1}{|Q|} \), and the number of scenarios is \((N - |K|)!\). Note, however, that we could use the algorithm to solve the CRP with any other probability distribution on the departure order of unknown containers.

We use a 2-stage stochastic optimization technique to solve this problem, where in the first-stage we minimize \( E[z] \) as follows:

\[
\min_{\sigma_1, \ldots, \sigma_{t^*-1}} \quad E[z] = \sum_{q \in Q} \frac{1}{|Q|} z(B(q)).
\]

where \( B(q) \) is the resulting bay when scenario \( q \) is realized, \( z(B(q)) \) is the total number of relocations for \( B(q) \), and \( \sigma_1, \ldots, \sigma_{t^*-1} \) are the first-stage decisions. The 2-stage problem can be solved with the A* algorithm as follows:

(1). We build the tree with \( t^* - 1 \) moves for time-steps \( 1, 2, \ldots, t^*-1 \), in a similar way as illustrated in Figure 3; we denote this tree by \( T_{[1,t^*-1]} \).

(2). For each node at level \( t^* - 1 \), we need to compute the expected number of remaining relocations. We enumerate all possible scenarios and solve the CRP with complete information corresponding to each scenario, using the A* algorithm.

(3). We find \( p^* \) (the optimal path or sequence of moves) that minimizes the expected total number of relocations over all paths.

(4). Once \( p^* \) up to time-step \( t^* - 1 \) is selected and we observe the information at \( t^* \), we use the A* algorithm to solve a specific instance through the end.

Notice that to find the optimal path for time interval \([1,t^*-1]\), we need to solve up to \((c - 1)(t^*-1)(N - |K|)!\) instances in Step (2) with the A* algorithm. Although A* is fast, the number of scenarios is prohibitively large and enumerating all scenarios is not feasible due to limited resources of memory and long computation time. We next explain how we use sampling and pruning to
overcome these issues. We also quantify the error incurred as a result of sampling and pruning. We refer to the resulting algorithm as ASA∗ (Approximate Stochastic A∗).

**Limiting the number of possible scenarios on each path.** We overcome this issue by sampling from the set of possible scenarios on each path and computing the number of relocations for the sampled departure orders rather than for all possible orders. For each path \( p \), let \( \bar{z} \) be the expected relocations from the samples, that is the sampling average. Also let \( E[z] \) be the true mean. To insure a large enough sample to get a good approximation, we use the following version of Hoeffding’s inequality:

\[
P(|E[z] - \bar{z}| > \delta) \leq 2 \exp \left( \frac{-2\delta^2}{(r_{\text{max}} - r_{\text{min}})^2} \right),
\]

(11)

where \( \delta \) is a pre-specified level of desired precision.

The random variable of interest (which is the number of relocations), is bounded by \( r_{\text{max}} \) and \( r_{\text{min}} \). Note that \( r_{\text{min}} = 0 \); for \( r_{\text{max}} \), we do not have a tight bound, but we can use \( N(P - 1) \) as an upper bound (since each of the \( N \) containers is blocked by at most \( P - 1 \) containers). Let us denote the desired probability for bounding the error (i.e., RHS of (11)) by \( \epsilon \). For a given \( \delta \) and \( \epsilon \), the required number of samples can be computed as follows.

\[
S \geq \frac{r_{\text{max}}^2 \ln \frac{\xi}{\epsilon}}{2\delta^2}.
\]

(12)

By sampling from the possible scenarios on each path, we can significantly reduce the number of scenarios (and thus the computation time). For example, for a bay with 7 columns, 4 tiers, 21 containers, and \( |K| = 6 \), the total number of possible scenarios on any of the paths at time \( t^* \) is about \( 10^{11} \). Using inequality (12), the total number of scenarios would be around 30000 for \( \delta = 0.5 \), \( \epsilon = 0.05 \), and \( r_{\text{max}} = 63 \). Note that by sampling, we incur an error and ASA∗ may choose a suboptimal path \( \hat{p} \) where \( E[z_{\hat{p}}] > E[z_{p^*}] \) (recall that \( p^* \) is the optimal path that would be chosen without sampling). In the next proposition, we show that such an error, denoted by \( e_1 \triangleq E[z_{\hat{p}}] - E[z_{p^*}] \), is bounded on expectation (over a uniform distribution on the initial bay).

**Proposition 4.1.** Suppose for each path, we estimate the number of relocations using \( S \) independent samples, where \( S \) is given in (12). Also, suppose ASA∗ chooses path \( \hat{p} \). Let \( e_1 \triangleq E[z_{\hat{p}}] - E[z_{p^*}] \). Then we have \( E[e_1] \leq 2\delta \sqrt{\frac{\pi}{2\ln(\xi/\epsilon)}} \).

**Pruning the paths of \( T_{[1,t^*-1]} \).** To address this issue, we use the upper bound and lower bounds to prune the nodes of \( T_{[1,t^*-1]} \), similar to the A∗ algorithm. However, since some of the containers are unknown before \( t^* \), we have to compute the expectations (\( E[L] \) and \( E[U] \)). Again, we use the idea of sampling and we estimate these values by computing \( \bar{L} \) and \( \bar{U} \) using \( S \) samples,
where \( S \) is obtained from Inequality (12). Because of the sampling error, we may prune an optimal path by mistake. In the next two propositions, we show that such an error is bounded, whether we prune some paths once at \( t^* - 1 \) (resulting in error \( e_2 \)) or prune some paths at several time-steps before \( t^* - 1 \) (resulting in error \( e_3 \)).

**Proposition 4.2.** Suppose that for each path \( p \) at time-step \( t^* - 1 \), we estimate the expected lower and upper bounds \((L_p, U_p)\) from \( S \) samples, where \( S \) is given in (12). Also, suppose ASA* chooses to prune one or more paths (each denoted by \( \tilde{p} \)). We have 
\[
E[e_2] \leq 2\delta \sqrt{\frac{\pi}{-\ln(\frac{\epsilon}{2})}}.
\]

**Proposition 4.3.** Suppose we prune some paths at \( m \) time-steps \( t_1, t_2, \ldots, t_m < t^* - 1 \). At each time of pruning and for each path \( p \), we estimate the expected lower and upper bounds \((L_p, U_p)\) from \( S \) samples where \( S \) is given in (12). Also, suppose that at each time of pruning, ASA* chooses to prune one or more paths (each denoted by \( \tilde{p} \)). The expected total loss, \( E[e_3] \), is bounded by 
\[
m \left( \frac{\epsilon^2}{2} \frac{d_{\min}}{2} + \frac{d_{\min}}{\delta} \sqrt{-\ln(\frac{\epsilon}{2}) \frac{\pi}{2}} \right) \left( \frac{\epsilon^2}{2} \frac{d_{\max}}{2} + \frac{d_{\max}}{\delta} \sqrt{-\ln(\frac{\epsilon}{2}) \frac{\pi}{2}} + U_{\max}^{\tilde{p}} \right),
\]
where \( U_{\max}^{\tilde{p}} = \max(U_{\tilde{p}1}, \ldots, U_{\tilde{p}m}) \), and 
\( d_{\min} = \min(d_1, \ldots, d_m) \).

In Propositions 4.1-4.3 we bound the loss that can be incurred by pruning some paths at time \( t^* - 1 \) (\( e_1 \) and \( e_2 \)) or at \( t < t^* - 1 \) (\( e_3 \)). Notice that \( e_1 \) and \( e_2 \) are independent of the bay size; thus if we prune paths only at \( t^* - 1 \), the loss remains unchanged and the relative loss (as a percentage of total relocations) decreases as the bay gets larger. Therefore, for large bays and “hard-to-solve” configurations, one can pick larger \( \delta \) and \( \epsilon \) that result in a smaller number of samples, and the relative error would still be small.

Table 1 shows \( e_1, e_2 \) and \( e_3 \). The losses are very small even for combinations of \( \delta \) and \( \epsilon \) that result in reasonably small number of samples. For example, for \( \delta \leq 1 \) and \( \epsilon < 0.1 \), \( E[e_1] \) and \( E[e_2] \) are no more than 2. Notice that \( e_1 \) and \( e_2 \) are independent of the bay size; so the relative error \( \frac{E[e_1] + E[e_2]}{E[z]} \) decrease as the bay gets larger (\( E[z] \) is the average number of relocations when full information is available). For the \( e_3 \), this measure is almost constant for different bay sizes and can be controlled by changing \( \epsilon, \delta \), and \( m \) (number of times that we do pruning at \( t < t^* - 1 \)).

In Section 5 we use ASA* to solve the CRP with incomplete information for a bay with 7 columns and 4 tiers and with different amounts of information initially available. Moreover, we introduce a heuristic for this problem and compare the results of ASA* with those of the heuristic.

5 Experimental Results

In this section, several experimental results are presented to further understand the effectiveness of our algorithms in both complete and incomplete information settings.
Table 1: Left table: Expected loss due to sampling and pruning at $t^* - 1$; Right table: Expected loss due to pruning at $t < t^* - 1$ for $\epsilon = 0.05$, $\delta = 0.5$, $d_{\text{min}} = 1$, $\mathcal{U}^{\text{max}}_p = 2N$, and $m = 5$

| $\delta$ | $\epsilon$ | $\mathbb{E}[e_1]$ and $\mathbb{E}[e_2]$ | $C$ | $\mathbb{E}[e_3]$ | % error($\frac{\mathbb{E}[e_3]}{\mathbb{E}[e_3]}$) |
|---|---|---|---|---|---|
| 0.1 | 0.01 | 0.15 | 10 | 0.91 | 0.07 |
| 0.1 | 0.05 | 0.18 | 15 | 1.36 | 0.073 |
| 0.1 | 0.1 | 0.2 | 20 | 1.81 | 0.074 |
| 0.5 | 0.01 | 0.77 | 25 | 2.26 | 0.075 |
| 0.5 | 0.05 | 0.92 | 30 | 2.72 | 0.076 |
| 0.5 | 0.1 | 1.02 | 35 | 3.17 | 0.076 |
| 1 | 0.01 | 1.54 | 40 | 3.62 | 0.076 |
| 1 | 0.05 | 1.85 | 45 | 4.07 | 0.076 |
| 1 | 0.1 | 2.05 | 50 | 4.52 | 0.076 |

In the complete information case, we study thoroughly the $A^*$ algorithm and the effect of parameters on the efficiency and performance of the algorithm. First we show that using the $S_N$ lower bound improves dramatically the computational tractability of the algorithm. Moreover, our experiments verify that on the optimal path of the tree, $S_N$ increases monotonically to the optimal solution after a few levels (this result is consistent with Proposition 2.5 that states the lower bound is not decreasing on any path). We also show the trade-off between $N$ and the number of instances solved optimally. Further, as a practical experiment, we compare 3 existing heuristics ([6], [1], [11]) with the $A^*$ algorithm. We introduce a new class of heuristics, the Tree Heuristic (TH-L), and show that they outperform the existing heuristics.

In the incomplete information setting, we show through experiments that the $ASA^*$ algorithm is fast and efficient and most of the instances are solvable within a reasonable time for medium-sized instances. We introduce a myopic heuristic which expands on the H heuristic in [1]. Using the $ASA^*$ algorithm and the heuristic, we study the value of information. Our results show that while more information results in fewer relocations, the marginal value of information decreases with the level of information. We also use the myopic heuristic to examine the effect of the level of information on the number of relocations for different bay sizes. We show that for any level of information, the ratio of relocations with incomplete information and complete information converges to a constant as the bay gets larger.

5.1 Experimental results for complete information

In most of the experiments, we consider medium-sized instances of the CRP. We consider bays of size 4 tiers and 7 columns with 21 containers ($P = 4 = h + 1$, $C = 7$ and $N = 21 = hC$ with 3 containers per column). We generate randomly 100,000 instances using the uniform distribution
on bays described in Section 4.

**The effect of lower bounds on the size of the $A^*$ Tree.** In Section 2, we introduced a family of lower bounds ($S_0, S_1, \ldots, S_N$). Clearly $S_N$ is more computationally expensive to use than $S_0$, but we proved that $S_N$ was a tighter lower bound. We show here that using $S_N$ improves the performance of the $A^*$ algorithm significantly.

For each instance, we solve it using 4 types of lower bounds $S_0, S_1, S_2$ and $S_{21}$. For each of them we record the number of nodes needed to solve optimally the instance. Figure 7 presents the results. First, we show in the box-plots, the distribution of the number of nodes for each of those lower bounds. Second, we give the average number of nodes.

![Figure 7: The effect of the Lower Bound on the SDT Approach](image)

| Lower Bound | $S_N$ | $S_2$ | $S_1$ | $S_0$ |
|-------------|------|------|------|------|
| $E \ [\text{Number of Nodes}]$ | 107  | 153  | 214  | 333  |

We give two main insights from this experiment. First, the average number of nodes needed to solve with $S_N$ is $1/3$ of the number of nodes with $S_0$. Also, we point out that using $S_1$ and $S_2$ instead of $S_0$ decreases the average number of nodes by $1/3$ and $1/2$, respectively.

The fact that the number of nodes is decreasing as we use tighter lower bounds is not surprising and is aligned with what we proved in Section 2. Moreover, this experiment suggests that introducing lower bounds that looks just one or two steps ahead makes a great improvement in terms of performance of the $A^*$ approach, without affecting the computation time significantly.

Second, it can be seen that the box is concentrated around 0 for each lower bound, which implies that most instances are solved within a few hundred nodes. What influences the average number of nodes are the “hard-to-solve” cases where a large number of nodes is needed. Using $S_N$ instead
of $S_0$ makes the $A^*$ algorithm more efficient by decreasing the number of nodes by a factor of 3 for those cases. From now on, we use $S_N$ as the lower bound. Recall that actually we only need to compute $S_k$ for $k \leq N - C$, as $S_{k+1}, \ldots, S_N$ are all equal to $S_k$. Taking advantage of this property further improves computational tractability.

The convergence rate of the lower bound. After introducing the lower bounds, we have shown in Proposition 2.2 and 2.5 that on any given path, the lower bound is non-decreasing. The efficiency of the algorithm mainly depends on the rate of increase of the lower bound on each path. If this rate is high, then the algorithm has a greater chance to find the optimal solution with fewer nodes, as the tree is pruned faster. The lower bounds can either “collapse” with the upper bounds, or exceeds the best incumbent available.

In this experiment and for the sake of clarity, we only focus on the rate of convergence of the lower bound on the optimal path. For each instance, we find the optimal path, and at each level $l$, we compute the difference between $S_N(B^l)$ and $z_{opt}(B^l)$. We average this difference on all instance and plot the results in Figure 8.

![Figure 8: The convergence rate of the lower bound on the optimal path](image)

First of all, we can see that even at level 0 (meaning at the root node), the average difference is less than 0.25. This shows that on average the lower bound is close to optimal, therefore it does not need to increase a lot to reach the optimal solution. Second, we can see that after level 10, the average gap is less than 0.001, meaning that after 10 relocations, in most cases, the lower bound has reached optimality on the optimal path. Therefore, this path will terminate at level 10 if the upper bound also collapsed to the optimal. Finally, the trend of the curve shows that the gap between optimal and the lower bound decreases faster at the beginning of the tree. A simple way to explain this is that the lower bound increases when a future “bad” move can be predicted. But such bad moves become harder to predict as the bay gets emptier (there are more empty columns in the bay).

The effect of $N$. By construction, the $A^*$ method is tunable.
In this experiment, we study the impact of $N$ on the average guaranteed gap with optimality, and the percentage of instances solved optimally. Notice that from Propositions 2.2 and 2.5, we know that the guaranteed gap is a non-increasing function of $N$. We solve each instance with different $N$ and we record the gap with the optimal solution obtained by the $A^*$ algorithm. We show the results in Figure 9. On the left, we give the average gap as a function of $N$ and on the right, we show the percent of instances solved optimally.

![Graph showing the average gap as a function of $N$.](image)

![Graph showing the percentage of instances solved optimally as a function of $N$.](image)

Figure 9: The Effect of $N$

First, we observe that the average guaranteed gap at the root node is very small (less than 0.4). Also, in more than 70% of instances, the optimal solution is found at the root node. Second, as we increase $N$, the gap necessarily decreases and the number of instances solved to optimality increases.

Finally, this experiment is very useful from a practical point of view in order to determine $N$. We can set an average gap that we allow in our solutions and infer the minimum $N$ that is needed to achieve this performance on average. For example for bays of 4 by 7, if we allow for a gap of 0.05 on average, we can set our $N$ to 100.

This experiment is related to the first experiment. As shown in Figure 7, most instances only require few hundreds of nodes, but there are some that require hundreds of thousands. This could explain the “concavity” of the function in the left plot and the “convexity” of the function in the right plot.

In conclusion, this experiment highlights the trade-off between the quality of the solution (represented by the average guaranteed gap) and its tractability (represented by $N$).

**Benchmark of a new heuristic and existing heuristics.** Many heuristics have been developed for the CRP. From a practical point of view, heuristics are valuable since they are fast and easy to implement in container terminals. Therefore, it is relevant to evaluate their performance.
to advise operators. In order to benchmark existing heuristics, one can measure:

- The distribution of the difference between heuristics and the optimal solution; and
- The expected performance ratio of the heuristic defined as

\[
PR(\text{heuristic}) = \frac{z_{\text{heuristic}} - z_{\text{opt}}}{z_{\text{opt}}},
\]

(13)

where \(z_{\text{heuristic}}\) is the number of relocations in the heuristic solution. PR shows the relative gap with optimality for a given heuristic.

From Figure 7, if we set \(N\) to 400,000, then all 100,000 instances are solved optimally using the \(A^*\) algorithm. In addition to heuristic H, we study two other existing heuristics presented by Kim and Hong (KH from [6] which uses the estimation of future relocation) and Petering and Hussein (LA-5 from [11] that takes into account “repositioning moves”, i.e. repositioning containers that are not in the same column as the target container).

| Gap | Heuristic H | Heuristic KH | Heuristic LA-5 | Heuristic TH-2 |
|-----|-------------|--------------|----------------|---------------|
| Optimal Distribution | 87.0% | 31.8% | 83.5% | 95.7% |
| 1 | 11.4% | 21.4% | 13.7% | 3.98% |
| 2 | 1.4% | 21.3% | 2.0% | 0.3% |
| ≥ 3 | 0.2% | 11.4% | 0.3% | 0.02% |
| \(E[PR]\) | 1.44% | 16.0% | 1.81% | 0.44% |

Table 2: Benchmarks of Heuristics on 100,000 instances

We use the same idea as in the \(A^*\) algorithm and introduce a new class of heuristics that can improve any existing heuristics; we refer to this class as Tree Heuristic (TH-L). The basic idea of TH-L is to take the L best columns and branch on them to construct a decision tree. The L best columns can be chosen using any of the existing heuristics that compute a score for each column. Using this principle and considering several good candidates for each relocation, we are less likely to make a mistake. Here, we implement the TH-L with the H heuristic. The algorithm is presented in Algorithm 2. In our experiment, we set \(L = 2\). Note that TH results in less or the same number of relocations compared to H since the path of H is included in the tree of TH-L.

Results for the four heuristics are summarized in Table 2. First, H is optimal in most instances (87%). This is one of the main reasons that heuristic H was chosen as an upper bound in the \(A^*\) method. Second, TH-2 is indeed closer to optimality than H, KH and LA-5, in distribution and in terms of the average performance ratio.
Algorithm 2 Tree Heuristic

1: procedure \[ Z_{TH} = \text{TreeHeuristic}(B, L) \]
2: \[ Z_{TH} \leftarrow 0 \]
3: while B is not empty do
4: \quad Retrieval:
5: \quad if target container \( n \) is on top of its column then
6: \quad \quad Retrieve \( n \) from B
7: \quad end if
8: \quad end while
9: end procedure

Finally, notice that the percentages for LA-5 do not sum to 100% since there are cases for which LA-5 is better than \( A^* \), which is because LA-5 considers “repositioning moves”. As a result, for 0.5% of the 100,000 instances, LA-5 solves the instance with one relocation less than \( A^* \).

In our last experiment we study how the parameter \( L \) affects the performance of the TH-L heuristic. We consider the same 100,000 instances and solve them with \( L \) varying from 1 to 6, and we record their performance ratio. Notice that TH with \( L=6 \) considers all possibilities for the blocking container therefore it gives the same solution as the \( A^* \) without the use of bounds, so \( PR(TH-6) = 0 \) and \( L=1 \) gives heuristic H.

| \( L \) | 1     | 2     | 3     | 4     | 5     | 6     |
|-------|-------|-------|-------|-------|-------|-------|
| PR (TH-L) | 1.44% | 0.44% | 0.27% | 0.20% | 0.16% | 0%    |

Table 3: Effect of parameter \( L \) on the performance of heuristic TH-L

The main observation is that the marginal gain of branching is maximum when we increase \( L \) from 1 to 2. By considering two promising columns instead of one, TH-2 finds better solutions for most instances where H was not optimal. Note that increasing \( L \) (considering more candidate columns) will further improve the solution; however the gain from more branching is small.
considering the exponentially increasing cost of computation, which is common in these types of combinatorial optimization problems.

5.2 Experimental results for the CRP with Incomplete Information

**Number of nodes and computation time for ASA*.** Stochastic optimization methods that are based on enumerating scenarios are usually computationally expensive. The ASA* algorithm, however, is fast and tractable due to the use of sampling and pruning that allow for suppressing many nodes in the tree. More importantly, the ASA* is tunable in the sense that one can set ε and δ to change the number of nodes and thereby solve an instance within a desired amount of time, and yet ensure that the loss from using large ε or small δ is bounded (as shown in propositions 4.1-4.3).

Figure 10a shows the cumulative distribution of the number of nodes (after pruning) for the tree up to t*, for a bay with 7 columns, 4 tiers, and 3 containers per column. It can be seen that for half of the instances, the tree up to t* has 100 or fewer nodes. Also, about 90% of instances have 200 or fewer nodes. Note that the nodes of the trees that are constructed after time t* for solving instances corresponding to different scenarios do not have too much of an effect on computation time because the A* algorithm is very fast.

Figure 10b shows the cumulative density of the average computation time. It can be seen that half of the instances are solved in less than two minutes and 90% of the instances are solved in 15 minutes or less.

Figure 10: Tractability of the ASA* algorithm

Although the ASA* algorithm is fast for medium-sized bays, it might not be tractable for large bays. We next introduce a myopic heuristic and use both ASA* and the heuristic to provide several insights on the value of information.
**Myopic Heuristic.** This is an extension of the heuristic H that we explained in Section 2. Similar to the setting that we explained in Section 4, the problem is to retrieve \( N \) containers from a bay with \( C \) columns and \( P \) tiers. In the incomplete information setting, the departure order of containers \( \{1, 2, \ldots, |K|\} \) are known at time zero; the departure order of containers \( \{|K|, |K|+1, \ldots, N\} \) become known at time step \( t^* > 0 \).

Suppose \( n \) is the target container located in column \( c \). If \( n \) is not blocked by any container, it is retrieved without any relocation. Otherwise, let \( r \) be the topmost blocking container in column \( c \), and \( c^* \) be the column where \( r \) should be relocated. The myopic heuristic determines \( c^* \) using the same rules as the H heuristic explained in Section 2, except that we assign an index of \( N+1 \) to all unknown containers and we set \( \min(c_i) = N+2 \) if \( c_i \) is empty. Ties are broken arbitrarily.

**Value of information.** To study the effect of the level of information initially available (\(|K|\)), we take 1000 random instances of a bay with 7 columns, 4 tiers, 21 containers, and 3 containers per column. We solve each instance with 6 levels of information: \(|K| = \lceil 0.25N \rceil, \lceil 0.375N \rceil, \lceil 0.5N \rceil, \lceil 0.625N \rceil, \lceil 0.75N \rceil, \) and \(|K| = \lceil 0.9N \rceil \), using the ASA\(^*\) algorithm and the myopic heuristic (\( \lceil x \rceil \) is the smallest integer larger than \( x \)). For all cases, we fix \( t^* \) at \( \lceil 0.25N \rceil + 1 \) to ensure that at every time-step before \( t^* \), at least one container is known. For each of the 12 cases, we compute the average of relocations over the 1000 instances. We then compare the relative gap of each case with the average relocation for the CRP with full information: \( \frac{z_{ASA}^* - z_{opt}}{z_{opt}} \) and \( \frac{z_{MH} - z_{opt}}{z_{opt}} \), where \( z(MH) \) is the average number of relocations obtained by the myopic heuristic.

Figure 11a shows the relative gap in relocations for different levels of information. With the ASA\(^*\) algorithm, the gap is about 8% when 25% of the containers (6 containers) are known at time zero. The gap reduces to 3% when half of the containers are initially known and is almost zero when 90% of containers are known at time zero. The same behaviour can be observed for the heuristic. In both cases, the marginal value of information becomes smaller when more information is available. This is more significant for the heuristic. For example, when the level of information increases from 25% to 50%, there is a significant drop in the gap; then the gap decreases more slowly and approaches zero at 100% information. Note that with 100% information, the myopic heuristic is the same as the H heuristic.

To get an insight into the value of information for the myopic heuristic, recall that it behaves similar to the H heuristic as long as \( \min(c_i) \) for all columns are known. Since all unknown containers have larger indices than the known containers, knowing at least one container in each column is sufficient obtain the same \( \min(c_i) \) as in the heuristic H. Thus, after some point, having more information does not have much of effect on the number of relocations when we use the myopic heuristic.

Figure 11b shows how the gap between the myopic heuristic and ASA\(^*\) shrinks as the level of information increases. When 25% of the containers are initially known, using the myopic heuristic results in 12% more relocations on average (compared to ASA\(^*\)). This gap drops to less than 2%
when all containers are known. Note that when more than 50% of the containers are initially known, the solution provided by the myopic heuristic is reasonably close to the solution of ASA* (less than 5%). Therefore, the myopic heuristic can be used in practice as it is easy to implement and efficient.

(a) Value of information: relative gap in relocations with the case of full information

(b) Comparing the myopic heuristic and the ASA* algorithm: relative gap in relocations for different levels of information

Figure 11: Experiment results for 7×4 bay

It is interesting to see the effect of different levels of information on the number of relocations in larger bays. Figure 12 shows the average relative gap in relocations for bays of different sizes. For each bay size, the average relative gap is computed for 100,000 instances, using the myopic heuristic. The most important observation is that the ratio of $E[z_{MH}]$ and $E[z_H]$ does not constantly increase with bay size. And, for each information level, this ratio converges to a constant as the bay gets larger. Two other observations can be made from Figure 12. First, the converging ratio for large bays drops fast as the level of information increases. For example, when 25% of containers are initially known, the ratio converges to 40%. This number drops to 3% when the level of information is 75%. Second, the rate of convergence is much faster when more information is available. For example, when 75% or more of the containers are initially known, the ratio is almost constant independent of the bay size. When 25% of the containers are known, the ratio converges to 1.4 for bays with 40 or more columns. Last, note that the asymptotic behaviour of the myopic heuristic in Figure 12 can be considered as an upper bound on the performance of the ASA* algorithm.

What is shown in Figure 12 provides useful insights for port operators. Considering the gain (fewer relocations) at each level of information, port operators can design appointment systems that capture this gain by offering faster or discounted service to the customers who provide their arrival time in advance.
Figure 12: Asymptotic behaviour of myopic heuristic: $\frac{E[z_{MH}]}{E[z_H]}$ approaches a constant as the bay gets larger.

**Benchmark of the myopic heuristic and existing heuristic.** To the best of our knowledge, the CRP with incomplete information has not been much studied. Here we compare our myopic heuristic with the RDH (revised difference heuristic) [16]. Zhaoh et al. show the percentage savings of RDH over a nearest relocation strategy for different bay sizes and for different amounts of information initially available. The greatest savings is about 50% and is realized for a bay with 12 columns and stack capacity of 7, with 50 or more containers initially known (see Figure 13 of [16]). The total number of containers is not indicated for this simulation, so we assume the bay is full with 72 containers. (With 50 known containers, the level of information is about 70%). For a bay with 12 columns and 3 containers per column, the maximum savings is about 40% and is realized when 22 or more containers are known (i.e., when the information level is about 60%).

In Figure 13, we show the percentage savings of the myopic heuristic over the nearest location strategy for bays with different numbers of columns, with 4 tiers and 3 containers per column. The savings are shown for 4 different levels of information. It can be seen that when level of information is 50%, the saving is about 48%, which is greater than the saving for RDH when even more information (about 70%) is available. When the level of information is 25%, the savings of the myopic heuristic is about 35%, whereas for the same setting, RDH results in less than 25% savings over the nearest relocation strategy.

6 Discussion

Managing the relocation moves is one of the main challenges in the storage yard of container terminals and has a direct effect on the costs and efficiency of yard operations. The Container
Relocation Problem (CRP) is notorious for its computational intractability and most research studies have designed heuristics in order to solve the problem, particularly for large bays.

In this paper, we revisited a fast and efficient method, the $A^*$ algorithm that provides optimal solution for the CRP. We proved several properties of the method as applied to the CRP and assessed its performance and computational tractability through experiments. We also studied the asymptotic behavior of optimal solutions of the CRP and showed that the ratio of the optimal number of relocations and a simple counting lower bound converges to 1 as the bay gets very large. Moreover, we showed through experiments that the H heuristic (introduced in [1]) has the same behavior. An important result of our experiments is that it is recommended to apply the heuristic H to large blocks of containers (several bays) in order to minimize the loss from using a heuristic rather than optimization methods (that are computationally expensive). We also presented a new heuristic and show through experiments that it outperforms existing heuristics.

Further, we extended our study to the CRP with incomplete information, which is an important case in practice because the retrieval order of containers are usually not known far in advance. We introduced a two-stage stochastic optimization framework ($ASA^*$) that optimizes the expected number of relocations given a probabilistic distribution on the departure order of containers. We also presented a myopic heuristic that is very fast and efficient for the CRP with incomplete information. We used $ASA^*$ and the heuristic to study the value of information for different bay sizes and showed that the relative gap between $ASA^*$ and the heuristic shrinks as the level of information increases. In fact, when the available information is more than 50%, the myopic heuristic is very efficient and can be used in practice. We also showed that the ratio of the number of relocations between heuristic H and the myopic heuristic converges to a constant fast as bays get larger, which again implies that the loss from using the myopic heuristic is minimized when

![Figure 13: Percentage of saving of myopic heuristic over nearest relocation strategy.](image)

Figure 13: Percentage of saving of myopic heuristic over nearest relocation strategy.
applied to large bays.

References

[1] Caserta, M., Schwarze, S., and Vo, S. (2012). A mathematical formulation and complexity considerations for the blocks relocation problem. *European Journal of Operational Research*, 219(1):96 – 104.

[2] Caserta, M., Schwarze, S., and Voß, S. (2009). A new binary description of the blocks relocation problem and benefits in a look ahead heuristic. 5482:37–48.

[3] Caserta, M., Voß, S., and Sniedovich, M. (2011). Applying the corridor method to a blocks relocation problem. *Or Spectrum*, 33(4):915–929.

[4] Forster, F. and Bortfeldt, A. (2012). A tree search procedure for the container relocation problem. *Computers & Operations Research*, 39(2):299 – 309.

[5] Hakan Akyz, M. and Lee, C.-Y. (2014). A mathematical formulation and efficient heuristics for the dynamic container relocation problem. *Naval Research Logistics (NRL)*, 61(2):101–118.

[6] Kim, K. H. and Hong, G.-P. (2006). A heuristic rule for relocating blocks. *Computers & Operations Research*, 33(4):940–954.

[7] Lee, Y. and Lee, Y.-J. (2010). A heuristic for retrieving containers from a yard. *Computers & Operations Research*, 37(6):1139–1147.

[8] Lehnfeld, J. and Knust, S. (2014). Loading, unloading and premarshalling of stacks in storage areas: Survey and classification. *European Journal of Operational Research*, 239(2):297 – 312.

[9] Matoušek, J. and Vondrák, J. (2001). *The Probabilistic Method: Lecture Notes*. KAM-DIMATIA series. Charles Univ.

[10] Olsen, M. and Gross, A. (2014). Average case analysis of blocks relocation heuristics. *5th International Conference, ICCL 2014, Valparaiso, Chile, September 24-26, 2014. Proceedings*, 8760:81 – 92.

[11] Petering, M. E. and Hussein, M. I. (2013). A new mixed integer program and extended look-ahead heuristic algorithm for the block relocation problem. *European Journal of Operational Research*, 231(1):120 – 130.

[12] Stahlbock, R. and Voß, S. (2008). Operations research at container terminals: a literature update. *OR Spectrum*, 30(1):1–52.
[13] Tanaka, S. and Takii, K. (2014). A faster branch-and-bound algorithm for the block relocation problem. *Automation Science and Engineering (CASE), 2014 IEEE International Conference on*, pages 7–12.

[14] Ünlüyurt, T. and Aydın, C. (2012). Improved rehandling strategies for the container retrieval process. *Journal of Advanced Transportation*, 46(4):378–393.

[15] Zhang, H., Guo, S., Zhu, W., Lim, A., and Cheang, B. (2010). An investigation of IDA* algorithms for the container relocation problem. *Proceedings of the 23rd International Conference on Industrial Engineering and Other Applications of Applied Intelligent Systems - Volume Part I*, pages 31–40.

[16] Zhao, W. and Goodchild, A. V. (2010). The impact of truck arrival information on container terminal rehandling. *Transportation Research Part E: Logistics and Transportation Review*, 46(3):327–343.

[17] Zhu, W., Qin, H., Lim, A., and Zhang, H. (2012). Iterative deepening A* algorithms for the container relocation problem. *Automation Science and Engineering, IEEE Transactions on*, 9(4):710–722.
Appendix

Proof of Proposition 2.2. Between level $l$ and $l + 1$, one move has been operated ($r$ moved to column $c_i$). If $r < \min(c_i)$ ("good" move) then $r$ was a blocking container in $B^l$ but not in $B^{l+1}$, thus $S_0(B^{l+1}) = S_0(B^l) - 1$. If $r > \min(c_i)$, we have $S_0(B^{l+1}) = S_0(B^l)$. So $S_0(B^{l+1}) = S_0(B^l) - \chi(r < \min(c_i))$ where $\chi$ is the indicator function. Therefore $L_0(B^{l+1}) = S_0(B^{l+1}) + (l + 1) = S_0(B^l) - \chi(r < \min(c_i)) + 1 + l \geq S_0(B^l) + l = L_0(B^l)$. □

Proof of Proposition 2.4. We have $L_n(B) - L_{n-1}(B) = \sum_{r \in R_n(B)} \chi(r > MOM(B_n))$. Since $\chi$ is a non negative function, the results follows. □

Proof of Proposition 2.5. Intuitively, this property comes from the fact that $S_n(B^{l+1}) \geq S_n(B^l) - 1$. Indeed, if this holds, we have

$$L_n(B^{l+1}) = S_n(B^{l+1}) + l + 1 \geq S_n(B^l) - 1 + l + 1 = S_n(B) + l = L_n(B)$$

So let us prove $S_n(B^{l+1}) \geq S_n(B^l) - 1$. Let $r$ be the container that is relocated between $B^l$ and $B^{l+1}$. We have two cases:

- If $r$ can do a good move, it has been counted $r$ only once in $S_n(B^l)$ as it is a blocking container in $B^l$ and it might or might not contribute to $S_n(B^{l+1})$. Therefore $S_n(B^{l+1}) \geq S_n(B^l) - 1$.

- If $r$ cannot do a good move, it has been counted twice in $S_n(B^l)$. But since it cannot do a good move, it is necessarily a blocking container in $B^{l+1}$ so it counts at least once in $S_n(B^{l+1})$. Thus $S_n(B^{l+1}) \geq S_n(B^l) - 1$.

This completes the proof. □

Proof of Proposition 2.7. Using heuristic $H$, a container is relocated only if it is blocking. For any blocking container, there exists at least one empty column which means there is a possible good move. So each blocking container is relocated at most once, i.e, $z_H(B) = S_0(B)$. Since $S_0(B) \leq z_{opt}(B) \leq z_H(B) = S_0(B)$ and we have equation [4]. □

Proof of Proposition 2.8. If 1 is not blocked then it is retrieved. In that case, there are $C$ containers left and thus we use Proposition 2.7 (note that in that case $S_1(B) = S_0(B)$).

Then let us suppose 1 is blocked by some containers. If there is more than one blocking container, all except the last one can be relocated to empty columns. So we come back to the case where 1 is only blocked by one container.

Now we consider two cases: If $r$ can do a good move and $H$ relocates $r$ to a good column and then retrieves 1. We call $B'$ the new configuration. By Proposition 2.7 we have $z_H(B') = S_0(B') = S_1(B')$. Notice that $z_H(B) = z_H(B') + 1$ and $S_1(B) = S_1(B') + 1$ and the result follows.

Now suppose $r$ cannot do a good move. This can only happen if $r = C + 1$ and all the other columns have exactly one container. It is easy to check that in that case, $z_H(B) = S_1(B)$. □
Proof of Proposition 2.9. Let’s introduce some notations. Let B be the initial configuration, B’ the configuration obtained with the heuristic after retrieving container 1 and |R1| the number of containers blocking container 1. Since B’ has C+1 containers or less, S_{1}(B’) \leq S_0(B’) + 1. Since there are C+1 containers and |R1| on the top of 1, there are at least (C-1) - (C+1 - (|R1| + 1)) = |R1| - 1 empty columns which are “good” columns for any containers, so S_0(B’) \leq S_0(B) - (|R1| - 1). Finally,

\[ z_H(B) = |R1| + z_H(B') \]
\[ \leq |R1| + S_1(B') \]
\[ \leq |R1| + S_0(B) + 2 - |R1| \]
\[ \leq S_0(B) + 2 \leq z_{opt} + 2. \]

The proof is very similar for C+k and works by induction. For k=2, the previous paragraph proves a stronger inequality. Let’s suppose it is true for k-1, and B has C+k containers. We have by induction hypothesis, \( z_H(B') \leq S_0(B') + \frac{k(k-1)}{2} \). Moreover \( S_0(B') \leq S_0(B) + k - |R1| \) (since there are at least (C-1) - (C + k - (|R1| + 1)) = |R1| - k empty columns). Finally \( z_H(B) = |R1| + z_H(B') \), so

\[ z_H(B) = |R1| + \frac{k(k-1)}{2} \]
\[ \leq |R1| + S_0(B) + \frac{k(k-1)}{2} \]
\[ = S_0(B) + \frac{k(k+1)}{2} \leq z_{opt}(B) + \frac{k(k+1)}{2}. \]

\[ \Box \]

Proof of Proposition 3.1. Let \( S_{1}^i(B_C) \) be the number of blocking containers in column \( i \). By the linearity of expectation, we have

\[ \mathbb{E}_C[S_0(B_C)] = \mathbb{E}_C \left[ \sum_{i=1}^{C} S_{1}^i(B_C) \right] = \sum_{i=1}^{C} \mathbb{E} \left[ S_{1}^i(B_C) \right] = \alpha_h \times C, \]

where \( \alpha_h = \mathbb{E} \left[ S_{1}^i(B_C) \right] \). The last equality comes from the fact that each column is identically distributed.

Now let us compute \( \alpha_h \). By definition, \( \alpha_h = \sum_{k=0}^{h-1} kp_{k,h} \) for \( k \leq h - 1 \). Let us start with \( p_{0,h} \).

Given \( h \) randomly chosen containers, the probability that they are placed in the first column in a descending order (meaning there is no blocking container) is \( p_{0,h} = \frac{1}{n} \). Now we compute \( p_{k,h} \) by recursion. Let \( n^* \) be the smallest container among these \( h \) containers. If \( n^* \) is located in tier \( j \) it means that \( (h-j) \) containers are blocking \( n^* \). Moreover, there are \( (k - (h-j)) \) blocking containers below tier \( j \) with probability \( p_{k-(h-j),j-1} \). Summing over all positions of \( n^* \), we have

\[ p_{k,h} = \sum_{j=1}^{h} \frac{1}{n} p_{k-(h-j),j-1} = \sum_{j=1}^{h} \frac{1}{n} p_{k-(h-j),j-1} \]
since $p_{k,h} = 0$ for $k > h - 1$ and $k < 0$.

**Proof of Theorem 3.3.** Since for all configurations $B_C$, $z_{opt}(B_C) \geq S_0(B_C)$ then $\frac{E_C[z_{opt}(B_C)]}{E_C[S_0(B_C)]} \geq 1$.

Moreover, we have:

\[
\frac{E_C[z_{opt}(B_C)]}{E_C[S_0(B_C)]} = 1 + \frac{E_C[z_{opt}(B_C)] - E_C[S_0(B_C)]}{E_C[S_0(B_C)]} = 1 + \frac{1}{\alpha_h C} \left( E_C[z_{opt}(B_C)] - \alpha_h C \right)
\]

which shows that the function $g(C)$ is essentially bounded.

Using Lemma 6.1, we have for all $C \geq h + 1$:

\[
E_{C+1} [z_{opt}(B_{C+1})] \leq E_C [z_{opt}(B_C)] + \alpha_h + h(P - 1)(C + 1)e^{-\theta(C+1)}
\]

\[
\Rightarrow E_{C+1} [z_{opt}(B_{C+1})] - \alpha_h (C + 1) \leq E_C [z_{opt}(B_C)] - \alpha_h C + h(P - 1)(C + 1)e^{-\theta(C+1)}
\]

\[
\Rightarrow g(C + 1) \leq g(C) + h(P - 1)(C + 1)e^{-\theta(C+1)}
\]

\[
\Rightarrow g(C) \leq g(h + 1) + h(P - 1) \sum_{i=h+2}^{C} (ie^{-\theta i}) \leq g(h + 1) + h(P - 1) \sum_{i=1}^{\infty} (ie^{-\theta i})
\]

\[
\Rightarrow g(C) \leq g(h + 1) + \frac{e^\theta h(P - 1)}{(e^\theta - 1)^2} = K'.
\]

Therefore using equations (14) and (16), we have

\[
\frac{E_C[z_{opt}(B_C)]}{E_C[S_0(B_C)]} \leq 1 + \frac{K}{C} = f(C),
\]

where

\[
K = \frac{K'}{\alpha} = \frac{g(h + 1) + \frac{e^\theta h(P - 1)}{(e^\theta - 1)^2}}{\alpha}
\]

which completes the proof of the theorem. □

**Lemma 6.1.** Let $z_{opt}(B_C)$ be the optimal number of relocations for configuration $B$, then there exists a constant $\theta$ (defined in equation (37)) such that:

\[
E_{C+1} [z_{opt}(B_{C+1})] \leq E_C [z_{opt}(B_C)] + \alpha_h + h(P - 1)(C + 1)e^{-\theta(C+1)}, \forall C \geq h + 1
\]
Proof of Lemma 6.1. Now we need to prove equation (19). Define a column to be “special” if all its containers are not smaller than \( \omega = (h-1)(C+1)+1 \). Now let us consider the following event:

\[
\Omega = \left\{ \text{The bay with } C+1 \text{ columns has at least one “special” column} \right\}.
\]

The intuition is the following: the probability of having a “special” column grows quickly to 1 as a function of \( C \) implying that the event \( \Omega \) happens with high probability. Now, conditioned on \( \Omega \), we more easily express the difference between bays of size \( C+1 \) and \( C \) in the following way. We claim that

\[
\mathbb{E}_{C+1} \left[ z_{opt}(B_{C+1}) | \Omega \right] \leq \mathbb{E}_C \left[ z_{opt}(B_C) \right] + \alpha h.
\]

Let \( B_{C+1} \) be a given bay of size \( C+1 \) that verifies \( \Omega \). Since columns in bays can be interchanged, we suppose that a “special” column is the first (leftmost) column of the bay. We also denote \( n_1, n_2, \ldots, n_h \) the containers of the first column. We know that \( n_1, n_2, \ldots, n_h \geq \omega \) and \( n_1 \neq n_2 \neq \ldots \neq n_h \). Finally let \( B_C \) be the bay \( B_{C+1} \) without its first column (see Figure 14).

![Figure 14: Bay decomposition of \( B_{C+1} \)](image)

First we prove that

\[
z_{opt}(B_{C+1}) \leq z_{opt}(B_C) + S_0 \begin{pmatrix} n_1 \\ \ldots \\ n_h \end{pmatrix}.
\]

To prove equation (22), we construct a feasible sequence \( \sigma \) for the bay of size \( C+1 \) for which the number of relocations is equal to the right side of equation (22). Let \( \sigma_{opt}(B_C) \) the optimal sequence for \( B_C \), \( t' \) be the first time step when the target container in \( \sigma_{opt}(B_C) \) is larger than \( \min\{n_1, n_2, \ldots, n_h\} \) and \( B'_C \) be the bay obtained at \( t' \) using \( \sigma_{opt}(B_C) \). Let the first \( t' - 1 \) moves of \( \sigma \) be the first \( t' - 1 \) moves of \( \sigma_{opt}(B_C) \). Note that \( B'_C \) has at most \( C-h \) containers due to the choice of \( \omega \). By Proposition 2.7, the number of relocations performed by \( \sigma_{opt}(B_C) \) from \( t' \) until the end is \( S_0(B'_C) \). Therefore

\[
z_{opt}(B_C) = \# \text{ relocations up to } t' \text{ done by } z_{opt}(B_C) + S_0(B'_C).
\]
After $t'$, we run heuristic H on $B'_{C+1} = \begin{pmatrix} n_1 \\ \vdots \\ n_h \end{pmatrix} \cup B'_C$.

We claim that $z_0$ is exactly the right side of equation (22). There are at most $C$ containers in $B'_{C+1}$, therefore using Proposition 2.7 we know that if we apply the heuristic H to this configuration, then the number of relocations done by H is $S_0(B'_{C+1}) = S_0(B'_C) + S_0\left( \begin{pmatrix} n_1 \\ \vdots \\ n_h \end{pmatrix} \right)$. Therefore

$$z_{opt}(B_{C+1}) \leq z_0(B_{C+1}) = \# \text{ relocations up to } t' \text{ done by } z_{opt}(B'_C) + S_0(B'_C) + S_0\left( \begin{pmatrix} n_1 \\ \vdots \\ n_h \end{pmatrix} \right),$$

and we have

$$z_{opt}(B_{C+1}) \leq z_{opt}(B_C) + S_0\left( \begin{pmatrix} n_1 \\ \vdots \\ n_h \end{pmatrix} \right), \text{property} \quad (24)$$

which proves equation (22).

Now we can take the expectation from both sides of equation (21) over a uniform distribution of the rest of the $hC$ containers that are not in the first column. We claim that the first term on the right hand-side of equation (21) is exactly $E_C[z_{opt}(B)]$. For any configuration that appears in $B_C$, we can map it to a unique configuration where all containers are between 1 and $hC$. Thus,

$$E_C \left[ z_{opt}(B_{C+1}) \begin{pmatrix} n_1 \\ \vdots \\ n_h \end{pmatrix} \right] \leq E_C \left[ z_{opt}(B_C) \right] + S_0\left( \begin{pmatrix} n_1 \\ \vdots \\ n_h \end{pmatrix} \right). \quad (25)$$

Next, we take the expectation of both sides of equation (25) over possible first columns, which is a “special” column. Now notice that if $B_{C+1}$ is generated uniformly in the sets of bays of size $C+1$, then conditioned on $\Omega$, the probability of having a certain column $[n_1, \ldots, n_h]^T$ is identical for any $n_1 \neq \ldots \neq n_h \geq \omega$ and it is given by

$$P \left( \begin{pmatrix} n_1 \\ \vdots \\ n_h \end{pmatrix} \bigg\vert \Omega \right) = \frac{(C-h)!}{C!} \cdot (26)$$

Moreover, if we withdraw this first column from $B_{C+1}$ and look at the last $C$ columns (after renum-bering the containers so that the numbers lie between 1 and $hC$), then this bay is also uniformly
distributed in the sets of bays with $C$ columns. Therefore we can write:

\[
E_{C+1} \left[z_{opt}(B_{C+1})\right] = \sum_{(n_1, \ldots, n_h) \atop n_i \neq n_j \atop n_i \geq \omega} \left( E_{C} \left[z_{opt}(B_{C+1})\right] \left[ \begin{array}{c} n_1 \\ \vdots \\ n_h \end{array} \right], \Omega \right) \times P \left( \left[ \begin{array}{c} n_1 \\ \vdots \\ n_h \end{array} \right] | \Omega \right) \\
= \sum_{(n_1, \ldots, n_h) \atop n_i \neq n_j \atop n_i \geq \omega} \left( E_{C} \left[z_{opt}(B_{C+1})\right] \left[ \begin{array}{c} n_1 \\ \vdots \\ n_h \end{array} \right] \right) \times P \left( \left[ \begin{array}{c} n_1 \\ \vdots \\ n_h \end{array} \right] | \Omega \right) \\
\leq E_{C} \left[z_{opt}(B_{C})\right] \sum_{(n_1, \ldots, n_h) \atop n_i \neq n_j \atop n_i \geq \omega} P \left( \left[ \begin{array}{c} n_1 \\ \vdots \\ n_h \end{array} \right] | \Omega \right) + \sum_{(n_1, \ldots, n_h) \atop n_i \neq n_j \atop n_i \geq \omega} S_0 \left( \left[ \begin{array}{c} n_1 \\ \vdots \\ n_h \end{array} \right] \right) \times P \left( \left[ \begin{array}{c} n_1 \\ \vdots \\ n_h \end{array} \right] | \Omega \right) \\
\leq E_{C} \left[z_{opt}(B_{C})\right] + \sum_{(n_1, \ldots, n_h) \atop n_i \neq n_j \atop n_i \geq \omega} S_0 \left( \left[ \begin{array}{c} n_1 \\ \vdots \\ n_h \end{array} \right] \right) \times P \left( \left[ \begin{array}{c} n_1 \\ \vdots \\ n_h \end{array} \right] | \Omega \right) \\
(27)
\leq 1.

The equality between (27) and (28) comes from the fact that if we know that $B_{C+1}$ has a “special” column, then we do not need to condition on $\Omega$. Equation (29) uses the fact that

\[
\sum_{n_1 \neq \ldots \neq n_h \atop n_i \geq \omega} P \left( \left[ \begin{array}{c} n_1 \\ \vdots \\ n_h \end{array} \right] | \Omega \right) = 1.
\]

Note that, given any $(n_1, \ldots, n_h)$ such that $n_i \neq n_j$, we have

\[
E \left[ S_0 \left( \left[ \begin{array}{c} n_1 \\ \vdots \\ n_h \end{array} \right] \right) \right] = \alpha_h,
\]

when the expectation is over a random order of $(n_1, \ldots, n_h)$. This is true regardless of the set $(n_1, \ldots, n_h)$ that is drawn from (See Remark 3.2). This implies that the second term in the right hand side of equation (29) is equal to $\alpha_h$; Therefore, we get equation (27).

Lemma 4.2 states that the event $\Omega$ has a probability that increases exponentially fast to 1 as a function of $C$.

Now we want to focus on the event $\overline{\Omega}$. We give an upper bound on $E_{C+1} \left[z_{opt}(B_{C+1})|\overline{\Omega}\right]$. For any configuration, in order to retrieve one container, we need at most $P - 1$ relocations (since at most $P - 1$ containers are blocking it), thus for any configuration, the optimal number of relocations is at most $P - 1$ times the number of containers ($h(C + 1)$) which gives us $h(P - 1)(C + 1)$ as an upper bound on the optimal number of relocations. We use this universal bound to get

\[
E_{C+1} \left[z_{opt}(B_{C+1})|\overline{\Omega}\right] \leq h(P - 1)(C + 1).
\]
Finally using Lemma 6.2 we have

$$\mathbb{E}_{C+1}[z_{opt}(B_{C+1})] = \mathbb{E}_{C+1}[z_{opt}(B_{C+1})|\Omega]\mathbb{P}(\Omega) + \mathbb{E}_{C+1}[z_{opt}(B_{C+1})|\overline{\Omega}]\mathbb{P}(\overline{\Omega})$$

$$\leq \mathbb{E}_{C+1}[z_{opt}(B_{C+1})|\Omega] + \mathbb{E}_{C+1}[z_{opt}(B_{C+1})|\overline{\Omega}] \times e^{-\theta(C+1)}$$

$$\leq \mathbb{E}_{C}[z_{opt}(B_{C})] + \alpha_{h} + h(P-1)(C+1)e^{-\theta(C+1)},$$

which completes the proof of Lemma 6.1. \qed

Lemma 6.2. Let $\Omega$ be the event defined by equation (27), then there exists a constant $\theta > 0$ such that

$$\mathbb{P}(\overline{\Omega}) \leq e^{-\theta(C+1)}$$

where $\theta$ is given by equation (37).

Proof of Lemma 6.2. Recall that

$$\Omega = \{ \text{The bay with } C+1 \text{ columns has at least one “special” column} \}.$$ 

We know that each bay of size $C+1$ can be mapped to a permutation $\pi$ of $S_{h(C+1)}$ taken uniformly at random. Let $q(.)$ be the function from $S_{h(C+1)}$ to $\mathbb{R}^+$ defined by

$$q : \pi \mapsto \text{number of “special” columns in the resulting bay of } \pi.$$ 

Note that

$$\mathbb{P}(\overline{\Omega}) = \mathbb{P}(q(\pi) = 0).$$

First we compute the expected value of $q(.)$

$$\mathbb{E}_{C+1}[q] = \mathbb{E}_{C+1}\left[ \sum_{i=1}^{C+1} \chi(c_i \text{ is a “special” column}) \right]$$

$$\mathbb{E}_{C+1}[q] = (C+1) \times \mathbb{P}(\{|c_1\text{ is a “special” column}|),$$

where we use linearity of expectation and the fact that columns are identically distributed.

A simple counting implies that [1]:

$$\mathbb{P}(\{|c_1\text{ is a “special” column}|) = \frac{(C+1)((C+1)-1)\ldots(C+1-h+1)}{h(C+1)[h(C+1)-1]\ldots[h(C+1)-h+1]}$$

$$\geq \left( \frac{(C+1)-h+1}{h(C+1)} \right)^h$$

$$\geq \left( \frac{2}{h(h+1)} \right)^h,$$

\(\text{(33)}\)

1Notice that when $C \to \infty$, the probability is equivalent to $(\frac{1}{h})^h$ which would guarantee a faster convergence rate.
where we use $C + 1 \geq h + 1$ to show the last inequality.

Therefore we know that

$$
\mathbb{E}_{C+1} [q] \geq (C + 1) \times \left( \frac{2}{h(h + 1)} \right)^h.
$$

We claim that $q(.)$ is well concentrated around its mean. In order to do so, we prove that $q(.)$ is 1-Lipschitz. Define $\rho$ the distance between two permutations $\pi_1, \pi_2 \in S_{h(C+1)}$ as $\rho(\pi_1, \pi_2) = |\{i \in [h(C + 1)] : \pi_1(i) \neq \pi_2(i)\}|$. We want to prove that

$$
|q(\pi_1) - q(\pi_2)| \leq \rho(\pi_1, \pi_2), \forall (\pi_1, \pi_2) \in S_{h(C+1)}.
$$

Let $\pi_1, \pi_2 \in S_{h(C+1)}$. Let us first consider the case where $\rho(\pi_1, \pi_2) = 2$. (Notice that if $\rho(\pi_1, \pi_2) \neq 0$ then $\rho(\pi_1, \pi_2) \geq 2$). In that case, we have $i, j \in [1, \ldots, n]$ such that $\pi_1(i) = \pi_2(j)$ and $\pi_1(j) = \pi_2(i)$. Let $B_1$ and $B_2$ be the configurations generated by $\pi_1$ and $\pi_2$. Having $\rho(\pi_1, \pi_2) = 2$ corresponds to the fact that if we swap 2 containers in $B_1$, we get $B_2$ and we denote those containers $a = \pi_1(i)$ and $b = \pi_1(j)$. We have three cases:

- $a$ and $b$ are both in “special” columns in $B_1$. In this case, swapping them will not change anything since both their new columns in $B_2$ will also be “special” and hence $|q(\pi_1) - q(\pi_2)| = 0$.

- $a$ and $b$ are both in columns that are not “special” columns in $B_1$. If $a, b \geq \omega$ or $a, b < \omega$ then we do not create any new special column in $B_2$. Now suppose that $a \geq \omega$ and $b < \omega$, then the column of $a$ in $B_2$ might be a “special” column, but the column of $b$ in $B_2$ cannot be “special”. Therefore in that case, $|q(\pi_1) - q(\pi_2)| \leq 1$.

- $a$ is in a “special” column in $B_1$ but $b$ is not. Now we know that $a \geq \omega$. If $b < \omega$ then the column of $b$ in $B_2$ cannot be “special” but the column of $a$ might be and in that case $|q(\pi_1) - q(\pi_2)| \leq 1$. If $b \geq \omega$, then the column of $b$ in $B_2$ is “special” and the column of $a$ in $B_2$ is not “special” which gives us $|q(\pi_1) - q(\pi_2)| = 0$. Note that the proof is identical if $b$ is in a “special” column in $B_1$ but $a$ is not.

So far we have shown that

$$
\text{If } \rho(\pi_1, \pi_2) = 2, \text{ then } |q(\pi_1) - q(\pi_2)| \leq 1.
$$

Now we suppose that $\rho(\pi_1, \pi_2) = k$ where $2 \leq k \leq h(C + 1)$. Note that we can construct a sequence of permutations $(\pi_1', \pi_2', \ldots, \pi_k')$ such that $\pi_1' = \pi_1$, $\pi_k' = \pi_2$ and $\rho(\pi_i', \pi_{i+1}') = 2$.

Now using this fact and equation (36),

$$
|q(\pi_1) - q(\pi_2)| = \sum_{i=1}^{k-1} |q(\pi_i') - q(\pi_{i+1}')| \leq \sum_{i=1}^{k-1} |q(\pi_i') - q(\pi_{i+1}')| \leq \sum_{i=1}^{k-1} 1 = k - 1 \leq k = \rho(\pi_1, \pi_2),
$$

39
which proves that \( q(\cdot) \) is 1-Lipschitz.

Now we use Theorem 8.3.3 of Matoušek and Vondrák, which states that 
\[
\mathbb{P}(q \leq \mathbb{E}_{C+1}[q] - t) \leq e^{-\frac{t^2}{8h(h+1)}}
\]
and apply it with \( t = \mathbb{E}_{C+1}[q] \) and equation (34) to get
\[
\mathbb{P}(q = 0) = \mathbb{P}(q \leq \mathbb{E}_{C+1}[q] - \mathbb{E}_{C+1}[q]) \leq e^{-\frac{(E_{C+1}(q))^2}{8h(h+1)}} \leq e^{-\theta(C+1)},
\]
where
\[
\theta = \frac{1}{8h\left(\frac{2}{h(h+1)}\right)^{2h}} > 0,
\]
which concludes the proof. \( \Box \)

**Proof of Corollary 9.** Using that \( S_0(B_C) \leq z_{gen}(B_C) \leq z_{opt}(B_C) \), we have \( 1 \leq \frac{E_{C}[z_{gen}(B_C)]}{E_{C}[S_0(B_C)]} \leq \frac{E_{C}[z_{opt}(B_C)]}{E_{C}[S_0(B_C)]} \leq f(C) \). Finally we know that \( E_{C}[S_0(B_C)] = a_0C \). \( \Box \)

**Proof of Proposition 4.1.** We can write \( E[e_1] \) as follows:
\[
E[e_1] = (E[E[z_p] - \bar{z}_p]) - (E[E[z_{p^*}] - \bar{z}_{p^*}]) + E[\bar{z}_p - \bar{z}_{p^*}] \leq E[|E[z_p] - \bar{z}_p|] + E[|E[z_{p^*}] - \bar{z}_{p^*}|]. \tag{38}
\]

The above inequality holds because ASA\(^*\) chose \( \bar{p} \), and thus we have \( \bar{z}_p < \bar{z}_{p^*} \). Moreover, we can compute \( E[|E[z_p] - \bar{z}_p|] \) and \( E[|E[z_{p^*}] - \bar{z}_{p^*}|] \) using their CDF. We denote \( |E[z_p] - \bar{z}_p| \) by \( \Delta \). Note that \( 0 \leq \Delta \leq \max \) and inequality (11) gives a bound on its CDF, when estimating \( \bar{z}_p \) with \( S \) samples. For \( E[\Delta] \), we have:
\[
E[\Delta] = \int_0^{\max} 1 - F_\Delta(x)dx \tag{39}
\]
\[
\leq \int_0^{\max} 2 \exp(-2Sx^2)dx \tag{40}
\]
\[
= 2 \int_0^{\max} \left(\frac{\pi}{2}\right)^2 dx \leq 2 \int_0^{\infty} \left(\frac{\pi}{2}\right)^2 dx = \delta \sqrt{\frac{\pi}{\ln(\frac{\pi}{2})}}. \tag{41}
\]

We can compute \( E[|E[z_{p^*}] - \bar{z}_{p^*}|] \) in a similar way.

The first equality in (41) is obtained by substituting \( S \) with the expression in (12). Using the bound on \( E[\Delta] \) and inequality (38), we can compute \( E[e_1] \) as follows:
\[
E[e_1] \leq 2\delta \sqrt{\frac{\pi}{-\ln(\frac{\pi}{2})}}. \tag{42}
\]
\( \Box \)
Proof of Proposition 4.2. For an arbitrary initial bay $B$, let $\hat{\rho}$ be the path with the minimum estimated upper bound. Notice that pruning $\tilde{\rho}$ may be a mistake if the true mean of $L_{\tilde{\rho}}$ is less than the true mean of $U_{\tilde{\rho}}$, i.e., $E[L_{\tilde{\rho}}] < E[U_{\tilde{\rho}}]$; in such a case, $e_2 \leq E[U_{\tilde{\rho}}] - E[z_{\tilde{\rho}}] \leq E[U_{\tilde{\rho}}] - E[L_{\tilde{\rho}}]$.

Otherwise $e_2$ is zero; thus we have $e_2 \leq (E[U_{\tilde{\rho}}] - E[L_{\tilde{\rho}}])^+$. Let us denote $L_{\tilde{\rho}} - E[L_{\tilde{\rho}}] \text{ and } E[U_{\tilde{\rho}}] - U_{\tilde{\rho}}$, respectively. Also, let $d$ be $L_{\tilde{\rho}} - U_{\tilde{\rho}}$. The true and estimated values of the upper bound and lower bound, and the loss are illustrated in Figure 15; $e_2$ is shown by the thick line segment. We can bound $e_2$ almost surely as follows:

$$e_2 \leq \left((E[U_{\tilde{\rho}}] - U_{\tilde{\rho}}) + (U_{\tilde{\rho}} - U_{\tilde{\rho}}) + (L_{\tilde{\rho}} - E[L_{\tilde{\rho}}])\right)^+ = (x_U - d + x_L)^+. \quad (43)$$

Figure 15: Illustration of the error due to pruning: the thick line segment is the error as a result of pruning the optimal path by mistake.

The expected loss, $E[e_2]$, can be bounded as shown in (46).

$$E[e_2] = E[(x_L + x_U - d)^+] \leq E[(x_L + x_U)^+] \leq E[|x_L + x_U|] \quad (44)$$
$$\leq E[|x_L|] + E[|x_U|] \quad (45)$$
$$\leq 2\delta \sqrt{\frac{\pi}{-\ln(\frac{\epsilon}{2})}}. \quad (46)$$

The first inequality in (44) holds because $d$ is always positive. Inequality (45) results from the triangular inequality, and the last inequality is obtained by replacing $E[|x_L|]$ and $E[|x_U|]$ by the expression in (41).

□

Proof of Proposition 4.3. For an arbitrary initial bay $B$, let $\rho_{\max}$ be the path that maximizes
\( \mathbb{E}[U_\rho] \) among \( \rho_1, \ldots, \rho_m \). We bound \( e_3 \) (the expected loss due to pruning) as follows:

\[
    e_3 \leq P(\text{mistake at } t_1 \cup \cdots \cup \text{mistake at } t_m)(\mathbb{E}[z_{\rho_{\max}}] - \mathbb{E}[z_f])
\]

\[
    \leq m \ P(\text{mistake at } t_1)(\mathbb{E}[U_{\rho_{\max}}] - 0)
\]

\[
    = m \ P(\text{mistake at } t_1)(\mathbb{E}[U_{\rho_{\max}}] - \overline{U}_{\rho_{\max}} + \overline{U}_{\rho_{\max}})
\]

\[
    \leq m \ P(\text{mistake at } t_1)(|x_U| + \overline{U}_{\rho_{\max}}).
\]

Notice that inequality [48] is obtained by using the union bound. Moreover, we replace the probability of mistakes at each step by the maximum probability. Without loss of generality we assume that maximum probability is at \( t_1 \). Also note that we do not know which path is \( \rho_{\max} \) because we have not observed \( \mathbb{E}[U_{\rho_1}], \ldots, \mathbb{E}[U_{\rho_m}] \). Nevertheless, regardless of the path, we can replace \( \mathbb{E}[U_{\rho_{\max}}] - \overline{U}_{\rho_{\max}} \) with \( |x_U| \) (by definition) to obtain [50]. Also, we can bound \( \overline{U}_{\rho_{\max}} \) by \( \overline{U}_{\rho} \).

Now we compute \( P(\text{mistake at } t_1) \). Let us denote \( |x_U| \) and \( |x_L| \) by \( x \) and \( y \), respectively. Also, let \( w = x + y \) and \( f_\rho(w) \) be the PDF of \( w \). Notice that the probability of making a mistake depends on the value of \( d \); the maximum probability corresponds to the stage with the smallest value of \( d \) and can be computed as follows:

\[
    P(\text{mistake at } t_1) = P(x + y - d_{\min} > 0) = \int_{d_{\min}}^{\infty} f_\rho(w) dw = \int_{d_{\min}}^{\infty} \int_{0}^{w} f_X(w - y) f_Y(y) dy dw.
\]

\( f_X(x) \) and \( f_Y(y) \) can be obtained from (11), and the above integral can be bounded as follows:

\[
    \int_{d_{\min}}^{\infty} \int_{0}^{w} f_X(w - y) f_Y(y) = \int_{d_{\min}}^{\infty} \int_{0}^{w} 4\left(\frac{-\ln(\frac{\epsilon}{\delta})}{\delta^2}\right)^2 y(w - y)\left(\frac{\epsilon}{2}\right)^2 \frac{(\epsilon^2 + (w - y)^2)}{\delta^2} dy dw = \left(\frac{\epsilon}{2}\right)^2 \frac{d_{\min}}{\delta} \sqrt{-\ln(\frac{\epsilon}{\delta})} \frac{\pi}{2} \operatorname{Erf}(\frac{d_{\min}}{\sqrt{2\delta}})
\]

\[
    \leq \left(\frac{\epsilon}{2}\right)^2 \frac{d_{\min}}{\delta} \sqrt{-\ln(\frac{\epsilon}{\delta})} \frac{\pi}{2} \left(\frac{\epsilon}{2}\right)^2 \frac{d_{\min}}{\delta}.
\]

From [50] [52] and expression [41] for \( \mathbb{E}[|x_U|] \), it follows that:

\[
    \mathbb{E}[e_3] \leq m \left(\left(\frac{\epsilon}{2}\right)^2 \frac{d_{\min}}{\delta} \sqrt{-\ln(\frac{\epsilon}{\delta})} \frac{\pi}{2} \left(\frac{\epsilon}{2}\right)^2 \frac{d_{\min}}{\delta} \right) \overline{U}_{\rho_{\max}} + \overline{U}_{\rho_{\max}}.
\]

where \( \overline{U}_{\rho_{\max}} = \max(\overline{U}_{\rho_1}, \ldots, \overline{U}_{\rho_m}) \), and \( d_{\min} = \min[d_1, \ldots, d_m] \).