On the Complexity of Container Stowage Planning

The Capacitated Zero-Shift Problem and the Hatch Overstow Problem

Rune Møller Jensen

IT University Technical Report Series **TR-2010-137**

ISSN 1600–6100 **December 2010**

ISSN 1600–6100

ISBN 9788779492301

Copies may be obtained by contacting:

IT University of Copenhagen
Rued Langgaards Vej 7
DK-2300 Copenhagen S
Denmark

Telephone: **+45 72 18 50 00**
Telefax: **+45 72 18 50 01**
Web **`www.itu.dk`**

On the Complexity of Container Stowage Planning

The Capacitated Zero-Shift Problem and the Hatch Overstow Problem

Rune Møller Jensen

Abstract

In this report we examine the complexity of stowage planning problems. A stowage planning problem is to stow containers in stacks such that the number of containers to remove in order to reach containers to retrieve is minimal. We first define and discuss an open problem called the capacitated zero-shift problem, where a set of containers must be placed in a set of stacks with fixed capacity without blocking each other. We then define the hatch overstow problem for container ships. The problem is to place a set of containers on top of hatches of a container vessel avoiding that containers are stowed on hatches that must be accessed. We prove the decision version of this problem to be NP-complete by a reduction from the set cover problem.

1 Introduction

Today, approximately 90 percent of non-bulk cargo worldwide is transported in more than 18 million containers. Containers are constructed such that they can be stored space efficiently directly on top of each other in stacks. Containers are almost always stored in this way both in stationary storage areas such as depots and port terminal yards and moving storage areas such as bays of container ships. Another characteristic of these storage areas is that containers arrive and depart at discrete points in time. For a typical depot and yard, trucks load and unload containers daily, while containers stowed in bays of vessels are loaded and unloaded at different ports. This, and the fact that stacks only can be accessed from the top, complicates the decision of where to stow containers in a storage area. We call this general discrete optimization problem *stowage planning*. Stowage planning is hard because it must be ensured that each container to retrieve from the storage area is at or near top positions of stacks.

A container in a stack stowed above another container that must be retrieved is called *overstowing* and the process of removing an overstowing container is called *shifting*. Thus, the goal of stowage planning is to minimize shifting or equivalently minimize overstowage. This is particularly important in bays of container ships, because shifting requires that the overstowing container is moved from the ship to the terminal yard and back, which is very expensive.

Despite the practical importance of stowage planning, it is still unknown whether a polynomial algorithm exists for stowing a set of containers in a set of stacks with finite capacity without causing overstowage or whether this problem can be shown to be NP-complete. In other words, while the complexity of many problems related to storing data items on electronic devices is well known, we know surprisingly little about the complexity of storing physical items in the common way of container stacks.

In this report, we first in Section 2 formally define this problem as the *Capacitated Zero-Shift Problem* and briefly discuss related problems with known complexity. We then consider the complexity of stowage planning for container ships. A stowage plan is made at each port of call and assigns each container to load in the port to a slot on the vessel. The ship is seldom empty and to avoid overstowage in the current port, on-board containers and containers to load in downstream ports must be taken into account. An efficient decomposition of the problem (e.g., [6, 8]) is first to consider all containers to load in the current port and a number of downstream ports. Rather than deciding specific slots for these containers, they are assigned to bays. In this way, overstowage between containers in individual stacks is ignored, but overstowage can still occur in this model. A bay is separated in a storage area below deck and on deck by a metal lid called a hatch. Since the containers stored on deck rest on the hatch and the hatch must be removed to reach containers below deck, overstowage between containers stored over and under a hatch must be avoided. We formally define this problem in Section 3 as the *Hatch Overstow Problem*. We prove by a reduction from the set cover problem

that this problem is NP-complete. Our definition and NP-completeness proof of this problem is a simplification of the work presented in [1].

2 The Capacitated Zero-Shift Problem

The Capacitated Zero-Shift Problem (CZP) is a decision problem that asks whether a set of containers that must be stored and retrieved at discrete time points can be stowed without shifting in a fixed number of stacks with limited capacity. Formally, an instance of the CZP is a tuple $\langle C, in, out, S, m\rangle$, where C is a finite set of containers, $in(c) \in \mathbb{N}$[1] ($out(c) \in \mathbb{N}$) is the time point that container c must be stowed in (retrieved from) one of the stacks, S is a finite set of stacks, and $m \in \mathbb{N}$ is the maximum number of containers that each stack can hold at any time.

The question is whether the containers can be assigned to the stacks such that no shifting is required to retrieve them. Formally, the question is whether there exists an assignment $A : C \to S$ that is within the stack capacity (i.e., $\forall t \in \mathbb{N}, s \in S \,.\, \big|\{\, c \mid A(c) = s, in(c) \le t < out(c)\}\big| \le m$) and requires no shifting (i.e., $\nexists v, w \,.\, A(v) = A(w) \wedge in(v) < in(w) < out(v) \wedge in(w) < out(v) < out(w)$).

Example 1 Consider a CZP with $C = \{c_1, c_2, c_3, c_4\}$, $in(c_1) = 1$, $in(c_2) = 2$, $in(c_3) = 3$, $in(c_4) = 1$, $out(c_1) = 7$, $out(c_2) = 4$, $out(c_3) = 7$, $out(c_4) = 5$, $S = \{s_1, s_2\}$, and $m = 2$. As depicted in Figure 1, the answer to this CZP is "yes", because the shown assignment is within the stack capacity and requires no shifting.

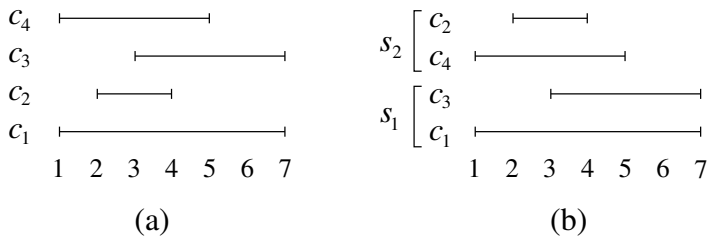

Figure 1: (a) A CZP instance with $S = \{s_1, s_2\}$ and $m = 2$. (b) an zero-shift assignment of containers to stacks showing that it is a "yes"-instance.

The uncapacitated version of the CZP where $m = \infty$ has been shown to be NP-complete for $|S| \ge 4$ by a reduction from coloring of overlap graphs [3] and is known to be polynomial for $|S| < 4$ [2, 7]. For $m = 1$, the CZP is solvable in polynomial time using a minimal cost flow formulation for interval scheduling on identical machines [4]. But even for $m = 2$, neither a polynomial time algorithm nor an NP-completeness proof is known.

3 The Hatch Overstow Problem

The Hatch Overstow Problem (HOP) is a decision problem that asks whether a set of containers that must be loaded and discharged in a set of numbered ports visited in the order $1, 2, \ldots$ can be stowed on a set of hatch covers without causing more than k hatch overstows when the hatches may have to be removed in some ports to access containers stowed below them. Formally, an instance of the HOP is a tuple $\langle C, in, out, H, r, m, k\rangle$, where C is a finite set of containers, $in(c) \in \mathbb{N}$ ($out(c) \in \mathbb{N}$) is the port that container c must be loaded to (unloaded from) one of the hatches, H is a finite set of hatches, $r(h) \in 2^{\mathbb{N}}$ is the set of ports where hatch h must be removed, $m \in \mathbb{N}$ is the maximum number of containers that each hatch can hold at any time, and $k \in \{0, \ldots, |C|\}$ is the maximum number of hatch overstows.

The question is whether the containers can be assigned to the hatches without causing more than k hatch overstows. Formally, the question is whether there exists an assignment $A : C \to H$ that is within the hatch capacity (i.e.,

[1] $\mathbb{N} = \{1, 2, \ldots\}$.

$\forall t \in \mathbb{N}, h \in H \,.\, \big|\{\, c \mid A(c) = h, in(c) \leq t < out(c)\}\big| \leq m$) and has at most k hatch overstows (i.e., $\big|\{\, c \mid \exists p \in r(A(c)) \,.\, in(c) < p < out(c)\}\big| \leq k$).

Example 2 Consider an HOP with $C = \{c_1, c_2, c_3, c_4, c_5\}$, $in(c_1) = 1$, $in(c_2) = 2$, $in(c_3) = 4$, $in(c_4) = 5$, $in(c_5) = 5$, $out(c_1) = 3$, $out(c_2) = 6$, $out(c_3) = 7$, $out(c_4) = 8$, $out(c_5) = 9$, $H = \{h_1, h_2\}$, $r(h_1) = \{2, 6, 8\}$, $r(h_2) = \{4, 9\}$, $m = 2$, and $k = 1$. As depicted in Figure 2 below, the answer to this HOP is "yes".

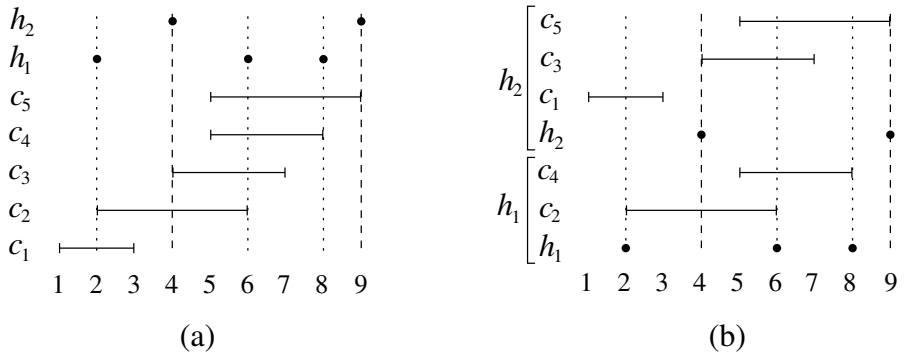

Figure 2: (a) An HOP instance with $m = 2$ and $k = 1$. (b) an assignment of containers to hatches with only one hatch overstow (for h_1) showing that it is a "yes"-instance.

We now prove that the HOP is NP-complete by a reduction from the Set Cover Problem (SCP). Recall that an instance of the SCP is a tuple $\langle S, \mathcal{A}, k' \rangle$, where S is a finite set, $\mathcal{A}$ is a collection of non-empty subsets of S, and $k' \in \{1, \ldots, |\mathcal{A}|\}$. The question is whether $\mathcal{A}$ contains a cover for S of size k' or less, i.e. a subset $\mathcal{A}' \subseteq \mathcal{A}$ with $|\mathcal{A}'| \leq k'$ such that every element of S belongs to at least one member of $\mathcal{A}$.[2]

Example 3 Consider an SCP with $\mathcal{A} = \{a_1, a_2, a_3\}$ where $a_1 = \{e_1, e_3\}$, $a_2 = \{e_3, e_4\}$, and $a_3 = \{e_1, e_2, e_4\}$, $S = \{e_1, e_2, e_3, e_4\}$, $k' = 2$. Clearly this is a "yes"-instance since S can be covered by $\mathcal{A}' = \{a_1, a_3\}$.

Theorem 1 The HOP is NP-complete.

Proof We have HOP $\in$ NP since the assignment A can be used as a certificate that clearly can be checked in polynomial time. We next prove that SCP $\leq_p$ HOP which shows that the HOP is NP-hard. The reduction algorithm begins with an instance $\langle \{e_1, \ldots, e_{|S|}\}, \{a_1, \ldots, a_{|\mathcal{A}|}\}, k' \rangle$ of the SCP. We shall construct a HOP instance $\langle C, in, out, H, r, m, k \rangle$ that has an assignment with no more than k hatch overstows if and only if the SCP instance has a cover with a size no greater than k'. The HOP instance is constructed by reducing the hatch capacity to one and using the containers to represent the elements in S and using the hatches to represent $\mathcal{A}$. The idea is that a container that belongs to a subset can be assigned to a hatch representing it without causing overstowage. To measure the size of these non-overstowing covers, $|\mathcal{A}|$ blocking containers and k' extra hatches are introduced. Formally, we have $C = S \cup \{b_1, \ldots, b_{|\mathcal{A}|}\}$, where $\langle in(e_i), out(e_i) \rangle = \langle 2i-1, 2i+1 \rangle$ for $1 \leq i \leq |S|$ and $\langle in(b_i), out(b_i) \rangle = \langle 1, 2|S|+1 \rangle$ for $1 \leq i \leq |\mathcal{A}|$. Further, $H = \mathcal{A} \cup \{f_1, \ldots, f_k\}$, where $r(a_i) = \{2j \mid e_j \in S \setminus a_i\}$ for $1 \leq i \leq |\mathcal{A}|$ and $r(b_i) = \{2j \mid e_j \in S\}$ for $1 \leq i \leq k$. Finally, we have $m = 1$ and $k = |\mathcal{A}|$. Clearly, this HOP instance can be constructed in polynomial time. As an example of the construction, Figure 3 shows the HOP instance of the SCP defined in Example 3.

We must show that this transformation of SCP into HOP is a reduction. First suppose that the SCP has a cover $\mathcal{A}' = \{a'_1, \ldots, a'_n\}$ where $n \leq k'$. For the corresponding HOP, assign each container representing an element in S to a hatch representing a subset in the cover that includes it. This is possible with $m = 1$ because none of these container

[2]There are slight variations of the SCP in the literature. The one present here differs from SP5 in [5] by requiring that the subsets of $\mathcal{A}$ are non-empty. It is trivial to show that SCP is NP-complete by a reduction from SP5.

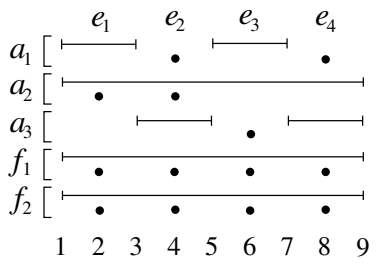

Figure 3: The HOP instance of the SCP defined in Example 3. As in Figure 2(b), we show an assignment that proves the HOP to be a "yes"-instance, since there is no more than 3 hatch overstows.

transports overlap each other. Further, none of these assignments overstow. Then assign $|\mathcal{A}| - n$ blocking containers to the hatches representing subsets that are not included in the cover and assign the remaining n blocking containers to extra hatches. This is possible since no other containers are assigned to these hatches and there are enough extra hatches because $n \leq k'$. Since all the subsets in $\mathcal{A}$ are assumed to be non-empty, we have that all $|\mathcal{A}|$ assignments of blocking containers overstow. The number of overstowing containers, however, is still no greater than k as required.

Conversely, suppose that the HOP has a feasible assignment with $n \leq k$ overstows. Since all blocking containers overstow, we must have $n \geq |\mathcal{A}|$. But since $k = |\mathcal{A}|$, we have $n = k = |\mathcal{A}|$. This means that no element containers overstow, which is only possible if they are assigned to hatches that represent subsets that form a cover $\mathcal{A}'$ of S. The size of this cover can at most be k' because every subset in the cover requires an extra hatch to move a blocking container to and there are only k' of these extra hatches. □

References

[1] M. L. Ajspur, R. M. Jensen, and N. Guilbert. Minimizing lid overstows in master stowage plans for container vessels is np-complete. Technical Report TR-2007-107, IT University of Copenhagen, 2008.

[2] A. Aslidis. Minimizing of overstowage in containership operations. *Operations Research*, 90:457–471, 1990.

[3] M. Avriel, M. Penn, and N. Shpirer. Container ship stowage problem: Complexity and connection to the coloring of circle graphs. *Discrete Applied Mathematics*, 103:271–279, 2000.

[4] K. I. Bouzina and H. Emmons. Interval scheduling on identical machines. *Journal of Global Optimization*, 9:379–393, 1996.

[5] M. R. Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. Freeman, 1979.

[6] J.G. Kang and Y.D. Kim. Stowage Planning in Maritime Container Transportation. *Journal of the Operations Research society*, 53(4):415–426, 2002.

[7] W. Unger. The complexity of coloring circle graphs. In *Proceedings of STACS 92*, volume 577 of *Lecture Notes in Computer Science*, pages 389–400. Springer, 1992.

[8] I. D. Wilson and P. Roach. Container Stowage Planning: A Methodology for Generating Computerised Solutions. *Journal of the Operational Research Society*, 51(11):248–255, 2000.