European Journal of Operational Research 249 (2016) 618–627

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Production, Manufacturing and Logistics

The parallel stack loading problem to minimize blockages

Nils Boysen[*], Simon Emde

Friedrich-Schiller-Universität Jena, Lehrstuhl für Operations Management, Carl-Zeiß-Straße 3, 07743 Jena, Germany

ARTICLE INFO

Article history:
Received 4 February 2015
Accepted 15 September 2015
Available online 28 September 2015

Keywords:
Container storage
Stacking yard
Stack loading
Dynamic programming

ABSTRACT

This paper treats an elementary optimization problem, which arises whenever an inbound stream of items is to be intermediately stored in a given number of parallel stacks, so that blockages during their later retrieval are avoided. A blockage occurs whenever an item to be retrieved earlier is blocked by another item with lower priority stored on top of it in the same stack. Our stack loading problem arises, for instance, if containers arriving by vessel are intermediately stored in a container yard of a port or if, during nighttime, successively arriving wagons are to be parked in multiple parallel dead-end rail tracks of a tram depot. We formalize the resulting basic stack loading problem, investigate its computational complexity, and present suited exact and heuristic solution procedures.

© 2015 Elsevier B.V. and Association of European Operational Research Societies (EURO) within the International Federation of Operational Research Societies (IFORS). All rights reserved.

1. Introduction

The decision of how to intermediately store an incoming stream of items in a storage area consisting of parallel stacks so that they can, later on, be retrieved without excessive effort for relocating items is a widespread decision problem with quite a few real-world applications. This paper formulates a very basic core problem, denoted as the parallel stack loading problem (PSLP), which can be characterized as follows:

Consider a given inbound stream of items that successively arrive at some storage yard. The incoming items, e.g., containers, boxes, or pallets, are to be intermediately stored in multiple parallel stacks. In each stack items can be stockpiled up to a given maximum height. Sometime after their intermediate storage, these items are to be retrieved again, so that each item j is assigned a weight p_j that indicates its priority of retrieval. For instance, if items i and j are assigned priority values $p_i = 3$ and $p_j = 1$, respectively, then item j is to be retrieved before i. Thus, storing i on top of j in the same stack would lead to a blockage (also denoted as mis-overlay or overstowage, see (Lehnfeld & Knust, 2014)), i.e., item i would have to be removed first in order to access item j. The PSLP aims to store the inbound stream of items in a given number of parallel stacks, such that the maximum stacking height is not violated and the number of blockages is minimized.

Consider the example depicted in Fig. 1, where five items are to be successively stored in a storage area consisting of two stacks each stackable up to three tiers high. If these five items, whose priority values are indicated by the numbers within the blocks, are stored as is depicted in the middle of the figure this solution to the PSLP results in two blockages. The item with priority value 5 blocks that with value 4 which in turn blocks the lowermost item of the stack. If we take a look at the later retrieval process of the items, which is, however, not explicitly modeled as a part of our PSLP, and presuppose that the items are to be retrieved in increasing sequence of their priority weights, then the items with weights 5 and 4 need to be relocated first in order to access the lowermost item with weight 3. Thus, by reducing the number of blockages during the loading problem we actually aim to minimize the effort for relocations during the retrieval process. Note that in our example the number of blockages equals the number of relocations, but this does not have to be the case. Minimizing the number of relocations during the retrieval process is known as the blocks relocation problem, which was shown to be NP-hard (Caserta, Schwarze, & Voss, 2012). To avoid the simultaneous solution of the complex retrieval problem already during the loading phase, it is a widely applied approach to utilize a surrogate objective for the true number of relocations (see Lehnfeld & Knust, 2014). We discuss this matter in more detail in Section 5.

1.1. Applications and literature review

Our PSLP occurs as a core problem in quite a few logistics applications. Three of them are depicted in Fig. 2 and briefly summarized in the following:

- A natural form of stack-based storage is *ground storage*, where items are stockpiled on top of each other. These items can be containers in stacking yards of ports (e.g., Caserta, Schwarze, & Voss, 2011a; Lehnfeld & Knust, 2014), steel slabs

[*] Corresponding author. Tel.: +49 3641 9 43100.
E-mail addresses: nils.boysen@uni-jena.de (N. Boysen), simon.emde@uni-jena.de (S. Emde).
URL: http://www.om.uni-jena.de/ (N. Boysen), http://www.om.uni-jena.de/ (S. Emde)

http://dx.doi.org/10.1016/j.ejor.2015.09.033
0377-2217/© 2015 Elsevier B.V. and Association of European Operational Research Societies (EURO) within the International Federation of Operational Research Societies (IFORS). All rights reserved.

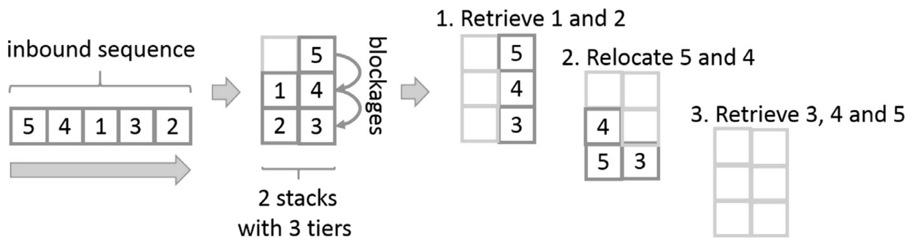

Fig. 1. Example for the PSLP.

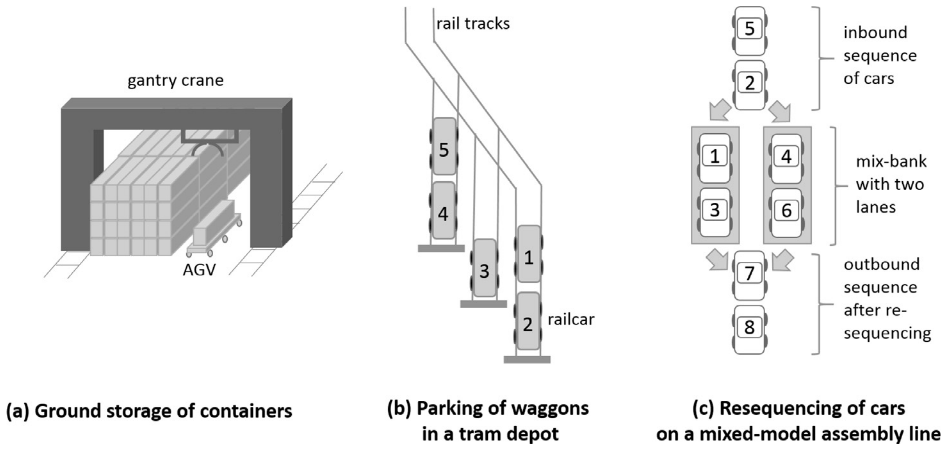

Fig. 2. Applications of the stack loading problem.

intermediately stored in the iron and steel industry (e.g., Zäpfel & Wasner, 2006; Tang, Zhao, & Liu, 2012), and pallets stockpiled in a warehouse (e.g., Nishi & Konishi, 2010). A large body of literature has accumulated over the past years in this field, which is reviewed in detail by the recent survey papers of Caserta et al. (2011a) and Lehnfeld and Knust (2014). However, most of the work summarized there addresses the premarshalling problem (e.g., Bortfeldt & Forster, 2012), where idle time of cranes is utilized to reshuffle existing storage compositions, or the unloading problem (also denoted as the block relocation problem (Caserta, Voss, & Sniedovich, 2011b; Kim & Hong, 2006)), where a sequence of items is to be retrieved from the stacks, such that the number of relocation moves for removing blocking boxes is minimized. The loading of items into stacks has mainly been treated in the container yard context, where the problem faces a lot of uncertainty. Typically, both the arrival sequence of containers and the retrieval sequence (from which we derive weights p_j) are susceptible to forecast errors. Thus, existing research either focuses on simple online stacking rules evaluated by simulation (see Dekker, Voogd, & Asperen, 2006; Borgman, van Asperen, & Dekker, 2010) or minimizes the expected number of reshuffles, e.g., by considering probability distributions of different weight classes, which mainly influence the retrieval time (see Kim, Park, & Ryu, 2000; Kang, Ryu, & Kim, 2006b; Zhang, Chen, Shi, & Zheng, 2010; Gharehgozli, Yu, de Koster, & Udding, 2014). In this paper, we consider deterministic information on both the arrival sequence and the priority weights, which at least “can be very useful as a benchmark” (Borgman et al., 2010). However, we postpone a deeper debate on the applicability of our PSLP to Section 5. A deterministic loading problem, which comes pretty close to ours and is inspired by the successive storage of steel plates, has been introduced by Kim, Koo, and Sambhajirao (2011). Their parallel stacks, however, have no capacity limits, which may be a pardonable relaxation in the steel industry where only thin plates are stacked, but limits the transferability to other applications such as container yards. Moreover, they consider an alternative objective function. They assume that blocking items are restacked into their origin stack once a blocked item is removed, so that they minimize the total number of boxes with lower priority stacked on top of each box. They also treat an alternative objective where blocking items are removed to some alternative stack (not being part of the problem) and not reinserted at all. For both objectives they provide mixed-integer programs and present straightforward heuristics, i.e., a rule-based start heuristic and an improvement heuristic based on two-opt.

- *Tram wagons:* If we think of our items as railcars or buses that successively arrive at a depot consisting of dead-end tracks (stacks) to be parked overnight, the relation to our PSLP is readily available. The sequence of departure at the next morning is decoded by our priority values p_j and it seems a valid objective for the parking process to store the wagons on the tracks, such that the effort for maneuvering them into the right departure sequence in the morning is minimized. However, existing research focuses on the release problem from the stacks and considers interchangeable wagons of the same type (Blasum, Bussieck, Hochstättler, Moll, Scheel, & Winter, 1999), treats real-time dispatching Winter and Zimmermann (2000), or integrates further real-world aspects (e.g., Freling, Lentink, Kroon, & Huisman, 2005; Jacobsen & Pisinger, 2011) such as differently sized wagons and tracks, complex time timetable constraints, and tracks accessible from both directions. A more detailed review on the literature on sorting processes in shunting yards is provided by the recent survey paper of Boysen, Fliedner, Jaehn, and Pesch (2012a).
- Another application of our stacking problem is the *resequencing of assembly lines* in the automotive industry (see the survey paper of Boysen, Scholl, & Wopperer (2012b)). Initially, the production sequence of cars is planned about three to four days before production starts and communicated to the part suppliers. Suppliers deliver the parts just-in-time to the respective assembly stations where parts remain in storage sorted right in the sequence of their later assembly. Unfortunately, the production sequence often gets stirred during the assembly process. Especially the

Download English Version:

https://daneshyari.com/en/article/480734

Download Persian Version:

https://daneshyari.com/article/480734

Daneshyari.com