Computers & Industrial Engineering 87 (2015) 410–424

Contents lists available at ScienceDirect

Computers & Industrial Engineering

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

A heuristic algorithm based on an improvement strategy to exploit idle time periods for the Stacking Problem

Christopher Expósito-Izquierdo *, Eduardo Lalla-Ruiz, Jesica de Armas, Belén Melián-Batista, J. Marcos Moreno-Vega

Department of Computer and Systems Engineering, University of La Laguna, Spain

ARTICLE INFO

Article history:
Received 16 October 2014
Received in revised form 22 May 2015
Accepted 23 May 2015
Available online 6 June 2015

Keywords:
Stacking Problem
Relocation movement
Stacking crane
Heuristic

ABSTRACT

In this paper, we address the Stacking Problem. Its objective is to determine the sequence of movements carried out by a stacking crane to store and retrieve a set of homogeneous blocks in a two-dimensional storage during a well-defined planning horizon in such a way that the number of relocation movements is minimized. We propose a heuristic algorithm that determines the target stack of each incoming block and those placed above the next to retrieve. Our heuristic algorithm also exploits the time periods in which the stacking crane is idle with the goal of improving its performance. The computational experiments reveal that our heuristic algorithm overcomes those approaches found in the related literature in terms of number of crane relocation movements. Its high performance and short computational times encourage its application in real-world environments.

1. Introduction

The Stacking Problem (SP) is an optimization problem in which the flow of homogeneous blocks (i.e., goods packaged into robust load units such as boxes, containers, and roll-cages) in a two-dimensional storage, composed of stacks with a maximum stacking height, is optimized (Rei & Pedroso, 2012). That is, it determines the position in which each incoming block must be stored during its release time and the movements to carry out for its removal during its retrieval time. The objective of the SP is to minimize the number of movements required to store and retrieve all the blocks by means of a stacking crane.

The main shortcoming associated with the stacking arises from the Last In First Out (LIFO) policy (Carlo, Vis, & Roodbergen, 2014). That is, only one crane movement is required during the block storage whenever its target slot is at the top of a stack in the two-dimensional storage. Similarly, one crane movement is required to retrieve a block whenever is currently placed at the top of a stack. Otherwise, relocation movements have to be performed in order to access the slot in which a target block is going to be stored or retrieved from Expósito-Izquierdo, Melián-Batista, and Moreno-Vega (2014).

The block relocation movements are unproductive, and consequently give rise to a low performance of the stacking crane. Minimizing the number of relocation movements leads to increase the storage productivity, decrease the working costs, reduce the machinery wear, etc. With the goal of minimizing the number of block relocation movements during a given planning horizon, appropriate planning strategies must be designed (Choe, Park, Oh, Kang, & Ryu, 2011). These planning strategies aim to place the incoming blocks in storage locations in which they will be accessible for their future retrieval.

The main contribution of this paper is to develop a heuristic algorithm aimed at solving the SP efficiently. This approach determines the target stack of each block to be stored and those placed above the next one to retrieve. Additionally, it exploits those time periods in which the stacking crane is idle in order to improve its performance.

The remainder of this paper is organized as follows. Section 2 describes the SP. Afterwards, Section 3 briefly reviews the main works from the related literature. Section 4 presents a heuristic algorithm for solving the SP. Subsequently, Section 5 discusses the computational experiments carried out in this work. Finally, Section 6 draws the concluding remarks extracted from the work and indicates several promising directions for further research.

2. Stacking Problem

The Stacking Problem (SP) aims to determine the sequence of movements carried out by a stacking crane to store and retrieve

* Corresponding author.
E-mail addresses: cexposit@ull.es (C. Expósito-Izquierdo), elalla@ull.es (E. Lalla-Ruiz), jdearmas@ull.es (J. de Armas), mbmelian@ull.es (B. Melián-Batista), jmmoreno@ull.es (J.M. Moreno-Vega).

http://dx.doi.org/10.1016/j.cie.2015.05.030
0360-8352/

a set of homogeneous blocks, $C = \{1, 2, \ldots, nC\}$, in a two-dimensional storage during a well-defined planning horizon, denoted as H (e.g., a working shift), in such a way that the number of relocation movements is minimized.

The two-dimensional storage in the SP is composed of $S = \{1, 2, \ldots, nS\}$ stacks and $T = \{1, 2, \ldots, nT\}$ tiers. The capacity of the two-dimensional storage is, therefore, defined as $M = nS \times nT$. Each slot within the two-dimensional storage can store a maximum of one block in each time period. The blocks can be placed either on the ground (tier 1) or on top of another block. It is worth mentioning that crushability or stability constraints of the blocks are not considered in the SP.

Moreover, each block $c \in C$ must be stored in the two-dimensional storage during its delivery time, $d(c) \geqslant 0$, and retrieved during its retrieval time, $d(c) < r(c) < H$. The stack in which the block c is currently placed within the two-dimensional storage is denoted as $s(c) \in S$ whereas its tier is denoted as $t(c) \in T$. The set of blocks currently placed above a given block $c \in C$ in the two-dimensional storage is denoted as $O(c)$ and formally defined as follows:

$$O(c) = \{c' \in C \mid (s(c') = s(c)) \wedge (t(c') > t(c))\}. \quad (1)$$

Furthermore, the number of blocks placed in the stack $s \in S$ is denoted as $h(s)$, whereas the earliest retrieval time of a block placed in s is defined as follows:

$$min(s) = \min\{r(c) \mid (c \in C) \wedge (s(c) = s)\}. \quad (2)$$

The set of stacks in the two-dimensional storage in which an incoming block can be placed is composed of those stacks in which there is at least one empty slot. This can defined as follows:

$$\Phi = \{s \in S \mid h(s) < nT\}. \quad (3)$$

Fig. 1 depicts an illustrative example of the state of a two-dimensional storage composed of $nS = 8$ stacks and $nT = 5$ tiers. At this point, the two-dimensional storage contains 22 blocks. In this case, stack 4 has no empty slots (i.e., $h(4) = nT$), in such a way that, no incoming blocks can be placed on it. According to Eq. (3), $\Phi = S \setminus \{4\}$ in this example. Similarly, stack 7 is empty (i.e., $h(7) = 0$), and then a maximum of nT blocks can be placed on it. Furthermore, each block in the example contains a number indicating its retrieval time. For instance, block placed in stack 8 and tier 4 should be retrieved in time period 96. As can be checked, the example is contextualized in time period $t = 16$. This means that:

- Those blocks with retrieval time before $t = 16$ have been already removed from the storage. In this case, at least four blocks have been already retrieved (named *outgoing blocks*) in time periods 3, 5, 9, and 15, respectively.
- Some blocks arrive at the storage over time (named *incoming blocks*). A release time is associated with each incoming block. For instance, the next incoming block is released in time period 17 and should be retrieved from the two-dimensional storage in time period 98. Later, a block to be retrieved in time period 35 is released in time period 24, and so forth.

A general classification of the blocks according to their retrieval times and the slots where they are placed has been proposed by Expósito-Izquierdo et al. (2014) for a closely related problem called Blocks Relocation Problem (Forster & Bortfeldt, 2012). Firstly, a non-located block is that one placed at a higher tier than another one in the same stack with an earlier retrieval time. See shaded blocks in Fig. 1. Let $\Omega(s)$ be the set composed of non-located blocks in stack s, defined as follows:

$$\Omega(s) = \{c \in C | (s(c) = s) \wedge \exists c' : (s(c') = s) \wedge (t(c') < t(c)) \wedge (r(c') < r(c))\}, \quad \forall\, s \in S. \quad (4)$$

The set composed of non-located blocks in the two-dimensional storage is thus defined as follows:

$$\Omega = \bigcup_{s \in S} \Omega(s). \quad (5)$$

Moreover, a well-located block is that one not placed above any other block with an earlier retrieval time in the same stack. Examples of well-located blocks are those with white background in Fig. 1. The set of well-located blocks is formally defined as follows:

$$\Upsilon(s) = \{c \in C \mid (s(c) = s) \wedge (c \notin \Omega(s))\}, \quad \forall\, s \in S. \quad (6)$$

The set of stacks composed of only well-located blocks is formally defined as follows:

$$\chi = \{s \in S \mid !\exists\, c \in C : (s(c) = s) \wedge (c \notin \Upsilon(s))\}. \quad (7)$$

The feasible movements to perform by the stacking crane are described as follows:

- *Release movement*. The next incoming block is stored at the top of one of the stacks in Φ (see Eq. (3)) during its release time. Once a release movement is carried out, the number of blocks increases in one unit.
- *Retrieval movement*. The next outgoing block is removed from the top of its stack during its retrieval time. Once a retrieval movement is carried out, the number of blocks decreases in one unit. In this work, we adopt a retrieval-favoured approach. Using this method, retrieval movements are firstly carried out in each time period of the planning horizon.

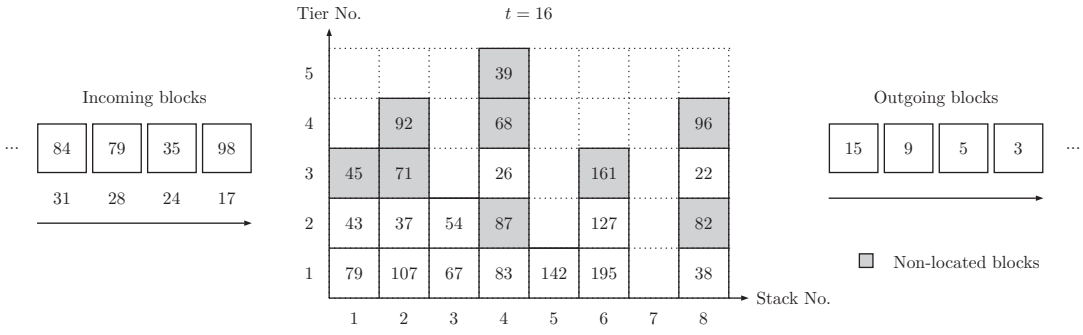

Fig. 1. Example of the Stacking Problem with $nS = 8$ stacks and $nT = 5$ tiers.