Computers and Operations Research 99 (2018) 206–217

Contents lists available at ScienceDirect

Computers and Operations Research

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

An exact algorithm for the Blocks Relocation Problem with new lower bounds

Kent E. Yucra Quispe[a], Carla N. Lintzmayer[b], Eduardo C. Xavier[a,*]

[a] Institute of Computing, University of Campinas, Brazil
[b] Center for Mathematics, Computation and Cognition, Federal University of ABC, Brazil

ARTICLE INFO

Article history:
Received 4 December 2017
Revised 21 June 2018
Accepted 24 June 2018
Available online 7 July 2018

Keywords:
Blocks Relocation Problem
Container Relocation Problem
Exact algorithms
Lower bounds

ABSTRACT

The Blocks Relocation Problem is an important problem in storage systems. An input instance for it consists of a set of blocks distributed in stacks where each block is identified by a retrieval number and each stack has a same maximum height limit. The objective is to retrieve all the blocks respecting their retrieval order and performing the minimum number of relocations. Only blocks at the top of a stack can be moved: either a block is retrieved, if it has the highest retrieval priority among the stacked blocks, or it is relocated to the top of another stack. Solving this problem is critical in storage systems because it saves operational time and resources. In this paper, we present two new lower bounds for the number of relocations of an optimal solution. We implemented an exact iterative deepening A* algorithm using these new proposed lower bounds and other well-known lower bounds from the literature. We performed several computational experiments to show that the new lower bounds improve the performance of the exact algorithm, solving to optimality more instances than when using other lower bounds when given the same amount of time.

1. Introduction

This paper is focused on the *Blocks Relocation Problem (BRP)*, also known as Container Relocation Problem in the literature, which generally emerges from storage systems. In storage systems, there are several types of items to be stored, such as containers and pallets. We will refer to those items as *blocks* throughout the paper. We consider that the blocks are stored in a series of stacks, where at each stack one block is above another or at the bottom of the stack. This is called the *stacking area*, which is the most common type of storage system of containers. This type of storage only allows one to access the top block of a stack, which can be *relocated* to the top of another stack or can be removed and placed outside the stacking area, a move called *retrieval*.

Consider a container terminal where blocks have to be retrieved from the stacking area and loaded onto trucks, following a given precedence order called *retrieval priority*. The precedence of these items may be motivated by several factors, such as the arriving order of container transportation trucks or the delivery order of containers in a ship. Given a stacking area and the precedence order of the blocks, the objective of BRP is to minimize the number of relocations in order to retrieve all blocks. In this paper, we present two new lower bounds for the minimum number of relocations, which are used in an exact iterative deepening A* algorithm.

Kim and Hong (2006) proposed two variations for the BRP. In the first one, each block has a unique retrieval priority, and in the second one two or more blocks can have the same retrieval priority. They proposed a branch and bound algorithm and a fast heuristic rule to determine the destination stack of each block trying to reduce the number of relocations in the future. This rule can be used in real time by any search procedure. The authors also introduced the concept of *blocking block*, which is a block that is preventing the retrieval of another block that is below it in the same stack. This concept is used as a lower bound.

Caserta et al. (2009) proposed a binary representation of the stacking area, which simplifies the transition from the current state of the stacking area to a state generated by a relocation or retrieval. They developed a lookahead mechanism that is adapted to this binary representation.

Lee and Lee (2010) introduced a new variant of the problem where each relocation has a cost equal to the distance between the two stacks over which the relocation is performed. First, they presented a simple heuristic to create a feasible solution. Then, they tried to reduce the number of relocations by solving a binary linear

* Corresponding author.
E-mail addresses: ra164889@students.ic.unicamp.br (K.E.Y. Quispe), carla.negri@ufabc.edu.br (C.N. Lintzmayer), eduardo@ic.unicamp.br, eduardo@ic.unicamp.br (E.C. Xavier).

https://doi.org/10.1016/j.cor.2018.06.021
0305-0548/

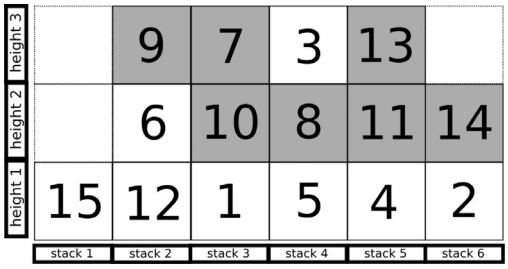

Fig. 1. An instance for the BRP with $S = 6$, $H = 3$, and $N = 15$.

integer program. Finally, they tried to reduce the cost of relocations by solving a mixed integer linear program.

Caserta and Voß (2009) presented a new heuristic, named corridor method, to the *pre-marshalling problem*. In the pre-marshalling problem, one is given an initial stacking area and the objective is to generate another configuration of it by performing only relocations such that the resulting stacking area does not contain blocking blocks. In another work, Caserta et al. (2011) applied the corridor method to the BRP without limiting the height of the stacks.

The BRP was shown to be NP-Hard by Caserta et al. (2012). They also presented a binary linear programming model that generates solutions for small instances of the problem. To break this limitation, realistic assumptions were introduced, which allowed them to create a new binary linear programming model and a new heuristic that was able to get good solutions for medium-sized instances.

Jovanovic and Voß (2014) proposed a new heuristic approach to the BRP, which considers not only the current block to be relocated but also the next block to be relocated. They named this heuristic by Min-Max. This heuristic reduced by 5% the number of relocations on average when compared to solutions generated by another heuristic proposed by Caserta et al. (2012).

Expósito-Izquierdo et al. (2014) presented several exact algorithms based on the A* search algorithm to solve the BRP. They found 17 optimal solutions from 45 instances that were considered by Caserta et al. (2012). They also presented a heuristic to find high-quality solutions within a short computational time.

Tanaka and Takii (2016) presented a new lower bound to the number of relocations based on previous lower bounds created by Kim and Hong (2006) and Zhu et al. (2012). The use of this new lower bound on an exact algorithm resulted in 1.848% more optimal instances being found than when using these previous lower bounds from the literature.

The use of Pattern Databases (PDBs) in exact algorithms has been effectively applied to solve challenging problems such as 16-puzzle (Culberson and Schaeffer, 1998) and Rubik's cube (Korf, 1997). For the BRP, it was first considered by Ku and Arthanari (2016), which use the concept of abstract states and PDB to shorten the exploration of the search space. They compared this method with other exact algorithms from the literature, showing that it outperforms the other methods regarding the time to find optimal solutions.

Contributions: In this paper, we present two new lower bounds for the BRP, where one is purely combinatorial and is denoted by LB-LIS. The other one derives from the creation of pattern databases (PDB) (Felner et al., 2004) and is denoted by LB-PDB. We use the two lower bounds in a general exact iterative deepening A* algorithm. We also explore, in this exact algorithm, the use of pattern databases as a way of memorizing part of the search space, in the same way as was done by Ku and Arthanari (2016). The exact algorithm, when using LB-LIS as a lower bound, found 2% more optimal instances than when using other lower bounds from the literature. The improved exact algorithm, which uses PDBs to reduce the search space, together with LB-LIS, resulted in 4.1% more instances being solved to optimality (5% if we consider the hardest instances).

This paper is organized as follows. In Section 2 we present a formal description of the problem while in Section 3 we describe the structure of the exact algorithm we use in the computational experiments. In Section 4 we present previous lower bounds found in the literature and a new one that we propose. In Section 5 we present the concept of pattern databases, which is also used to induce a new lower bound. In Section 6 we present the computational results. Finally, in Section 7 we present our conclusions and some final remarks.

2. Problem description

In the BRP we are given N blocks $b_1, b_2, \ldots, b_N$ distributed in a stacking area consisting of S stacks $s_1, s_2, \ldots, s_S$, with a maximum height of H for each stack. The height of a stack s_x, denoted by $height(s_x)$, is the number of blocks stacked on it. Thus, in any possible configuration of the stacking area we must have $height(s_x) \leq H$ for all $1 \leq x \leq S$. Each block b_i, for $1 \leq i \leq N$, has a *retrieval priority* defined as i, which indicates its retrieval order. Therefore, the lowest value means the highest priority, so b_1 is the first block to be retrieved.

We denote by t the block with the highest retrieval priority in the stacking area at any given moment, also called *target block* of the stacking area, or target block of the instance. We call *target stack* the stack containing block t. Similarly, we denote by t_x the block with the highest retrieval priority in stack s_x at any given moment, also called *target block of stack* s_x.

Since we only have access to the block at the top of any stack, there are only two available moves. A *relocation* is a move $s_x \Rightarrow s_{x'}$ that takes the top block of stack s_x and put it at the top of stack $s_{x'}$, where it must be valid that $height(s_{x'}) < H$ before the relocation. Also, in the variation that we are considering, a relocation from s_x is allowed only if the target block t is in s_x, so that we can retrieve t later, after relocating all blocks above it. The BRP problem under this restriction is still NP-hard and this assumption is used in several works from the literature (Caserta et al., 2012). A *retrieval* is a move $s_x \Rightarrow s_0$ that removes the top block of stack s_x from the stacking area if such block is t (s_0 is an artificial stack).

An instance of the BRP consists of the dimensions of the stacking area, S and H, the number N of blocks stored, and an initial configuration of these blocks. A solution to an instance of the BRP is a sequence of relocations and retrievals that clears the initial stacking area. Note that the number of retrievals is constant and equal to N, so the goal is to retrieve all the blocks respecting their retrieval priorities using the minimum number of relocations. See Fig. 1 for an example where we have $t = 1$ and we can only relocate blocks 7 and 10, in this order, before retrieving block 1. After relocating blocks 7 and 10, we retrieve block 1, and then the new target block becomes $t = 2$. A possible sequence of relocations to retrieve blocks 1, 2, 3, and 4 is $\mathcal{S} = \{(s_3 \Rightarrow s_1), (s_3 \Rightarrow s_1), (s_3 \Rightarrow s_0), (s_6 \Rightarrow s_3), (s_6 \Rightarrow s_0), (s_4 \Rightarrow s_0), (s_5 \Rightarrow s_3), (s_5 \Rightarrow s_3), (s_5 \Rightarrow s_0)\}$.

Let b be a block in some stack s_x and let a be some block that is placed below b in the same stack, i.e., the height of b in s_x is greater than the height of a. If $b > a$, then b is called a *blocking block*. In Fig. 1, blocks 7, 8, 9, 10, 11, 13, and 14 are blocking blocks.

3. Iterative deepening A* algorithm

In this section we describe the exact algorithm we used in our computational experiments, which is the Iterative Deepening A* (IDA*) algorithm. The IDA* was first introduced by Korf (1985) in 1985 as a general search method, and has been applied to several