Contents lists available at ScienceDirect

Computers & Operations Research

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

A beam search approach to the container loading problem

I. Araya [*], M.-C. Riff

Departamento de Informática, Universidad Técnica Federico Santa María, Av. España 1680, Valparaíso, Chile

ARTICLE INFO

Available online 16 September 2013

Keywords:
Beam search
Container loading problem
Load planing
Combinatorial optimization
Constructive algorithms

ABSTRACT

The *single container loading problem* is a three-dimensional packing problem in which a container has to be filled with a set of boxes. The objective is to maximize the space utilization of the container. This problem has wide applications in the logistics industry. In this work, a new constructive approach to this problem is introduced. The approach uses a beam search strategy. This strategy can be viewed as a variant of the branch-and-bound search that only expands the most promising nodes at each level of the search tree. The approach is compared with state-of-the-art algorithms using 16 well-known sets of benchmark instances. Results show that the new approach outperforms all the others for each set of instances.

1. Introduction

The single container loading problem (CLP) is a three-dimensional packing problem in which a large parallelepiped or *container* has to be filled with smaller parallelepipeds or *boxes*. The objective is to maximize the total volume of loaded boxes. Depending on the problem, the rotation of boxes may be freely allowed, restricted to certain orientations, or prohibited. Problem instances are commonly classified in *strongly heterogeneous* (many types of boxes) and *weakly heterogeneous* (few types of boxes) [1]. Under the improved typology proposed by Wascher et al. [2], the weakly heterogeneous CLP is classified as a three-dimensional rectangular single large object placement problem (3D SLOOP), while the strongly heterogeneous CLP is classified as a single knapsack problem (3D SKP).

The CLP is NP-hard [3]. Few exact algorithms have been proposed to solve it. Fekete and Schepers [4] present a general framework for multiple dimensional packing problems. Martello et al. [5] develop an exact branch-and-bound method for the CLP. Nowadays, heuristics and meta-heuristics are the only viable alternative to find near-optimal packings when time is crucial. A compacted and not necessarily disjoint classification of these methods is proposed by Zhu et al. [6]. They group packing approaches, according to the way in which the loading plans are generated, in three categories: constructive, divide-and-conquer and local-search. *Constructive* approaches generate solutions by loading boxes into the container until no more boxes can be loaded. Most of the approaches belong to this group. For instance, the heuristic of Bischoff and Ratcliff [7] is a constructive method that fills the container with stacks arranged on the floor of the container. The tree search approach of Eley [8], the tabu search method of Bortfeldt [9], the hybrid simulated annealing and tabu search of Mack et al. [10] and the tree-search-based method of Zhu et al. [6] fill the container with cuboid blocks of boxes. *Divide-and-conquer* methods divide the container into subcontainers. Then, they recursively solve the resultant smaller problems before recombining them into a complete solution. Examples in this category include the works of Chien and Wu [11], Lins et al. [12] and Morabito and Arenales [13]. *Local search* methods start with a complete loading plan, then apply neighborhood operators to generate new loading plans. The works of Gehring and Bortfeldt [14], Parreño et al. [15] and Mack et al. [10] belong to this category.

Nowadays, the most successful approaches for CLP are the block-building-based ones. They are constructive methods which use blocks, instead of boxes, to construct solutions. A block is a subset of boxes that is placed compactly inside its minimum bounding cuboid. They are classified into two types. *Simple blocks* are composed of only one type of boxes. *General blocks* can be composed of multiple types of boxes and/or a set of boxes of the same dimensions placed using different orientations. Stacks and layers of boxes can be seen as particular kinds of blocks. The most successful block-building approaches use incomplete search tree methods to select, at each step, which block to place [4,6,16]. They use general blocks which are generated by combining the existing boxes and blocks in pairs repeatedly.

Zhu et al. [6] propose an analytic framework to describe block-building-based approaches. The framework is based on six common elements present in most of these approaches: (K1) the representation of the free space in the container; (K2) the mechanism to generate blocks; (K3) the heuristic used to rank free spaces; (K4) the heuristic used to rank boxes; (K5) the heuristic used to decide how a selected block is placed in the

* Corresponding author. Tel.: +56 322944516.
E-mail addresses: iaraya@inf.utfsm.cl (I. Araya), mcriff@inf.utfsm.cl (M.-C. Riff).

0305-0548/$ - see front matter © 2013 Elsevier Ltd. All rights reserved.
http://dx.doi.org/10.1016/j.cor.2013.09.003

selected free space and (K6) the overall search strategy. They also proposed G2LA, a greedy-based approach, taking key elements from other approaches and implementing a new search-tree-based method for selecting which block to place at each step. When G2LA has been proposed, it outperformed the best single-threaded approaches.

A similar idea is followed in this work. The proposed approach takes most of the key elements from the G2LA algorithm but replacing the greedy-based search strategy by a beam-search-based one. The complexity of the new approach is similar to G2LA.

The remainder of the paper is organized as follows. In Section 2, the single container loading problem is defined, Section 3 describes the new beam-search-based approach, Section 4 reports the computational experiments, Section 5 discuss the relation of our approach to the state-of-art ones and Section 6 concludes the work and gives ideas about the future work.

2. The single container loading problem

The single container loading problem may be stated as follows. Given a container Γ of dimensions L, W, H and a set of boxes $C=\{c_1,c_2,\ldots,c_N\}$, the goal of the problem is to place boxes into the container in order to maximize the total volume of loaded boxes. Boxes do not overlap with any other box and lie completely inside the container with their faces parallel to the container faces.

Formally, the objective may be stated as follows:

$$\max \sum_{i=1}^{N} p_i * V(c_i), \tag{1}$$

where $V(c_i)$ corresponds to the volume of the box c_i and p_i is a boolean variable indicating if the box c_i is placed into the container.

Let $l_0(c_i)$, $w_0(c_i)$ and $h_0(c_i)$ be respectively the length, width and height of a box c_i in its *original orientation*. A box can be placed into the container in any orientation which maintains its faces parallel to the container faces. Let l_i, w_i and h_i be respectively the auxiliary variables indicating the length, width and height of the box arranged in one of its six orientation variants inside the container. The following constraints must be satisfied for all $i=1..N$:

$$\begin{aligned}
&(l_i,w_i,h_i)=(l_0(c_i),w_0(c_i),h_0(c_i))\vee\\
&(l_i,w_i,h_i)=(l_0(c_i),h_0(c_i),w_0(c_i))\vee\\
&(l_i,w_i,h_i)=(w_0(c_i),l_0(c_i),h_0(c_i))\vee\\
&(l_i,w_i,h_i)=(w_0(c_i),h_0(c_i),l_0(c_i))\vee\\
&(l_i,w_i,h_i)=(h_0(c_i),l_0(c_i),w_0(c_i))\vee\\
&(l_i,w_i,h_i)=(h_0(c_i),w_0(c_i),l_0(c_i))
\end{aligned}$$

Additional constraints, taken from the large number of constraints found in practice (see [7]), are also considered:

- *Orientation constraint:* For each box, the number of allowed orientations is restricted (e.g., the top side of a refrigerator must be always on top).
- *Stability constraint (optional):* To guarantee load stability, the bottom sides of all boxes not placed directly on the container floor must be completely supported by the top sides of one or more boxes. When this constraint is imposed, we call the resultant problem variant: *single container loading problem with full support* (CLP-FS)

Let $(x_i,y_i,z_i)\in\{0,1,\ldots,L-l_i\}\times\{0,1,\ldots,W-w_i\}\times\{0,1,\ldots,H-h_i\}$ be the location of the lowest-leftest-and-deepest corner of the box c_i into the container. The following constraint, indicating that a box c_i does not overlap with any other box c_j in the container, must be satisfied for all $i=1..N$:

$$\neg p_i \vee \neg p_j \vee (x_i+l_i\le x_j)\vee(y_i+w_i\le y_j)\vee(z_i+h_i\le z_j) \quad \forall j=1..N, j\ne i \tag{2}$$

3. BSG-CLP: a beam-search-based approach to the CLP

In this section a new block-building approach is introduced. BSG-CLP is a *B*eam-*S*earch-based algorithm that uses a *G*reedy-based function to evaluate states in the search graph. Beam search can be viewed as an adaptation of the branch-and-bound search that expands only a subset of the most promising nodes at each level of the search graph (see [17]).

3.1. Representation

BSG-CLP works by exploring the search space to find a path from some initial state to some terminal state. Intermediate states (or partial solutions) correspond to partial loading plans and consist of three components: a list R of overlapping cuboids which represents the residual space in the container, a list of the remaining boxes C, and a list of blocks B.

A block $b\in B$ is a cuboid containing a set of boxes of C *efficiently placed*, i.e., these boxes use a higher percentage of the cuboid (e.g., $>98\%$). The boxes do not overlap with any other box, satisfy the orientation constraint and lie completely inside b. The block orientation is fixed.

In the initial state, the list R contains only one cuboid: the container Γ; the list C corresponds to the initial list of boxes given by the problem and B is given by a preprocessing method detailed in Section 3.2. A terminal state (or complete solution) corresponds to a loading plan in which no more boxes can be added. In other words, any box in C fits a cuboid in R.

3.1.1. State transition

Consider a nonterminal state consisting of a list of cuboids $R=\{r_1,r_2,\ldots,r_p\}$, a list of remaining boxes C and a list of blocks $B=\{b_1,b_2,\ldots,b_o\}$. The only way in which BSG-CLP moves from one state to another is by loading a block $b_i\in B$ into a free space cuboid $r_j\in R$. Thus, the search space can be represented by a directed acyclic graph (DAG), where the vertices are the states, and the edges are the transitions.

When a block b_i is loaded into r_j, C is updated by removing the boxes in b_i and B is updated by removing *unfeasible blocks*, i.e., blocks consisting of boxes which are no more in C.

In order to update the residual space (R) a more complex procedure is performed. The procedure is briefly explained in the following section.

3.1.2. Representation of the residual space (K1)

We use a *cover representation* [6] for representing the residual space. That is, the residual space is represented by a list of cuboids $R=\{r_1,r_2,\ldots,r_p\}$ which may be overlapped. This representation was proposed by Lim et al. [18] and has been used in several works [19,15,20,21,6,16].

Using the cover representation, when a block b_i is loaded into r_j, three new overlapping cuboids r_{j1},r_{j2},r_{j3} are generated (see Fig. 1). In a similar way, each free space cuboid in R that overlaps with the placed block may generate up to 6 new cuboids. The cuboid r_j and the free space cuboids overlapping b_i are removed from R. Then, all the new generated cuboids $r_{j1},r_{j2},r_{j3},\ldots$ are added into R. Finally, all the non-maximal cuboids are removed from R. More details related to the procedure for updating R can be found in [18,22].