Contents lists available at ScienceDirect

Discrete Applied Mathematics

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

Partial enumeration algorithms for Two-Dimensional Bin Packing Problem with guillotine constraints

Andrea Lodi [a,*], Michele Monaci [b], Enrico Pietrobuoni [a]

[a] DEI "Guglielmo Marconi", Università di Bologna, Viale Risorgimento 2, 40136 Bologna, Italy
[b] DEI, Università di Padova, via Gradenigo 6/A, 35131 Padova, Italy

ARTICLE INFO

Article history:
Received 4 November 2014
Received in revised form 28 May 2015
Accepted 17 September 2015
Available online xxxx

Keywords:
Two-dimensional packing
Guillotine cuts
Heuristic algorithms
Computational experiments

ABSTRACT

We consider the variant of the Two-Dimensional Bin Packing Problem in which items have to be obtained by a series of guillotine cuts and cannot be rotated. We present a heuristic algorithm based on partial enumeration, and computationally evaluate its performance on a large set of instances from the literature. Computational experiments show that the algorithm is able to produce proven optimal solutions for a large number of problems, and gives a tight approximation of the optimum in the remaining cases.

1. Introduction

In the *Two-Dimensional* Bin Packing Problem (2BP) one is requested to pack a given set of small rectangular *items* into a minimum number of larger rectangular *bins*. Items must be packed without overlapping and orthogonally, with their edges parallel to the edges of the bin. As packing items into bins also models the problem of cutting items from larger sheets, this problem finds several applications in different contexts, e.g., in cutting wood or glass, loading boxes into vehicles, warehousing of goods, newspapers pagination among others (see, e.g., [7,17]). Recently, [9] considered two-dimensional packing problems arising in Mobile WiMAX systems, in which a set of data packets is transmitted at each frame in a two-dimensional fashion by using consecutive time slots and frequencies.

According to the specific application at hand and to the type of patterns that may be produced, many 2BP variants have been studied so far in the literature. A relevant case arises when one is allowed to rotate items by 90 degrees, to possibly achieve a better usage of the bin area. Note that rotation can be allowed for a subset of items only, whereas it should be avoided in case the bin has special features (e.g., it is a decorated piece of glass). Another relevant special case is to produce k-staged patterns, i.e., solutions in which each item can be obtained with a sequence of (at most) k edge to edge cuts parallel to the edges of the bin. This kind of problems, introduced in [5], is motivated in applications in which automatic machines are used to cut the items, and some cost is incurred for each cut that is executed. Many papers in the literature addressed the case in which $k = 2$, i.e., where items have to be packed into *levels*. Integer Linear Programming (ILP) models for level packing with a polynomial number of variables and constraints were given in [8,11], and were extended to the case $k = 3$ in [16]. The problem in which no explicit bound is imposed on the number of cuts that are allowed is known as the *Two-Dimensional Guilloting Bin Packing*. Fig. 1 shows an example of a non guillotine pattern for a given set of items (left), as well as a packing of the same items that satisfies the guillotine requirement (right).

[*] Corresponding author.
E-mail addresses: andrea.lodi@unibo.it (A. Lodi), monaci@dei.unipd.it (M. Monaci), enrico.pietrobuoni@gmail.com (E. Pietrobuoni).

http://dx.doi.org/10.1016/j.dam.2015.09.012
0166-218X/© 2015 Elsevier B.V. All rights reserved.

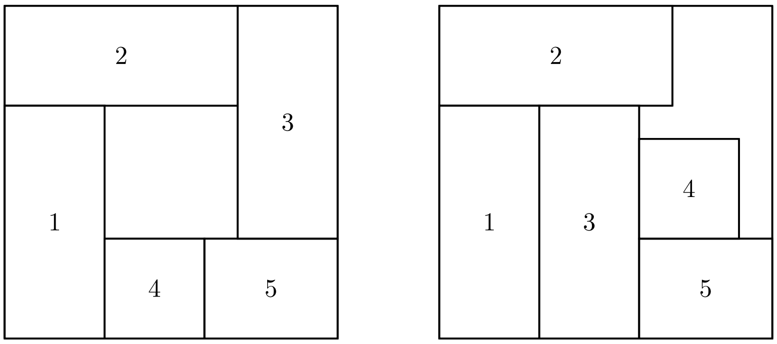

Fig. 1. Example of non guillotine and guillotine patterns.

In this paper we consider the Two-Dimensional Guillotine Bin Packing, and assume that items have a fixed orientation and cannot be rotated. Following the three-field notation proposed in [10] we will denote this problem by 2BP|O|G and the version of the problem in which guillotine constraint is not imposed by 2BP|O|F. Note that, according to the typology of cutting and packing introduced in [18], the latter problem can be classified as two-dimensional, rectangular SBSBPP. Finally, we will denote by $N = \{1, \dots, n\}$ the set of items to cut, the jth having width w_j and height h_j, and by W and H the sizes of the bins. Finally, we assume without loss of generality that all input data are integer numbers.

An exact algorithm based on integrating column generation and constraint programming for solving both 2BP|O|F and 2BP|O|G was given in [14], whereas in [3] a dynamic programming algorithm was used to solve a related two-dimensional packing problem. As far as heuristic solutions are concerned, we observe that many algorithms for 2BP (e.g., finite-best-strip and finite-first-fit [1], floor–ceiling and knapsack-heuristics [10]) pack items according to a 2-staged policy, hence producing patterns that are guillotinable. As to heuristics specifically devoted to 2BP|O|G we mention the agent-based algorithm proposed in [15], a constructive heuristic given in [2] and the three insertion heuristics introduced in [4]. A number of alternative methods for solving Two-Dimensional Bin Packing problems are given in [6].

Paper contribution and organization. In this paper we present a heuristic algorithm, based on a truncated enumerative scheme, for solving 2BP|O|G. In Section 2 we give possible ways for packing a given set of items into a bin according to a guillotine pattern, and present the general idea of our approach; a number of improvements over the basic version of the algorithm are discussed in Section 3. Section 4 reports our computational experiments on a large set of instances from the literature. This analysis provides as a byproduct an interesting computational evaluation of the solution worsening when moving from 2BP|O|F to 2BP|O|G. Finally, Section 5 draws some conclusions and presents future research directions.

2. Basic heuristic algorithm

Our algorithm is based on an enumeration tree that defines, at each level p, the current content for the pth bin in the solution. The algorithm is heuristic in nature, in that not all possible feasible ways of packing the bins are taken into account. Leaf nodes correspond to complete solutions, in which all items have been packed. Intermediate nodes have a number of descendant nodes, associated with different ways to pack the next bin in a guillotine way. In particular, each bin is filled by applying a *packing strategy* that packs one item at a time according to a given *selection rule* and *guillotine split rule*. The selection rule determines the next item to pack (and its position in the bin), whereas the guillotine split rule is used to ensure the produced pattern being guillotinable. In our algorithm we use a set $\mathcal{S}$ of selection rules and a set $\mathcal{G}$ of guillotine split rules, and generate a descendant node for each pair (s, g) with $s \in \mathcal{S}$ and $g \in \mathcal{G}$. The search tree is then explored according to a depth-first strategy, by considering descendant nodes according to the selection rule and then by the guillotine split rule. Obviously, if a feasible solution is found with value, say z, no descendant nodes are generated from nodes at level $z - 1$ as these nodes cannot produce any improvement on the incumbent.

The packing strategy used in the algorithm is described in Section 2.1, while the sets of selection rules and guillotine split rules implemented are described in Sections 2.2 and 2.3, respectively.

2.1. Packing the current bin

In this section we describe the packing strategy that is used, at each node, to pack the current bin. We will denote by $J \subseteq N$ the set of items that have not been allocated in the previous bins (i.e., in the previous levels of the tree). Recall that each node implements a fixed selection rule and a fixed guillotine split rule.

Our packing strategy packs one item at a time in the free space of the bin, so as to guarantee that a guillotinable pattern is obtained. In particular, we maintain a list $\mathcal{F} = \{F_1, \dots, F_m\}$ of pairwise disjoint rectangular (free) spaces where the items in J can be allocated. Initially, $\mathcal{F} = \{F_1\}$, i.e., there is only one space that corresponds to the entire bin.

At each iteration, we determine the set $\mathcal{P}$ of pairs (j, F_i) associated with feasible packings of an item $j \in J$ in a free space $F_i \in \mathcal{F}$ and compute, for each such pair, an *insertion score* according to the given selection rule. The pair, say (j^*, F_{i^*}), that