Information Processing Letters 116 (2016) 15–21

Contents lists available at ScienceDirect

Information Processing Letters

www.elsevier.com/locate/ipl

A priority heuristic for the guillotine rectangular packing problem

Defu Zhang [a,b,*], Leyuan Shi [b], Stephen C.H. Leung [c], Tao Wu [b]

[a] Department of Computer Science, Xiamen University, 361005, China
[b] Department of Industrial and Systems Engineering, University of Wisconsin-Madison, USA
[c] Department of Management Sciences, City University of Hong Kong, Hong Kong

ARTICLE INFO

Article history:
Received 5 April 2014
Received in revised form 21 June 2015
Accepted 6 August 2015
Available online 21 August 2015
Communicated by S.M. Yiu

Keywords:
Packing problem
Heuristic algorithm
Recursive
Design of algorithms

ABSTRACT

A new priority heuristic is presented for the guillotine rectangular packing problem. This heuristic first selects one available item for a given position by a priority strategy. Then it divides the remaining space into two rectangular bins and packs them recursively, and its worst-case time complexity is $T(n) = O(n^2)$. The proposed algorithm is a general, simple and efficient method, and can solve different packing problems. Computational results on a wide range of benchmark problems have shown that the proposed algorithm outperforms existing heuristics in the literature, on average.

1. Introduction

Cutting and packing problems are related to many areas of operations research as they have diverse industrial applications such as apparel manufacturing, glass cutting, multiprocessor task scheduling, cargo loading and integrated circuit layout design. These applications can be formulated as packing problems with their respective constraints and objectives. Possible constraints include guillotine cutting and fixed orientation packing. Guillotine cutting is often required in many industrial fields since the machine cuts different types of materials into many small pieces using orthogonal cutting. The objective is to minimize the waste or height of material or maximize space utilization in the bin.

According to the typology of packing problems in Wäscher et al. [17], the guillotine rectangular packing problems (GRPP) include two classes, each class includes two variants:

- Strip packing problem (SPP): Given an open bin of width W and unlimited height, and a set of n rectangular items with sizes (h_i, w_i), $i = 1, \ldots, n$, the objective is to place each item in the bin without overlapping, such that the bin's required height is minimized. This problem includes two variants: OG and RG, where O denotes the case where items are placed with a fixed orientation, G denotes guillotine constraints are required, and R denotes items may be rotated by 90 degrees.
- Single bin packing problem (SBPP): Given a rectangular bin of width W and height H, and a set of n rectangular items, the objective is to maximize space utilization (or "filling rate") of the rectangular bin. Similarly, this problem includes two-variants: OG and RG.

GRPP is NP-hard and is difficult to solve (Belov [1], Messaoud et al. [12]). Some researches for GRPP have shown exact algorithms only solve small-scale problems (Hifi and

* Corresponding author at: Department of Computer Science, Xiamen University, 361005, China. Tel.: +86 0592 5918207; fax: +86 0592 2580258.
E-mail address: dfzhang@xmu.edu.cn (D. Zhang).

http://dx.doi.org/10.1016/j.ipl.2015.08.008
0020-0190/

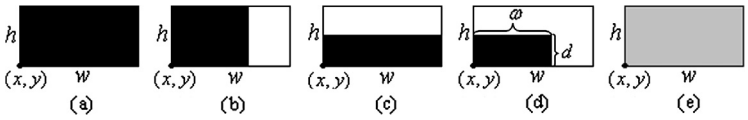

Fig. 1. Possible cases for S while placing one item.

M'Hallah [6], Cui et al. [5]), heuristic algorithms are preferred if fast computing is required for real-world applications.

Constructive heuristic algorithms cannot guarantee a solution of good quality but they can find a feasible solution in relatively short time. In particular, they can be combined with exact algorithms and metaheuristic algorithms. Coffman et al. [3], Coffman and Shor [4] presented several level-oriented heuristic algorithms: first-fit decreasing height (FFDH) algorithm, best-fit decreasing height (BFDH) algorithm. Lodi et al. [10] introduced the floor-ceiling (FC) algorithm. Martello et al. [11] also developed a heuristic algorithm (JOIN). Zhang et al. [20] presented a heuristic recursive algorithm (HR). Bortfeldt [2] further improved BFDH and obtained a BFDH* algorithm. Polyakovsky and M'Hallah [16] presented a new guillotine bottom left (GBL) heuristic algorithm. Ortmann et al. [14] developed four new and improved heuristic algorithms: modified size-alternating stack algorithm (SASm), best fit with stacking algorithm (BFS), stack ceiling {with re-sorting} algorithm (SC{R}), and stack level algorithm (SL_5).

Based on constructive heuristics, metaheuristics are widely applied to solve GRPP (Bortfeldt [2], Polyakovsky and M'Hallah [16], Wei et al. [19], Hong et al. [7]), and obtained some excellent results. Therefore, it is important to design a general heuristic algorithm that can quickly find a solution for GRPP.

2. New priority heuristic algorithm

A recursive technique is useful for GRPP as it may be used to restrict items' locations such that they can satisfy the guillotine constraint. The concept is simple as a rectangular space may be divided into several smaller rectangular spaces, and each smaller space can be divided recursively. From Fig. 1(a), we observe that the rectangular bin S is determined by its position (x, y), and its width w and its height h. The core purpose of the proposed algorithm is to fill S efficiently.

Each unplaced item can be placed into S resulting in five scenarios (the placed item is marked in black): Fig. 1(a)–(e) express cases (a)–(e) respectively. It can be observed intuitively that case (a) is the best because the item fills up the whole space. Cases (b) and (c) are better than case (d) because the remaining space in cases (b) and (c) are rectangles, while the remaining space in case (d) requires careful partitioning. Case (e) represents that no item can be placed into S and S is wasted. Which of cases (b) or (c) is better depends on the practical problems. Different problems may consider different sorting of items. For strip packing problem, sorting is usually selected by non-increasing height. Under this case, case (b) is said to be better than case (c) because the item with the larger height may have more chance to be selected to place first.

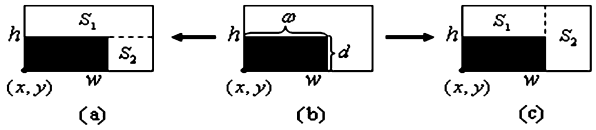

Fig. 2. The way of partitioning of S.

Similarly, if sorting is done by non-increasing width, then case (c) is better than case (b). Assume that the items are sorted by non-increasing height, then items that match cases (a), (b), (c), (d) and (e) are assigned priority 1, 2, 3, 4 and 5, respectively.

Among unplaced items, those with the highest priority (1 is the highest) are chosen for placement first. Stop packing S when case (e) occurs because all unpacked items cannot be packed into S. For cases (b) and (c), the number of bins to be placed does not increase. If two or more items have the same priority, then the first hit item in the sorted list is packed first. For case (d), the division of S is important, and is done as follows: Let $\min w$ and $\min h$ be the two parameters related to all unplaced items where, for fixed orientation packing, $\min w$ is the minimum width of all unplaced items and $\min h$ is the minimum height of all unplaced items. According to Fig. 2(b), if the value $w - \omega$ is less than $\min w$, S is divided into S_1 and S_2, as in Fig. 2(a) instead of as in Fig. 2(c), which implies that S_2 will be wasted, however, S_2 in Fig. 2(c) is larger than S_2 in Fig. 2(a). Therefore, this partition can make S_1 larger, so that it can be used by other unplaced items. Similarly, if the value $h - d$ is less than $\min h$, then S is divided into S_1 and S_2, as in Fig. 2(c), implying that S_1 will be wasted. Otherwise, there are two ways to divide S, as in Fig. 2(a) and (c). One partition is as in Fig. 2(a), another partition is shown in Fig. 2(c). Which partition is selected depends on if ω is less than $\min w$. If ω is less than $\min w$, then the former is selected because condition $\omega < \min w$ leads to waste of bin S_1 as in Fig. 2(c).

In fact, orientation of the partition is the determining factor and depends on the way the items are sorted. The aim of the strip packing problem is to minimize the height of the bin, such that the partition in Fig. 2(c) is more efficient because items with large heights have greater chance to be placed. For the single bin packing problem, orientation of the partition is mainly determined by the way the items are sorted. Orientation of the partition is shown in Fig. 2(c), when items are sorted by non-increasing width, because the case of $\omega < \min w$ rarely occurs and items with large heights have a greater chance to be placed. In fact, partitioning may be proceeded more carefully by considering $d < \min h$ or by taking other factors into account. For example, for the case of $d < \min h$, dividing S into S_1 and S_2 as in Fig. 2(a) will result the wastage of S_2. In this case, the partition as in Fig. 2(c) is better. However, it is a complex issue and depends on the nature of the