European Journal of Operational Research 241 (2015) 674–685

Contents lists available at ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

Dynamic reduction heuristics for the rectangle packing area minimization problem ☆

Kun He [a], Pengli Ji [a,*], Chumin Li [a,b]

[a] School of Computer Science and Technology, Huazhong University of Science and Technology, Wuhan 430074, China
[b] School of Computer Science and Technology, University of Picardie Jules Verne, Amiens 80039, France

ARTICLE INFO

Article history:
Received 7 October 2013
Accepted 21 September 2014
Available online 2 October 2014

Keywords:
Packing
Floorplanning
Layout optimization
Area minimization
Open dimension problem

ABSTRACT

The rectangle packing area minimization problem is a key sub-problem of floorplanning in VLSI design. This problem places a set of axis aligned two-dimensional rectangular items of given sizes onto a rectangular plane such that no two items overlap and the area of the enveloping rectangle is minimized. This paper presents a dynamic reduction algorithm that transforms an instance of the original problem to a series of instances of the rectangle packing problem by dynamically determining the dimensions of the enveloping rectangle. We define an injury degree to evaluate the possible negative impact for candidate placements, and we propose a least injury first approach for solving the rectangle packing problem. Next, we incorporate a compacting approach to compact the resulting layout by alternatively moving the items left and down toward a bottom-left corner such that we may obtain a smaller enveloping rectangle. We also show the feasibility, compactness, non-inferiority, and halting properties of the compacting approach. Comprehensive experiments were conducted on 11 MCNC and GSRC benchmarks and 28 instances reported in the literature. The experimental results show the high efficiency and effectiveness of the proposed dynamic reduction algorithm, especially on large-scale instances with hundreds of items.

1. Introduction

The rectangle packing area minimization problem (RPAMP) is an NP-hard problem. It aims to place all the axis-aligned rectangular items of known sizes onto a plane completely and without overlapping to minimize the area of the enveloping rectangle. As a subproblem of floorplanning, the RPAMP has important applications in the area of VLSI design. For the floorplanning problem, we are given a set of rectangular modules and a net list specifying the interconnections among the modules. The goal is to find a feasible layout the same as RPAMP, such that the area of the enveloping rectangle and the total length of the interconnections among the modules is minimized.

Basically, there are two main approaches to the RPAMP: (1) a heuristic searching method based on layout representations and (2) a reduction method that transforms an instance of the RPAMP to a series of instances of the strip packing problem (SPP) or the rectangle packing problem (RPP). The RPAMP is a two-variable open dimension problem (Wäscher, Haußner, & Schumann, 2007), while the SPP and the RPP are a one-variable open dimension problem and a two-dimensional (2D) knapsack problem. In addition to the searching method and the reduction method, other methods such as branch and bound (Chan & Markov, 2004), linear optimization (Kim & Kim, 2003) are also used to solve the RPAMP.

Layout representation is one of the most important techniques used in the first approach, as it determines the size of the searching space and the complexity of transformation between a representation and the corresponding layout. Layout representation can be classified into slicing representation and nonslicing representation. The layout coded by slicing representation should satisfy guillotine cutting, so slicing representation may miss optimal layouts; meanwhile, nonslicing representation can cover all the optimal layouts. Hence it is commonly believed that nonslicing representation can yield better results than slicing representation.

Numerous representations have been put forward in the past two decades, and there are four classical nonslicing representations. Murata, Fujiyoshi, Nakatake, and Kajitani (1996) proposed a sequence pair representation that used two sequences to represent the geometric relation of the items, placed the items on a grid structure, and constructed the corresponding constraint graphs to evaluate the objective function. Lin and Chang (2005) presented a transitive constraint graph (TCG) by using two transitive closure graphs to identify the geometric relation of the items. TCG is equivalent to sequence

☆ This work was supported by the National Science Foundation of China under Grants 61472147, 61173180 and 61272014.

* Corresponding author. Tel.: +86 27 1867 233 1252; fax: +86 27 87545004.
E-mail addresses: brooklet60@gmail.com (K. He), jipengli8@gmail.com (P. Ji), chu-min.li@u-picardie.fr (C. Li).

http://dx.doi.org/10.1016/j.ejor.2014.09.042
0377-2217/

pair because they share the same $O((n!)^2)$ solution space and there is a bijective mapping between them. Two representations based on tree structure, O-tree (Guo, Cheng, & Yoshimura, 1999) and B*-tree (Chang, Chang, Wu, & Wu, 2000), are the most widely used representations, because their solution space is in the size of $O(n!2^{2n-2}/n^{1.5})$, which is the lowest complexity reported in the literature. B*-tree uses a binary ordered tree, while O-tree uses an ordered tree with an arbitrary vertex degree. Therefore, B*-tree is faster and easier to implement. Chang et al. (2000) presented the comparisons and analyses on different representations.

Simulated annealing (SA; Chen & Chang, 2006; Chen & Yoshimura, 2008; Chen, Zhu, & Ali, 2011; Pisinger, 2007) is the most popular searching strategy for the RPAMP, while memetic algorithm (MA; Tang & Yao, 2007), particle swarm optimization (PSO; Chen, Guo, & Chen, 2010) and local search (Imahori, Yagiura, & Ibaraki, 2005; Li, Li, & Zhou, 2010) have also been adopted by researchers. To reduce the searching complexity of SA, Fast-SA (Chen & Chang, 2006) integrated a random search with hill-climbing that divided the annealing process into three stages: the high-temperature random search stage, the pseudo-greedy local search stage, and the hill-climbing search stage. By enumerating all the possible inserted positions in the sequence pairs for a selected item, Chen and Yoshimura (2008) applied an insertion after remove (IAR) method to accelerate the SA. Chen et al. (2011) presented a hybrid simulated annealing (HSA) method based on a new operation on B*-tree. Tang and Yao (2007) proposed a memetic algorithm (MA) that used an effective genetic search method to explore the search space and an efficient local search method to exploit information in the search region.

The main idea of the second approach is to construct a set of candidate widths or heights of the enveloping rectangle to transform an RPAMP instance into a series of instances of the RPP or the SPP and then to design algorithms for the RPP and the SPP. Because various reduction methods adopt similar approaches to constructing candidate dimensions of the enveloping rectangle, the design of the reduction method focuses on seeking efficient RPP and SPP algorithms. By combining an improved least flexibility first principle and a greedy search, Wu and Chan (2005) introduced a deterministic optimization algorithm for the RPP. Based on the conceptions of corner action and smooth degree, He, Huang, and Jin (2012) proposed a best fit algorithm (BFA) for the RPP. Bortfeldt and Gehring (2001) proposed a hybrid genetic algorithm to generate layout with a layer-type structure for the RPP. To solve the SPP, (Leung, Zhang, & Sim, 2011) suggested a two-stage intelligent search algorithm that first constructed a solution greedily, then improved the solution by a local search and a simulated annealing algorithm. Bortfeldt (2006) introduced a layer building method best fit decreasing height* (BFDH*) algorithm, which worked without any encoding of solutions, but fully defined layouts are manipulated by means of specific genetic operators.

In this paper, we develop a dynamic reduction algorithm (DRA) for the RPAMP. First, DRA uses a constructive method (Bortfeldt, 2013) to generate a set of candidate widths for the enveloping rectangle and initializes a promising filling rate (the filling rate = the area of all the placed items divided by the area of the enveloping rectangle). Then, at each iteration, it constructs an RPP instance based on the current width dynamically selected from the candidate widths and current promising filling rate. Then, it uses a least injury first (LIF) algorithm, which is an algorithm developed from the BFA (He et al., 2012), to calculate the generated RPP instance. If LIF can place all the items on the enveloping rectangle, then we move all the items toward the left and bottom iteratively to adjust the obtained layout to a compact one. This process is repeated until the promising filling rate can no longer be improved. Fig. 1 shows the flow chart of LIF.

We implemented DRA and tested it on 11 MCNC (Microelectronic Center of North Carolina) and GSRC (Gigascale Systems Research Center) benchmarks and four other RPAMP benchmarks proposed by Imahori et al. (2005). The experimental results showed that DRA renovated the current best results on eight instances. At the same time it also matched the current best results on the other three instances. Then, we ran DRA on 24 RPAMP instances proposed by Bortfeldt (2013), and DRA renovated results on 10 of 12 instances having 200 items.

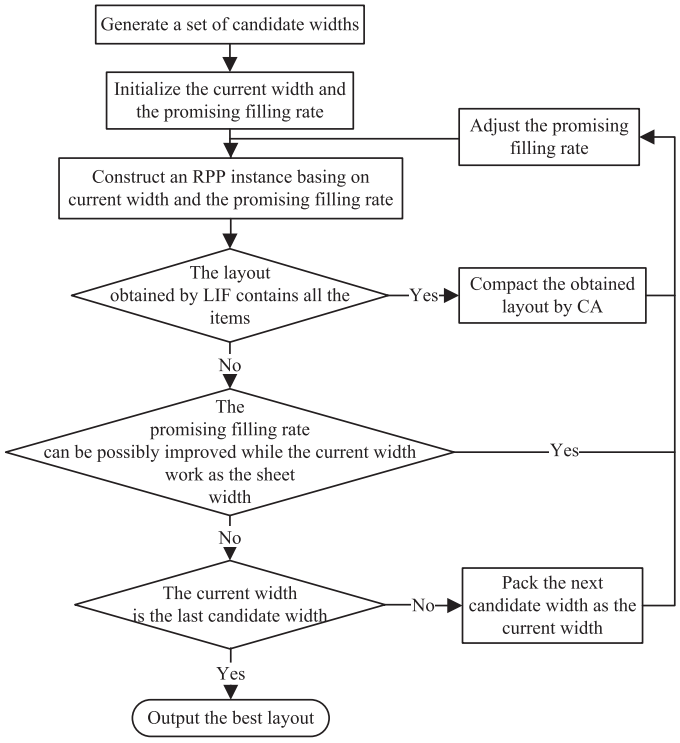

Fig. 1. The flow chart of DRA.

The subsequent sections are organized as follows. Section 2 gives a formal statement on the problem definition. Section 3 introduces the LIF algorithm. Section 4 describes the compacting algorithm, and DRA is introduced in detail in Section 5. Empirical studies on the DRA are presented in Section 6, and the conclusions are presented in the end.

2. Problem statement

Given a set of n rectangular items with each item i $(1 \le i \le n)$ having width w_i and height h_i, the RPAMP requires determining a feasible arrangement of all the items on a larger rectangular plane with variable dimensions. The objective is to minimize the area of the enveloping rectangle (hereafter abbreviated as sheet). Let the sheet be embedded in the first quadrant of a 2D Cartesian reference frame in such a way that the bottom-left vertex coincides with the origin. For each item i, let (x_{i1}, y_{i1}) and (x_{i2}, y_{i2}) denote the coordinates of the bottom-left and upper-right vertexes, respectively. Let width w_c and height h_c represent the two dimensions of the sheet. Then, RPAMP can be formulated as follows:

$$\min w_c h_c$$

s.t.

(1) $(x_{i2} - x_{i1}, y_{i2} - y_{i1}) \in \{(w_i, h_i), (h_i, w_i)\}$
(2) $\max(x_{i1} - x_{j2}, x_{j1} - x_{i2}, y_{i1} - y_{j2}, y_{j1} - y_{i2}) \ge 0$
(3) $0 \le x_{ik} \le w_c, 0 \le y_{ik} \le h_c, k \in \{1, 2\}$

In constraints (1)–(3), i, j apply to $1, 2, \ldots, n$ and $i \ne j$. Constraint (1) implies that each item should be placed orthogonally on the sheet; constraint (2) indicates that no overlap occurs between any two items; and constraint (3) means all the items are placed completely on the sheet.