数理解析研究所講究録
第 1799 巻 2012 年 153-156

Heuristic Algorithms for Rectilinear Block Packing

Yannan Hu (Nagoya University)

Hideki Hashimoto (Nagoya University)

Shinji Imahori (Nagoya University)

Mutsunori Yagiura (Nagoya University)

1 Introduction

The rectilinear block packing problem is a problem of packing a series of arbitrary shaped rectilinear blocks into a larger rectangular container without overlap so as to minimize a given objective function. This problem involves many industrial applications, such as VLSI design, timber/glass cutting, textile industry and newspaper layout. It belongs to a subset of classical packing problems and is known to be NP-hard. A special case of the rectilinear block packing problem is the rectangle packing problem, for which many efficient algorithms have been proposed.

This paper proposes heuristic algorithms for the rectilinear block packing problem by generalizing representative construction heuristics for rectangle packing. The basic idea of our algorithms is that we regard a rectilinear block as a set of rectangles with certain constraints on their relative positions. Hence, we can also deal with packing problems in which each item consists of a set of rectangles having some constraints on the relative positions between them. The main strategy of our algorithms is the *bottom-left strategy*, which derives from the *bottom-left algorithm* for rectangle packing [2]. In this strategy, whenever a new item is being packed into the container, it will be placed at the *BL position* relative to the current layout. The BL position of a new item relative to the current layout is defined as the leftmost point among the lowest *bottom-left stable feasible positions*, where a bottom-left stable feasible position is a point such that the new item can be placed without overlap and cannot be moved neither leftward nor downward. We consider two standard rules for choosing the new item from the remaining items. The resulting algorithms are called the *bottom-left algorithm* and the *best-fit algorithm*.

We propose efficient implementations by incorporating a sweep line method for finding BL positions. Our algorithms can deal with both cases where the rotation of items is allowed and not. A series of experiments on some benchmark instances are performed and two example layouts computed by our algorithm are reported.

2 Problem Description

We are given a set of n items $R = \{R_1, R_2, \ldots, R_n\}$ of rectilinear blocks, where each rectilinear block takes a deterministic shape and size from a set of t types $T = \{T_1, T_2, \ldots, T_t\}$. We are also given a rectangular container with fixed width and unrestricted height. The objective is to pack all the items orthogonally without overlap into the container so as to minimize the height of the container. In this paper, two cases of this problem are considered: (1) all the items are not allowed to be rotated, and (2) all the items can be rotated 90°, 180° or 270°.

3 Efficient Implementation of Construction Heuristics

For a new rectilinear block to be placed next and a set of placed items, the *overlap number* of a point (x, y) is defined as the number of items that overlap with the new block when it is placed at (x, y). In this paper, we use no-fit polygons (abbreviated as NFPs) [1] as a crucial technique to calculate overlap numbers. The NFPs describe the positions where a new item to be packed can be placed without intersection relative to the current layout. We define the *cross point* as the crossing point where an NFP's right edge crosses another's top edge. Observe that, bottom-left stable feasible positions will only appear at non-overlapping cross points. By checking the overlap number of the cross points, we can easily compute the BL position.

In this paper, we use the Find2D-BL algorithm [4] that enumerates bottom-left stable feasible positions. It uses a sweep line, which is parallel to the x-axis. With the sweep line moving upward from the bottom, the first bottom-left stable feasible position appears as the BL position.

3.1 Bottom-Left Algorithm

The bottom-left algorithm can be generally explained as follows: We are given a set of n rectilinear items R and an order of items (e.g., decreasing order of width). The algorithm packs the items one by one according to the given order, where each item is placed at its BL position of the current layout.

3.2 Best-Fit Algorithm

The basic idea of the best-fit algorithm comes from the *best-fit algorithm* for the rectangle packing problem, which proposed by Burke et al. [3]. The best-fit algorithm can be generally explained as follows: We are given a set of n rectilinear items R and priority among them. The algorithm packs items one by one, and in each iteration, it dynamically chooses a rectilinear block among the remaining items to pack by the following rules:

- Calculate the BL positions of all the remaining items.
- The rectilinear block, whose BL position takes the smallest x-coordinate among those with the lowest y-coordinate, is packed in this iteration (if there exist ties, the one with the highest priority is chosen).

4 Time Complexity

The number of rectangles which are divided from a rectilinear block R_i is denoted by m_i. M denotes the sum of m_i over all the n rectilinear blocks. The sum of m_i of the rectilinear blocks from t distinct types is denoted by m.

To find the BL position of R_i, the Find2D-BL algorithm takes $O(Mm_i \log M)$ time. Hence, the time complexity of the bottom-left algorithm is $O(M^2 \log M)$. For the best-fit algorithm, we propose some ideas to reduce the time complexity from $O(nMm \log M)$ to $O(Mm \log M)$.

Note that there are not more than four rotation angles for each rectilinear block. This implies that the time complexities of the two algorithms for the case where rotation is allowed are the same as the case without rotation.

5 Computational Result

Figure 1 and 2 show two example layouts computed by our best-fit algorithm and bottom-left algorithm. In these two experiments, we take the same instance and the same container to compare the results. The number of items is 1280, the width of the container is 220, and rotation is considered.

The occupation rate obtained by our best-fit algorithm is 98.59% with the running time of 8.30 seconds. The occupation rate by the bottom-left algorithm is 95.92% and its running time is 11.26 seconds.

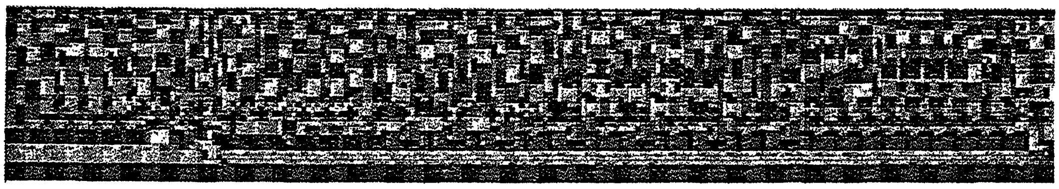

Figure 1: A layout obtained by the best-fit algorithm

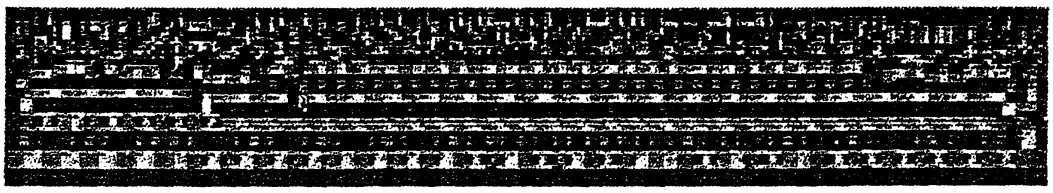

Figure 2: A layout obtained by the bottom-left algorithm

6 Conclusions

In this paper, we proposed efficient ways of implementing the bottom-left algorithm and the best-fit algorithm for the rectilinear block packing problem. We analyzed the time complexity of our algorithm, and showed that the bottom-left algorithm runs in $O(M^2 \log M)$ time and the best-fit algorithm runs in $O(Mm \log M)$ time. We also performed a series of experiments based on some benchmark instances. The occupational rate of the packing layouts computed by our algorithms was 95% on average for instances with more than 1000 items. Even for instances with more than 10,000 rectilinear blocks to be packed, the proposed algorithms work in a reasonable computation time.

References

[1] R.C. Art: An approach to the two dimensional irregular cutting stock problem, IBM Cambridge Science Center, 36.Y08 (1966).

[2] B.S. Baker, E.G. Coffman Jr. and R.L. Rivest: Orthogonal packings in two dimensions, SIAM Journal on Computing, vol. 9, pp. 846–855 (1980).

[3] E.K. Burke, G. Kendall and G. Whitwell: A new placement heuristic for the orthogonal stock-cutting problem, Operations Research, vol. 52, pp. 655–671 (2004).

[4] S. Imahori, Y. Chien, Y. Tanaka and M. Yagiura: Enumerating bottom-left stable positions for rectangles with overlap, In Proceedings of the 9th Forum on Information Technology 2010 (FIT2010), Issue 1, pp. 25–30.