数理解析研究所講究録
第 1676 巻 2010 年 100-114

# 一次元連続ビンパッキング問題に対する厳密解法
# (An Exact Method for the One-dimensional Contiguous Bin Packing Problem)

京都大学・情報学研究科 荒堀 洋平 (Yohei Arahori)*
Graduate School of Informatics, Kyoto University
京都大学・情報学研究科 今道 貴司 (Takashi Imamichi)†
Graduate School of Informatics, Kyoto University
京都大学・情報学研究科 永持 仁 (Hiroshi Nagamochi)
Graduate School of Informatics, Kyoto University

**Abstract**

Given a set of rectangles and a rectangular container with a fixed width, called a strip, the two-dimensional strip packing problem (2SP) requires all the given rectangles to be placed orthogonally without overlap within the strip so as to minimize the height of the strip. The problem and its variants have many applications in steel and textile industries, and it has indirect application in scheduling problems. However, 2SP is known to be NP-hard. The one-dimensional contiguous bin packing problem (1CBP) is a relaxed problem of 2SP. 1CBP is originally proposed for giving a lower bound on the optimal value to the 2SP [1]. 1CBP is also known to be NP-hard. In this paper, we propose an exact algorithm for 1CBP incorporating a branch-and-bound algorithm for 1CBP with fixed height problem (1CBPFH), a decision problem of 1CBP which asks whether there is a feasible placement of all rectangles within the strip with fixed width and height, and a heuristic algorithm for 2SP with fixed height problem (2SPFH). Our algorithm can deal with not only 1CBP without rotations but also 1CBP with rotations of 90 degrees.

We conducted experiments using benchmark instances. Our algorithm succeeded to find the optimal values for most of these instances in a practical time. Especially, we found that the optimal values of instances "gcut02" and "cgcut02" (without rotations) are 1187 and 64, respectively, which have not been known by any of the existing algorithms.

*Key words*: One-dimensional contiguous bin packing problem; Two-dimensional strip packing problem; Canonical Form; Branch-and-bound.

## 1 Introduction

The *two-dimensional strip packing problem (2SP)* requires all given rectangles to be placed orthogonally without overlap into one rectangular strip, called the *strip*, with a fixed width and a variable height so as to minimize the height of the strip. 2SP is known under various names including the *(orthogonal) rectangular strip packing problem*. According to the recent typology of packing problems [2], 2SP without rotations is categorized into *the two-dimensional rectangular open dimension problem with a single variable dimension (2D rectangular ODP)*. Most of the variants of the rectangle packing problem, including 2SP, are known to be NP-hard. Rectangle packing problems have many applications in the steel and textile industries, and they also have indirect applications in scheduling problems [3] and in other areas [4, 5].

The *one-dimensional continuous bin packing problem (1CBP)* is a relaxed problem of 2SP, which is proposed by Martello et al. [1] to obtain a lower bound on the optimal value of 2SP. In 1CBP, each rectangle split into a set of rectangular *bands* with height one. 1CBP requires

*Present affiliation: 東レ株式会社 (Toray Industries, Inc.)
†Present affiliation: IBM 東京基礎研究所 (IBM Research - Tokyo)

all these sets of bands to be placed into the original strip in such a way that the bands in each set are allowed to take different $x$-coordinates as long as they are placed at contiguous $y$-coordinates. 1CBP also has two variants with and without rotations of 90 degrees, which are both known to be NP-hard.

In this paper, we design an exact algorithm for 1CBP *with* and *without* rotations of 90 degrees based on the branch-and-bound method. For this, we introduce a new idea on a lower bound on optimal values and a branching operation based on the *canonical form* of placements to reduce search space. In this paper, we omit the details of the bounding operations and a heuristic algorithm to obtain a feasible solution due to the limitation of pages. Experimental results on benchmark instances revealed that our algorithm can find optimal solutions to most benchmark instances in a practical time.

## 1.1 Related Work

Among the many variants of rectangle packing problems, 2SP is one of the problems most intensively studied [4].

Baker et al. [6] proposed a construction heuristics called the *bottom-left-fill (BLF)* algorithm for 2SP, and many related papers have appeared, e.g., an efficient implementation [7]. Different types of construction heuristics have also been proposed recently, e.g., the *best-fit* heuristics [8]. Construction heuristics are often incorporated in metaheuristics to improve the quality of solutions. One of the common ways is to use metaheuristics for searching sequences from which BLF (or similar heuristics) generates good placements [9, 10, 11, 12, 13]. In this scheme, a sequence is an encoded solution and the heuristic such as BLF is a decoding algorithm. Metaheuristics incorporating different types of heuristics have also been proposed; e.g., GRASP [14]. Murata et al. [15] proposed a simulated annealing and Imahori et al. [5, 16] presented iterated local search algorithms based on a different coding scheme called the *sequence-pair representation*. For more about heuristic algorithms, see a survey [17].

Compared with the research on heuristics to 2SP, the research on exact method to 2SP has started recently. Martello et al. [1] proposed exact algorithms for 2SP without rotations and succeeded in solving benchmark instances with up to 200 rectangles. They introduced 1CBP and proposed a branch-and-bound algorithm to obtain a good lower bound on the optimal value of 2SP, where the algorithm constructs a solution by placing rectangles one by one. Alvarez-Valdez et al. [18] developed a new lower bound based on relaxations of an integer formulations of 2SP and reduced the tree search of their branch-and-bound algorithm using some dominance criteria. They designed an algorithm for 1CBP by incorporating the bisection method and the feasibility check using CPLEX. The algorithms in these papers exploit the constraint that the rectangles are not allowed to be rotated, and therefore they are not directly applicable to the cases where rotations are allowed. On the other hand, Kenmochi et al. [19] proposed an exact branch-and-bound algorithm for 2SP with and without rotations of 90 degrees based on the *g-staircase* placement. Because 1CBP was originally introduced only for a lower bound for 2SP, the literature is limited. To the best of our knowledge, only Martello et al. [1] and Alvarez-Valdez et al. [18] proposed algorithms for 1CBP.

Some theoretical aspects and heuristic/exact algorithms for 2SP are summarized in [20].

### 1.2 Organization

The organization of this paper is as follows. In Section 2, we formulate 2SP and 1CBP. We also introduce the decision problem *2SP with fixed height (1CBP with fixed height)* for 2SP (resp., 1CBP). In Section 3, we explain the outline of our entire algorithm EXACT1CBP. In Section 4, we introduce a new lower bound for 1CBP. In Section 5, we present an exact the algorithm BB-1CBPFH based on the branch-and-bound method. The experimental results on benchmark instances are reported in Section 6 and we give a concluding remark in Section 7.

## 2 Formulations

### 2.1 Two-dimensional Strip Packing Problem (2SP)

We let $(W, H)$ mean a strip with fixed width $W$ and height $H$. An instance $(I, W)$ of 2SP consists of a set $I = \{r_1, r_2, \ldots, r_n\}$ of $n$ rectangles and a width $W$ of a strip, where the height of the strip is the objective to be minimized. Note that the widths and heights are all integers. 2SP requires the $n$ rectangles to be placed without overlap into the strip $(W, H)$ so as to minimize the height $H$ of the strip. We designate the bottom left corner of the strip as the origin of the $xy$-plane, letting the $x$-axis be the direction of the width of the strip, and the $y$-axis be the direction of the height. We represent the location of each rectangle $i$ in the strip by the coordinate $(x_i, y_i)$ of its bottom left corner (see Figure 1). The set $\pi = \{(x_i, y_i) \mid r_i \in I\}$ of coordinates is called a *placement* of $I$.

2SP *without rotations* is formulated as follows:

$$\begin{array}{lll}
\text{minimize} & H & \\
\text{subject to} & x_i + w_i \leq W, \quad r_i \in I, & (1) \\
 & y_i + h_i \leq H, \quad r_i \in I, & (2) \\
 & x_i + w_i \leq x_j \ \text{ or } \ x_j + w_j \leq x_i \ \text{ or} & \\
 & \quad y_i + h_i \leq y_j \ \text{ or } \ y_j + h_j \leq y_i, \quad r_i, r_j \in I, \ i \neq j, & (3) \\
 & x_i, y_i \geq 0, \quad r_i \in I, & (4)
\end{array}$$

where the height $H$ and the $x$- and $y$-coordinates $x_i$ and $y_i$ ($r_i \in I$) are variables and the strip width $W$ and the height $h_i$ and width $w_i$ ($r_i \in I$) are given constants. The constraints (1), (2) and (4) require all rectangles to be placed within a strip $(W, H)$. The constraint (3) prevents rectangles from overlapping each other. A placement $\pi$ is *feasible* to an instance $(I, W)$ without rotations if it satisfies the constraints (1), (2), (3), and (4). Otherwise, $\pi$ is *infeasible*. We denote the optimal value $H$ of a given instance $(I, W)$ without rotations by $OPT_{2\mathrm{SP}}(I, W)$.

We also define the two-dimensional strip packing problem with fixed height (2SPFH) without rotations by regarding $(I, W, H)$ as an instance such that the rectangles in $I$ are required to be places in the strip $(W, H)$ without overlap. We call a 2SPFH instance $(I, W, H)$ *feasible* if it admits a feasible placement of $I$ within strip $(W, H)$ (i.e., $H \geq OPT_{2\mathrm{SP}}(I, W)$) and *infeasible* otherwise.

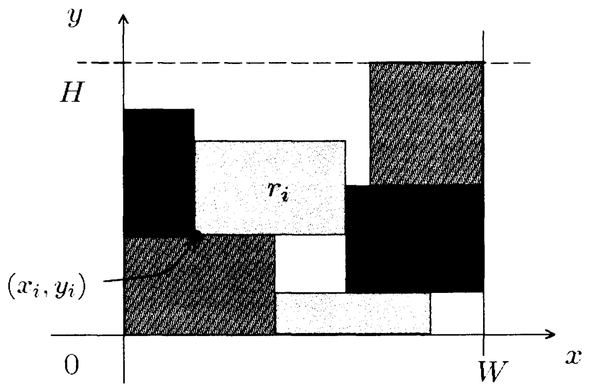

Figure 1: A feasible placement of six rectangles to the strip packing problem

## 2.2 One-dimensional Contiguous Bin Packing Problem (1CBP)

In 1CBP, each rectangle with width $w_i$ and height $h_i$ is treated as a set of $h_i$ rectangles with width $w_i$ and height 1, each of which we call a *band*. 1CBP requires all these sets of bands to be placed into the original strip in such a way that the bands in each set are allowed to take different $x$-coordinates as long as their are placed at contiguous $y$-coordinates. Note that we do not need to consider a choice of the $x$-coordinate of each band because every band has height 1. Let $\{b_i^k \mid k = 0, \dots, h_i - 1\}$ denote the set of bands obtained from a rectangle $r_i \in I$, and $y_i^k$ denote the $y$-coordinate of band $b_i^k$. Without loss of generality, we assume that $y_i^k = y_i^0 + k,\ k = 0, \dots, h_i - 1$ holds. Hence we only need to decide the $y$-coordinate of the first band $b_i^0$ of each rectangle $r_i \in I$. We denote $y_i^0$ by $y_i$ for all $r_i \in I$. A set $\pi_J = \{y_i \mid r_i \in J\}$ of $y$-coordinates for a subset $J \subseteq I$ is called *a 1CBP-placement of $J$*. For a 1CBP-placement $\pi_J$ of a subset $J \subseteq I$ and a real number $y \geq 0$, *the set $X(y, \pi_J)$ of intersecting rectangles at $y$-coordinate $y$* is defined as $X(y, \pi_J) = \{r_i \in J \mid y_i \leq y < y_i + h_i\}$. *The width $\xi(y, \pi_J)$* at a $y$-coordinate $y \geq 0$ of the placement $\pi_J$ is defined as $\xi(y, \pi_J) = \sum_{r_i \in X(y,\pi_J)} w_i$. Then the 1CBP problem without rotations is formulated as follows.

$$\begin{aligned} \text{minimize} \quad & H \\ \text{subject to} \quad & y_i + h_i \leq H, \quad r_i \in I, & (5) \\ & \xi(y, \{y_i \mid r_i \in I\}) \leq W, \quad 0 \leq y < H, & (6) \\ & y_i \geq 0, \quad r_i \in I. & (7) \end{aligned}$$

We say that a 1CBP-placement $\pi_I = \{y_i \mid r_i \in I\}$ is *feasible* within the strip $(W, H)$ if $\pi_I$ satisfies the constraints (5), (6), and (7). We denote the optimal value $H$ of a given instance $(I, W)$ without rotations by $OPT_{\text{1CBP}}(I, W)$. Note that $OPT_{\text{1CBP}}(I, W) \leq OPT_{\text{2SP}}(I, W)$ holds because 1CBP is a relaxed problem of 2SP.

Analogously with the decision version 2SPFH of 2SP, we consider a decision problem of 1CBP, called 1CBP with a fixed height (1CBPFH), which tests if there is a 1CBP-placement for all given rectangles in $I$ within the strip $(W, H)$.

## 3 Outline of the Entire Algorithm

In this section we introduce the outline of our entire algorithm ExACT1CBP to 1CBP. For simplicity, we consider 1CBP without rotations. Note that the algorithm for 1CBP with rotations of 90 degrees is obtained analogously. Given an instance $(I, W)$, ExACT1CBP$(I, W)$ finds the optimal height $H^\star$ and an optimal placement $\pi^\star$ using a procedure for computing a lower bound on $OPT_{\text{1CBP}}(I, W)$ and two procedures as follows:

- RestrictedStair: A heuristic algorithm for 2SPFH.
  Given an instance $(I, W, H)$ of 2SPFH, RestrictedStair returns a feasible solution if it succeed to find one. Otherwise, it returns a message "failure." Note that the message "failure" does not guarantee that the 2SPFH instance $(I, W, H)$ is infeasible. There may exist any feasible solution that RestrictedStair cannot find. Also note that the y-coordinates of a feasible solution of 2SPFH instance $(I, W, H)$ is also a feasible solution of 1CBPFH instance $(I, W, H)$ because 1CBPFH is a relaxed problem of 2SPFH. We omit the detail of RestrictedStair in this paper.

- BB-1CBPFH: An exact algorithm for 1CBPFH.
  Given an instance $(I, W, H)$ of 1CBPFH, BB-1CBPFH returns a feasible solution if any. Otherwise, it returns a message "infeasible." See Section 5 for detail.

ExACT1CBP first computes a lower bound using procedures given in Section 4. Then we set the height $H$ of the strip to the lower bound and consider $(I, W, H)$ as a 1CBPFH instance, which asks whether there is a feasible placement of $I$ within the strip $(W, H)$. For this problem, we first apply RestrictedStair$(I, W, H)$, which searches a restricted type of placements of $I$ attempting to find a feasible placement quickly. If RestrictedStair$(I, W, H)$ returns a feasible placement $\pi$ of $I$, then it is optimal and we halt. Note that Restricted-Stair$(I, W, H)$ may not find a feasible placement of $I$ even if $OPT_{\text{1CBP}}(I, W) = H$. Hence if it returns "failure," then we still need to check whether $OPT_{\text{1CBP}}(I, W) = H$ or not.

We then compute $H^\star = OPT_{\text{1CBP}}(I, W)$ using BB-1CBPFH. We first find a range $[l, u]$ such that $H^\star \in [l, u]$. We let $i = 0$, $l_0 = H$, $u_0 = H$, $\text{step}_0 = 1$, and apply BB-1CBPFH$(I, W, u_0)$ to check the feasibility of a 1CBPFH instance $(I, W, u_0)$. Until a 1CBPFH instance $(I, W, u_i)$ is feasible, we increment $i$ by 1 and let $l_{i+1} = u_i + 1$, $u_{i+1} = u_i + \text{step}_i$, and $\text{step}_{i+1} = \text{step}_i \times 2$. We then find $H^\star \in [l, u]$ by bisection method. ExACT1CBP is formally described in Algorithm 1.

## 4 Lower Bounds on Optimal Values

We use two lower bounds on $OPT_{\text{1CBP}}$ in our algorithm ExACT1CBP. One is a simple lower bound introduced by Martello et al. [1]. The other is one that we newly developed, called *partition lower bound.*

### 4.1 Simple Lower Bound

We first review two simple lower bounds by Martello et al. [1]. A lower bound $LB_h$ based on the heights of rectangles is defined by

$$LB_h(I, W) = \begin{cases} \max_{r_i \in I} h_i & \text{if rotations are not allowed,} \\ \max_{r_i \in I} \min\{w_i, h_i\} & \text{if rotations of 90 degrees are allowed.} \end{cases}$$

**Algorithm 1** : EXACT1CBP$(I, W)$

**Input:** An instance $(I, W)$ of a set of rectangles and a strip width $W$.
**Output:** A pair $(H^\star, \pi^\star)$ of the optimal value and an optimal placement.

```
1: Compute a lower bound $LB$ on $OPT_{1CBP}(I, W)$;
2: $H \leftarrow LB$;
3: if RESTRICTEDSTAIR$(I, W, H)$ returns a feasible placement $\pi^\star$ of $I$ within a strip $(W, H)$ then
4:    $H^\star \leftarrow H$;
5:    return $(H^\star, \pi^\star)$
6: end if;
7: $l \leftarrow H$; $u \leftarrow H$; step $\leftarrow 1$;
8: while BB-1CBPFH$(I, W, u)$ returns "infeasible" do
9:    $l \leftarrow u + 1$;
10:   $u \leftarrow u +$ step;
11:   step $\leftarrow$ step $\times 2$
12: end while;
13: while $l < u$ do
14:   $m \leftarrow \lfloor (l + u)/2 \rfloor$;
15:   if BB-1CBPFH$(I, W, m)$ returns a feasible placement $\pi^\star$ then
16:      $H^\star \leftarrow m$;
17:      $u \leftarrow m$;
18:   else
19:      $l \leftarrow m + 1$
20:   end if
21: end while;
22: return $(H^\star, \pi^\star)$.
```

A lower bound $LB_c$ based on the area of rectangles is defined by

$$LB_c(I, W) = \left\lceil \sum_{r_i \in I} w_i h_i / W \right\rceil ,$$

which is called *continuous lower bound.* By combining the two lower bounds, we obtain a lower bound $LB_0(I, W) = \max\{LB_h(I, W), LB_c(I, W)\}$. We easily see that $LB_0(I, W) \leq OPT(I, W)$ holds. This lower bound is used in the previous researches [1, 18, 19].

## 4.2 Partition Lower Bound

We propose a new lower bound *partition lower bound* for 1CBP without rotations based on "PARTITION" problem [21], which is known to be NP-complete. The partition lower bound can be obtained by taking a subset $J \subseteq I$ satisfying some conditions and computing a lower bound on $OPT_{\mathrm{1CBP}}(J, W)$. For a set $S$ of rectangles, let $h(S) = \sum_{r_i \in S} h_i$ denote the sum of the heights of all the rectangles in $S$.

**Lemma 1.** *For a given instance $(I, W)$ of 1CBP, let $J \subseteq I$ be a set of at least three rectangles such that the sum of widths of any three rectangles in $J$ is greater than $W$. For such a subset $J$, let $w' = \min_{r_i \in J} w_i$, $A_{W'} = \{r_i \in J \mid w_i + w' > W\}$ and $B_{W'} = J \setminus A_{W'}$, and define*

$$LB_{p0}(J, W) = h(A_{W'}) + \min\{h(B') \mid B' \subseteq B,\ h(B') \geq h(B)/2\}. \tag{8}$$

*Then $LB_{p0}(J, W) \leq OPT_{\mathrm{1CBP}}(J, W) \leq OPT_{\mathrm{1CBP}}(I, W)$ holds.* □

Note that the second term of the sum in (8) corresponds to the optimization version of PARTITION problem, which can be solved by dynamic programming.

Lemma 1 implies that

$$LB_p(I, W) = \max\{LB_{p0}(J, W) \mid J \subseteq I,\ w_i + w_j + w_k > W,\ \forall r_i, r_j, r_k \in J\}$$

is a lower bound on $OPT_{\mathrm{1CBP}}(I, W)$ as well, which we call *partition lower bound.* We can compute $LB_p$ in $O(n^2 h(I))$ time.

## 5 Branch-and-Bound Algorithm for 1CBPFH

We design algorithms BB-1CBPFH and RESTRICTEDSTAIR based on the branch-and-bound method. BB-1CBPFH is an exact algorithm for 1CBPFH and RESTRICTEDSTAIR is a heuristic algorithm for 2SPFH. Note that RESTRICTEDSTAIR searches a restricted type of placements to find a feasible solution quickly and may miss a feasible solution due to the restriction on the search space, but without the restriction, it performs as an exact algorithm.

In this section, we explain the canonical form of 1CBP-placements and the branching operation of BB-1CBPFH. We omit the details of bounding operations of BB-1CBPFH and the entire algorithm of RESTRICTEDSTAIR in this paper.

## 5.1 Branch-and-Bound method

The branch-and-bound method is one of the representative methodologies for designing exact algorithms for combinatorial optimization problems [1, 22, 23]. It is based on the idea that a problem instance can be solved by dividing it into partial problem instances and then solving all of them recursively. The operation of dividing a problem instance is called a *branching operation.*

In the branch-and-bound method, we check each partial problem instance before the division, and if we find that the instance has no feasible solution, we terminate the instance without the division. The operation of the termination of the partial problem instances is called a *bounding operation.*

For 1CBPFH, let $P_0(I, W, H)$ denote a given problem instance, which requires all rectangles in $I$ to be placed in a strip $(W, H)$. The process of applying branching operations can be expressed by a rooted tree, called a *search tree* rooted at the node that corresponds to $P_0(I, W, H)$ and the children of a node correspond to the partial problem instances generated by the branching operation applied to the node; thus each node in the search tree corresponds to a partial problem instance. Let $P_k(I, W, H)$ denote the $k$th partial problem instance generated during an execution of the branch-and-bound algorithm. Each instance $P_k(I, W, H)$, $k \geq 1$ is given by a placement $\pi_J = \{(x_i, y_i) \mid r_i \in J\}$ for a subset $J \subseteq I$, and the objective of instance $P_k(I, W, H)$ is to determine whether the remaining rectangles in $I \setminus J$ can be placed within the strip $(W, H)$ without changing the placement $\pi_J$. We call $P_k(I, W, H)$ *feasible* if all the remaining rectangles can be placed within the strip together with the placement $\pi_J$. For 1CBPFH, we use a branching operation that adds a rectangle $r_i \in I \setminus J$ to the placement $\pi_J$ of $J$ to form a placement $\pi_{J\cup\{i\}}$ of $J \cup \{i\}$.

If it turns out that $P_k(I, W, H)$ is feasible or infeasible for some reason on the information that has been obtained so far, then we can skip the generation of the partial problem instances from $P_k(I, W, H)$ without losing a chance to know the feasibility of the original problem instance $P_0(I, W, H)$. If $P_k(I, W, H)$ is feasible, then so is $P_0(I, W, H)$. If $P_k(I, W, H)$ is infeasible, then any partial problem generated from $P_k(I, W, H)$ is infeasible. In this case, we say that a *bounding operation terminates* $P_k(I, W, H)$. A partial problem instance is called *active* if it has been neither terminated nor divided into partial problem instances. The list of all active partial problem instances is maintained during the execution until either a partial problem instance is turned out to be feasible or no active partial problem instances are left. The entire search terminates concluding that $P_0(I, W, H)$ is feasible in the former case and infeasible in the latter. The basic components of the branch-and-bound method BB-1CBPFH for 1CBPFH are described as follows.

**Nodes:** The root node represents the empty strip, and a node of depth $d$ in a search tree represents a placement $\pi_J$ of a subset $J \subseteq I$ of $d$ rectangles.

**Branching operation:** A branch to a child from a node with a placement $\pi_J$ of a subset $J \subseteq I$ corresponds to placing a rectangle in $I \setminus J$ at a position in the open space of the current placement $\pi_J$. A branching operation generates those children corresponding to all possible positions of all rectangles in $I \setminus J$.

**Bounding operations:** If the algorithm finds that a partial problem does not have a feasible placement, it terminates the corresponding node. If it obtains a feasible placement at a leaf node, then the entire search terminates immediately since in this case the answer is yes.

**Search strategy:** We adopt the depth first search. The set of all active nodes, denoted by $A$, is maintained as a stack (an ordered list maintained with the last-in first-out rule); whenever the search moves on to a new active node, it chooses the node most recently added to $A$.

The entire framework of BB-1CBPFH$(I, W, H)$ is described in Algorithm 2.

**Algorithm 2** : BB-1CBPFH$(I, W, H)$

```
1: Renumber the indices of rectangles in $I$ according to the non-increasing order of widths
   (breaking ties by non-increasing heights);
2: $A \leftarrow \{P_0(I, W, H)\}$;
3: while $A \neq \emptyset$ do
4:   Let $u \in A$ be the node most recently added to $A$;
5:   if the 1CBP-placement $\pi_u$ corresponding to $u$ is a 1CBP-placement of $I$ then
6:     return $\pi_u$
7:   end if;
8:   if either $u$ has no new child node to be generated or is terminated by one of the
     bounding operations then
9:     Remove $u$ from $A$;
10:  else
11:    Generate a new child node $v$ of $u$ according to the branching operation in Section 5.4;
12:    Add the generated node $v$ to $A$
13:  end if
14: end while;
15: return "infeasible."
```

## 5.2 Canonical Form of 1CBP-placements

In this subsection we introduce the *canonical form for 1CBPFH without rotations.* Note that the canonical form with rotations of 90 degrees can be obtained analogously. *The key of* $r_i$ is defined by $(y_i, r_i)$. Let $\pi$ be a 1CBP-placement of $J \subseteq I$. *The code* $c(\pi)$ *of* $\pi$ is defined by the sequence $[(y_{t_1}, r_{t_1}), \ldots, (y_{t_n}, r_{t_n})]$ of keys of all rectangles in $I$ sorted in the lexicographical order, i.e., $y_{t_i} < y_{t_{i+1}}$ or $(y_{t_i} = y_{t_{i+1}}$ and $t_i < t_i)$ holds for all $i = 1, 2, \ldots, n-1$. We show an example of a code in Figure 2. Then the code $c(\pi)$ of $\pi$ is $[(0, r_3), (0, r_6), (1, r_7), (2, r_1), (2, r_4), (2, r_5), (3, r_2)]$.

A 1CBP-placement $\pi$ is called *canonical* if it has the lexicographically minimum code among all feasible 1CBP-placements of $I$. Let $\pi$ be a 1CBP-placement of a subset $J \subseteq I$. We call rectangle $r_i \in J$ is *bottom justified* if

$$y_i = 0 \quad \text{or} \quad \xi(y_i - 1, \pi) + w_i > W$$

holds. We call a 1CBP-placement $\pi$ of $J$ *bottom justified* if all rectangles in $J$ are bottom justified. We then have a following lemma.

**Lemma 2.** *Every canonical 1CBP-placement is bottom justified.* □

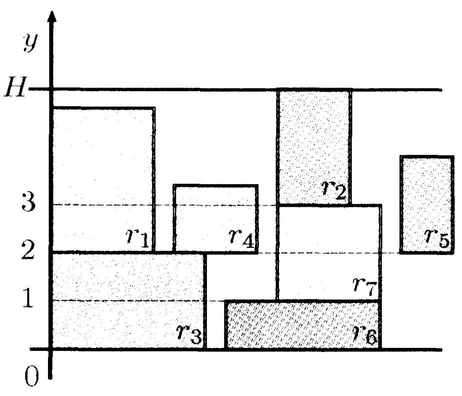

Figure 2: A 1CBP-placement $\pi$ of a rectangle set $J = \{r_1, r_2, \ldots, r_7\}$.

### 5.3 Parent 1CBP-Placement

Let $\pi_J$ be a 1CBP-placement of a subset $J \subseteq I$ and $\overline{y}(\pi_J) = \max\{y_i \mid r_i \in J\}$. We call a set $R$ of rectangles *removable* if $R = \{r_i \in J \mid y_i = \overline{y}(\pi_J)\}$. We also call the rectangle $r_i \in R$ with the maximum index *admissible*. *The parent 1CBP-placement of* $\pi$ is defined as the 1CBP-placement obtained from $\pi$ by removing the admissible rectangle. For any 1CBP-placement $\pi$, the parent 1CBP-placement of $\pi$ is uniquely defined, because the admissible rectangle for $\pi$ is uniquely defined.

**Lemma 3.** *Let $\pi$ be a 1CBP-placement of $J \subseteq I$ and $\pi'$ be the 1CBP-placement obtained from $\pi$ by removing the admissible rectangle $r \in J$. The 1CBP-placement $\pi$ can be obtained by placing rectangle $r$ at $y$-coordinate $y$ with $y \geq \overline{y}(\pi')$.* □

**Lemma 4.** *Let $\pi$ be a 1CBP-placement of $J \subseteq I$ and $\pi'$ be the 1CBP-placement obtained from $\pi$ by removing the admissible rectangle $r \in J$, i.e., $\pi'$ is the parent 1CBP-placement of $\pi$. If $\pi$ is bottom justified, then $\pi'$ is also bottom justified.* □

By Lemma 4, any bottom justified 1CBP-placement can be constructed from the empty 1CBP-placement by placing rectangles so that the resulting 1CBP-placement remains bottom justified after each 1CBP-placement.

### 5.4 Branching Rule for 1CBPFH

In the branch-and-bound method, we focus on searching the canonical 1CBP-placements in order to reduce the entire search space. We show a property of 1CBP-placements before describing a branching operation.

Let $\pi$ be a bottom justified 1CBP-placement of $J \subseteq I$ and $r \in I \setminus J$ be a rectangle. We say that a real $y \geq 0$ is *critical to rectangle* $r$ if the 1CBP-placement $\pi'$ obtained from $\pi$ by placing $r$ at $y$-coordinate $y$ is bottom justified. We then have a following lemma and design a branching operation based on the lemma.

**Lemma 5.** *For a 1CBP-placement $\pi_J$, each rectangle $r \in J$ has at most one critical value.* □

**Branching Operation 1:** For a node $u$ with its 1CBP-placement of a subset $J \subseteq I$, we always choose a rectangle $r$ from $I \setminus J$ which has a critical value to generate a child node of $u$ with a partial 1CBP-placement obtained by placing $r$ to $\pi$.

## 6 Experimental Results

We report the computational results on our algorithm EXACT1CBP. We coded the algorithm in the C++ language and used a PC with a Intel Xeon X5260 (3.3GHz) CPU and 16GB memory for computational experiments in this section. The time limit for each instance is set to one hour.

### 6.1 Instances

We used 2SP instances available at DEIS – Operations Research Group Library of Instances[1]. The instances used for our computational experiments are categorized into five groups "ht," "beng," "gcut," "cgcut" and "ngcut," where ht = {ht01, ..., ht09}, beng = {beng01, ..., beng10}, gcut = {gcut01, ..., gcut04}, cgcut = {cgcut01, ..., cgcut03} and ngcut = {ngcut01, ..., ngcut12}. The instances were used in the computational experiments in [1, 18, 19].

### 6.2 Experimental Results

In the tables in this subsection, column '$n$' shows number of rectangles, column '$W$' shows widths of a strip, and column '$H^*$' shows optimal values. Column '$T_R$' shows the total computation times in seconds of RESTRICTEDSTAIR and column '$T_1$' shows the total computation times in seconds of BB-1CBPFH. The mark 'T.O.' means that the search did not stop within the time limit. If RESTRICTEDSTAIR found a feasible placement or BB-1CBPFH did not terminate in the time limit, '–' is written in the column '$T_1$.' If EXACT1CBP could not find the optimal value, we write '–' in the column '$H^*$.'

Table 1 and Table 2 are the computation results with and without rotations of 90 degrees, respectively. We observe that our heuristics approach was effective. We succeeded to obtain the optimal solutions for gcut02 and cgcut02 without rotations, which have not been solved by the existing algorithms.

Martello et al. [1] and Alvarez-Valdez et al. [18] proposed exact algorithms for 1CBP. However, since they did not separate the computation of 1CBP from the entire algorithm for 2SP, we avoid comparing their results with ours.

## 7 Conclusion

We proposed an exact algorithm for 1CBP. The algorithm consists of two procedures, an exact algorithm for 1CBP, called BB-1CBPFH and a heuristics approach to 2SPFH, called RESTRICTEDSTAIR. Each procedure is designed based on the branch-and-bound method using canonical forms. We also propose a new lower bound, called partition lower bound. Through computational experiments, we confirmed that the proposed algorithm is effective. It is a future work to design an efficient heuristics to 1CBPFH.

[1]http://www.or.deis.unibo.it/research_pages/ORinstances/ORinstances.htm

Table 1: Computation results with rotations of 90 degrees

| | $n$ | $W$ | $H^*$ | $T_R$ | $T_1$ |
|---|---|---|---|---|---|
| ht01 | 16 | 20 | 20 | 0.00 | – |
| ht02 | 17 | 20 | 20 | 0.00 | – |
| ht03 | 16 | 40 | 20 | 0.01 | – |
| ht04 | 25 | 40 | 15 | 0.00 | – |
| ht05 | 25 | 40 | 15 | 0.00 | – |
| ht06 | 25 | 5 | 15 | 0.01 | – |
| ht07 | 28 | 60 | 30 | 0.10 | – |
| ht08 | 29 | 60 | 30 | 0.08 | – |
| ht09 | 28 | 60 | 30 | 0.01 | – |
| beng01 | 20 | 25 | 30 | 0.00 | – |
| beng02 | 40 | 25 | 57 | 0.00 | – |
| beng03 | 60 | 25 | 84 | 0.25 | – |
| beng04 | 80 | 25 | 107 | 0.02 | – |
| beng05 | 100 | 25 | 134 | 0.86 | – |
| beng06 | 40 | 25 | 36 | 0.00 | – |
| beng07 | 80 | 40 | 67 | 0.00 | – |
| beng08 | 120 | 40 | 101 | 0.01 | – |
| beng09 | 160 | 40 | 126 | 0.03 | – |
| beng10 | 200 | 40 | 156 | 0.31 | – |
| gcut01 | 10 | 250 | 696 | 0.00 | 0.08 |
| gcut02 | 20 | 250 | – | 0.26 | T.O. |
| gcut03 | 30 | 250 | – | 1.72 | T.O. |
| gcut04 | 50 | 250 | – | 2943.73 | T.O. |
| cgcut01 | 16 | 10 | 23 | 0.00 | – |
| cgcut02 | 23 | 70 | 63 | 0.01 | – |
| cgcut03 | 62 | 70 | – | T.O. | – |
| ngcut01 | 10 | 10 | 20 | 0.01 | 0.00 |
| ngcut02 | 17 | 10 | 28 | 0.00 | – |
| ngcut03 | 21 | 10 | 28 | 0.00 | – |
| ngcut04 | 7 | 10 | 18 | 0.00 | 0.00 |
| ngcut05 | 14 | 10 | 36 | 0.00 | – |
| ngcut06 | 15 | 10 | 29 | 0.00 | – |
| ngcut07 | 8 | 20 | 10 | 0.00 | 0.00 |
| ngcut08 | 13 | 20 | 33 | 0.46 | 0.06 |
| ngcut09 | 18 | 20 | 49 | 0.01 | – |
| ngcut10 | 13 | 30 | 59 | 0.03 | 0.00 |
| ngcut11 | 15 | 30 | 51 | 2.73 | 1.21 |
| ngcut12 | 22 | 30 | 77 | 0.18 | – |

The experiments are conducted on a Xeon X5260 (3.3GHz) CPU with the time limit of 3600 secs.

Table 2: Computation results without rotations of 90 degrees

| | $n$ | $W$ | $H^*$ | $T_R$ | $T_1$ |
|---|---|---|---|---|---|
| ht01 | 16 | 20 | 20 | 0.00 | – |
| ht02 | 17 | 20 | 20 | 0.00 | – |
| ht03 | 16 | 40 | 20 | 0.00 | – |
| ht04 | 25 | 40 | 15 | 0.00 | – |
| ht05 | 25 | 40 | 15 | 0.00 | – |
| ht06 | 25 | 5 | 15 | 0.00 | – |
| ht07 | 28 | 60 | 30 | 0.01 | – |
| ht08 | 29 | 60 | 30 | 23.37 | – |
| ht09 | 28 | 60 | 30 | 0.00 | – |
| beng01 | 20 | 25 | 30 | 0.25 | – |
| beng02 | 40 | 25 | 57 | 5.43 | – |
| beng03 | 60 | 25 | 84 | 0.02 | – |
| beng04 | 80 | 25 | 107 | 0.01 | – |
| beng05 | 100 | 25 | 134 | 0.03 | – |
| beng06 | 40 | 25 | 36 | 0.00 | – |
| beng07 | 80 | 40 | 67 | 0.04 | – |
| beng08 | 120 | 40 | 101 | 0.01 | – |
| beng09 | 160 | 40 | 126 | 3.65 | – |
| beng10 | 200 | 40 | 156 | 0.25 | – |
| gcut01 | 10 | 250 | 1016 | 0.00 | 0.00 |
| gcut02 | 20 | 250 | 1187 | 0.70 | 202.10 |
| gcut03 | 30 | 250 | 1803 | 0.09 | – |
| gcut04 | 50 | 250 | – | 6.18 | T.O. |
| cgcut01 | 16 | 10 | 23 | 0.00 | – |
| cgcut02 | 23 | 70 | 64 | 278.59 | 515.45 |
| cgcut03 | 62 | 70 | – | T.O. | – |
| ngcut01 | 10 | 10 | 23 | 0.00 | 0.00 |
| ngcut02 | 17 | 10 | 30 | 0.03 | 0.03 |
| ngcut03 | 21 | 10 | 28 | 0.00 | – |
| ngcut04 | 7 | 10 | 20 | 0.00 | 0.00 |
| ngcut05 | 14 | 10 | 36 | 0.00 | – |
| ngcut06 | 15 | 10 | 31 | 0.05 | 0.11 |
| ngcut07 | 8 | 20 | 20 | 0.00 | – |
| ngcut08 | 13 | 20 | 33 | 0.02 | 0.00 |
| ngcut09 | 18 | 20 | 50 | 2.71 | 0.52 |
| ngcut10 | 13 | 30 | 80 | 0.00 | 0.01 |
| ngcut11 | 15 | 30 | 52 | 0.01 | 0.01 |
| ngcut12 | 22 | 30 | 87 | 0.00 | – |

The experiments are conducted on a Xeon X5260 (3.3GHz) CPU with the time limit of 3600 secs.

## Acknowledgments

This research was partially supported by a Research Fellowships of the Japan Society for the Promotion of Science for Young Scientists and a Scientific Grant in Aid from the Ministry of Education, Science, Sports and Culture of Japan.

## References

[1] S. Martello, M. Monaci, and D. Vigo, "An exact approach to the strip-packing problem," *INFORMS Journal on Computing*, vol. 15, no. 3, pp. 310–319, 2003.

[2] G. Wäscher, H. Haußner, and H. Schumann, "An improved typology of cutting and packing problems," *European Journal of Operational Research*, vol. 183, no. 3, pp. 1109–1130, 2007.

[3] J. Turek, J. L. Wolf, and P. S. Yu, "Approximate algorithms for scheduling parallelizable tasks," in *SPAA 1992: Proceedings of the 4th Annual ACM Symposium on Parallel Algorithms and Architectures*, (New York, NY, USA), pp. 323–332, ACM Press, 1992.

[4] E. Hopper and B. C. H. Turton, "A review of the application of meta-heuristic algorithms to 2D strip packing problems," *Artificial Intelligence Review*, vol. 16, no. 4, pp. 257–300, 2001.

[5] S. Imahori, M. Yagiura, and T. Ibaraki, "Local search algorithms for the rectangle packing problem with general spatial costs," *Mathematical Programming*, vol. 97, no. 3, pp. 543–569, 2003.

[6] B. S. Baker, E. G. Coffman, Jr., and R. L. Rivest, "Orthogonal packings in two dimensions," *SIAM Journal on Computing*, vol. 9, no. 4, pp. 846–855, 1980.

[7] B. Chazelle, "The bottomn-left bin-packing heuristic: An efficient implementation," *IEEE Transactions on Computers*, vol. 32, no. 8, pp. 697–707, 1983.

[8] E. K. Burke, G. Kendall, and G. Whitwell, "A new placement heuristic for the orthogonal stock-cutting problem," *Operations Research*, vol. 52, no. 4, pp. 655–671, 2004.

[9] E. Hopper and B. C. H. Turton, "An empirical investigation of meta-heuristic and heuristic algorithms for a 2D packing problem," *European Journal of Operational Research*, vol. 128, no. 1, pp. 34–57, 2001.

[10] M. Iori, S. Martello, and M. Monaci, *Metaheuristic Algorithms for the Strip Packing Problem*, vol. 78 of *Applied Optimization*, ch. 7. Springer, 2003.

[11] S. Jakobs, "On genetic algorithms for the packing of polygons," *European Journal of Operational Research*, vol. 88, no. 1, pp. 165–181, 1996.

[12] D. Liu and H. Teng, "An improved bl-algorithm for genetic algorithm of the orthogonal packing of rectangles," *European Journal of Operational Research*, vol. 112, no. 2, pp. 413–420, 1999.

[13] L. H. W. Yeung and W. K. S. Tang, "Strip-packing using hybrid genetic approach," *Engineering Applications of Artificial Intelligence*, vol. 17, no. 2, pp. 169–177, 2004.

[14] R. Alvarez-Valdes, F. Parreño, and J. M. Tamarit, "Reactive grasp for the strip-packing problem," *Computers & Operations Research*, vol. 35, no. 4, pp. 1065–1083, 2008.

[15] H. Murata, K. Fujiyoshi, S. Nakatake, and Y. Kajitani, "VLSI module placement based on rectangle-packing by the sequence-pair," *IEEE Translational on Computer-Aided Design of Integrated Circuits and Systems*, vol. 15, no. 12, pp. 1518–1524, 1996.

[16] S. Imahori, M. Yagiura, and T. Ibaraki, "Improved local search algorithms for the rectangle packing problem with general spatial costs," *European Journal of Operational Research*, vol. 167, no. 1, pp. 48–67, 2005.

[17] S. Imahori, M. Yagiura, and H. Nagamochi, *Practical Algorithms for Two-Dimensional Packing*, vol. 10 of *Chapman & Hall/CRC computer & information science series*, ch. 36. CRC Press, 2007.

[18] R. Alvarez-Valdes, F. Parreño, and J. M. Tamarit, "A branch and bound algorithm for the strip packing problem," *OR Spectrum*, vol. 31, no. 2, pp. 431–459, 2009.

[19] M. Kenmochi, T. Imamichi, K. Nonobe, M. Yagiura, and H. Nagamochi, "Exact algorithms for the two-dimensional strip packing problem with and without rotations," *European Journal of Operational Research*, vol. 198, no. 1, pp. 73–83, 2009.

[20] A. Lodi, S. Martello, and M. Monaci, "Two-dimensional packing problems: A survey," *European Journal of Operational Research*, vol. 141, no. 2, pp. 241–252, 2002.

[21] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.

[22] T. Ibaraki, "Enumerative approaches to combinatorial optimization," *Annals of Operations Research*, vol. 10–11, no. 1–4, pp. 3–342, 1988.

[23] N. Lesh, J. Marks, A. Mcmahon, and M. Mitzenmacher, "Exhaustive approaches to 2D rectangular perfect packings," *Information Processing Letters*, vol. 90, no. 1, pp. 7–14, 2004.