ISSN 0005-1179, Automation and Remote Control, 2011, Vol. 72, No. 1, pp. 141–159. © Pleiades Publishing, Ltd., 2011.
Original Russian Text © M.A. Mesyagutov, E.A. Mukhacheva, G.N. Belov, G. Scheithauer, 2011, published in Avtomatika i Telemekhanika, 2011, No. 1, pp. 154–173.

Packing of One-dimensional Bins with Contiguous Selection of Identical Items: An Exact Method of Optimal Solution

M. A. Mesyagutov,* E. A. Mukhacheva,* G. N. Belov, and G. Scheithauer****

**Ufa State Aviation Technical University, Ufa, Russia*
***Technical University of Dresden, Dresden, Germany*

Received July 15, 2009

Abstract—Consideration was given to the one-dimensional bin packing problem under the conditions for *heterogeneity* of the items put into bins and *contiguity* of choosing identical items for the next bin. The branch-and-bound method using the "next fit" principle and the "linear programming" method were proposed to solve it. The problem and its solution may be used to construct an improved lower bound in the problem of two-dimensional packing.

DOI: 10.1134/S0005117911010127

1. INTRODUCTION

The following problem of one-dimensional packing is considered as the basis: needed is to place items of identical width w_i and length $l_i = 1$, $i \in I := \{1, \ldots, m\}$, into bins of identical width W and length $L = 1$ meeting at that the following conditions.

(1^0) Only different items may be placed in the same bin (*heterogeneity*).
(2^0) If not all items of type $i \in I$ are packed in bins, then the type of the packed item is retained for placing in the next bin (*contiguity of choice*).

At that, any two items are regarded as different even if their widths coincide. Needed is to *minimize* the number N of the packed bins.

With the above formulation, this problem was considered by different independent researchers as the one-dimensional contiguous cutting stock (1CCS) problem [1–3]. The problem of item placement which precedes the physical process of cutting will be called the one-dimensional contiguous bin packing (1CBP) problem. Its solution is representable as a sequence of packed bins with indication of their number χ_j for each singe-type packing (see Fig. 1a).

In what follows, we represent the one-dimensional contiguous packing without breaks of bins as shown in Fig. 1b which may be interpreted as placement of bins on a pallet. In distinction to the *bin packing* (BP), we call it for brevity the *bin allocation* (BA). For convenience of presentation we follow below this symbology and images for 1CBP using bin placement on the pallets. This problem was considered by the present authors in connection with developing the metaheuristics with a local lower bound to solve the problem of two-dimensional packing [4].

Figure 1c in essence depicts the two-dimensional packing of the rectangulars outlined by bold lines, which cannot be said about the placement in Fig. 1b. This packing is decomposed into unit bins representing contiguous one-dimensional packings where, for example, item 3 occupies different positions in the first and second bins. Solution of the 1CBP (1CCS) problem was shown [4] to be the lower bound of two-dimensional packing of the set $\{\overline{w}, \overline{l}\}$ of rectangulars of size $\overline{w}_i = w_i$, $\overline{l}_i = b_i$, and the permissible solution of the 1CBP problem as determined by the "next fit" (NF) algorithm

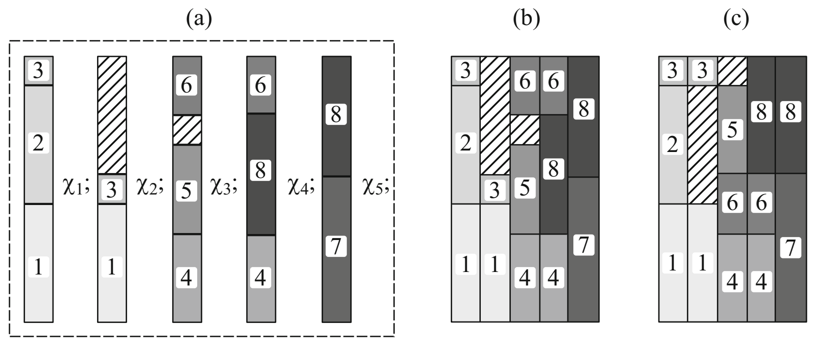

Fig. 1. Sequence of packings with the condition for contiguity of choice of the packed items: (a) With bin discontinuity; (b) placement of the pallet is not an outline of the two-dimensional packing; (c) placement on the pallet.

is the lower bound for the permissible solution of the 2SP problem in a neighborhood constrained by the permutations of some items packed in the same bin [5].

The 1CBP problem has applications in their own rights such as the scheduling problems [6]. Let us assume that there is a computer system. We interpret its computational resource as the bin length and the tasks to be calculated as the packed items whose width represents the consumed computing power of the problem, and the need for items, the time of its solution. Needed is to solve problems using the computer system during the minimum time. This problem represents the cumulative non-preemptive resource scheduling.

2. MATHEMATICAL MODELS

We present the mathematical models of one-dimensional and two-dimensional packings. The problem of one-dimensional packing (1BP) is considered as the basic one. The main problem is that of contiguous one-dimensional packing (1CBP) which is called also the *orthogonally oriented* problem because it is used in the block technology to solve the problem of two-dimensional orthogonal packing [1, 3, 4].

The 1BP problem is defined by the information vector (W, m, w, b), where W is the bin width, m is the number of different items, $w = (w_1, \dots, w_i, \dots, w_m)$, w_i is the width of item of kind $i \in I$, $b = (b_1, \dots, b_i, \dots, b_m)$, where b_i is the number of items of kind $i \in I$. All initial data are integers.

The packing vector

$$\alpha^j := (a_{1j}, \dots, a_{mj})^{\intercal}$$

having the Boolean components $a_{ij} = 1$ if the item i is placed in the bin or $a_{ij} = 0$, otherwise, is assigned to the packing (j) in a one-dimensional bin. The vector α^j indicates the items packed in the jth bin. The following is valid for the permissible packing of the one-dimensional bin α^j:

$$\sum_{i=1}^{m} w_i a_{ij} \leqslant W. \tag{1}$$

To fix positions of the items in a bin, it is advisable to use the packing *pattern*

$$p^j := ((1(j), y_1), \dots, (i(j), y_i), \dots), \tag{2}$$

where $i(j)$ is the number of the item occupying the position i in the pattern j and y_i is the minimal ordinate of the item $i(j)$. The packing (j) and/or its pattern are denoted below by p^j.

Definition 1. Two packings p^u and p^v are referred to as equivalent if $\alpha^u = \alpha^v$.

Let N be the number of bins where all items can be placed. The matrix $A = (a_{ij})$, where $i = 1, \ldots, m$, $j = 1, \ldots, N$, is called the packing matrix. Then, the problem of optimal packing of the one-dimensional bins comes to the following integer problem:

$$\min N = \left\{ \sum_{j=1}^{N} f_j \mid f = (f_1, \ldots, f_N) \in \mathbb{Z}_+^N;\ \sum_{j=1}^{N} f_j \alpha^j = b \right\}, \tag{3}$$

where $f_j = 1 \vee 0$ indicates whether the packing j is used or not. The pair (f, N) defines the permissible (optimal) solution of the 1BP problem. The vectors α^j restore the determined solutions. It is independent either of the order of item packing in the bin or of their coordinates. The patterns like (2) may be conveniently used to restore the packings.

The 1BP problem was shown to be $\mathcal{NP}$-hard [7]. It is solved using the methods of exhaustive search such as the branch-and-bound method [8], relaxation by linear programming [9], and meta-heuristics [10, 11] which reject the unpromising variants. A special case of 1BP, the 1CBP problem with the information vector for 1BP is here the main problem.

We denote by $I = \{1, \ldots, m\}$ the set of all packed items, I_j the set of items packed in the jth bin, I_j^+ the set of those items whose stock is exhausted after packing in the jth bin, and I_j^- the set of items whose packing was begun in the jth bin. Then, the additional conditions in the contiguous problem are as follows:

(1^0) *heterogeneity of items.* The items $i(j)$ of each cortege I_j are different;
(2^0) *contiguity of items.* If any of the items $i(j) \notin I_j^+$, then $i(j) \in I_{j+1}$.

Obviously, the set of permissible solutions of the 1CBP problem is a subset of the solutions of the 1BP problem.

A hybrid branch-and-bound method with reduced enumeration specified by constraints (1^0) and (2^0) and proposed estimate of branching obtained through the linear programming method is suggested here to solve 1CBP. The branching strategy was also changed in this method into a strategy based on solving a linear-programming problem.

The problems 1CBP and 1BP are structurally similar.

We consider the following two-dimensional strip packing problem (2SP) with the initial information (W, m, w, l), where W is the strip width (strip length $L = \infty$), m is the number of rectangular items; $w = (w_1, \ldots, w_m)$ and $l = (l_1, \ldots, l_m)$, (w_i, l_i) being the width and length of the ith item.

We introduce a rectangular coordinate system where the axis OX coincides with the horizontal face of the bin and the axis OY, with the vertical face. Then, solution of the problem 2SP is representable as the set

$$\mathrm{RP} := \{(x_1, y_1), \ldots, (x_m, y_m)\},$$

where (x_i, y_i) are the minimal coordinates of the item i. The set RP is called the *permissible orthogonal packing* if the following conditions are satisfied:

- orthogonal placement of items in the packing:
  for (x_i, y_i), $i = 1, \ldots, m$ and any other vertex (x_i^k, y_i^k) of the ith item,

$$((x_i^k = x_i) \vee (x_i^k = x_i + l_i)) \wedge ((y_i^k = y_i) \vee (y_i^k = y_i + w_i)); \tag{4}$$

- nonoverlapping of items:
  $\forall i, j \in I$, $i \neq j$:

$$((x_i \geqslant x_j + l_j) \vee (x_j \geqslant x_i + l_i)) \vee ((y_i \geqslant y_j + w_j) \vee (y_j \geqslant y_i + w_i)); \tag{5}$$

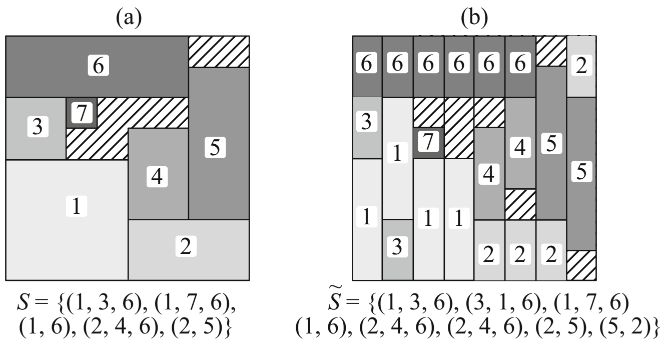
(a)
(b)
6
3
7
5
4
1
2
S = {(1, 3, 6), (1, 7, 6), (1, 6), (2, 4, 6), (2, 5)}
S̃ = {(1, 3, 6), (3, 1, 6), (1, 7, 6) (1, 6), (2, 4, 6), (2, 4, 6), (2, 5), (5, 2)}

Fig. 2. Block diagrams of (a) rectangular packing and (b) one-dimensional contiguous packing.

- nonoverlapping of items and strip faces:
  $\forall i \in I$:

$$(x_i \geqslant 0) \wedge (y_i \geqslant 0) \wedge ((y_i + w_i) \leqslant W). \tag{6}$$

Problem 2SP. Needed is to minimize the length

$$L = \max_{i \in I} (x_i + l_i) \tag{7}$$

of the occupied part of strip on the set of permissible packings satisfying conditions (4)–(6).

E.A. Mukhacheva and A.S. Filippova proposed a decomposition of packing into vertical blocks underlying the block technology of modeling the heuristic algorithms [1, 3, 5, 11]. A similar horizontal decomposition introduced by V.M. Kartak enabled creation of transparent criteria for non-intersection of rectangulars [4]. The packing block structure and its corresponding set of packing patterns are shown in Fig. 2a. One of its corresponding contiguous packings 1CBP is shown in Fig. 2b.

Definition 2. The maximal set of equivalent patterns of packings $p^1, \ldots, p^\chi$ with the packing vector α is called the block and denoted by $\tilde{S} = (\alpha, \chi)$, where χ is the block cardinality (length).

Figure 2b shows five blocks: $\tilde{S_1} = (\alpha^1, 2)$, $\tilde{S_2} = (\alpha^2, 1)$, $\tilde{S_3} = (\alpha^3, 1)$, $\tilde{S_4} = (\alpha^4, 2)$, $\tilde{S_5} = (\alpha^5, 2)$. At that, two different patterns correspond to each of the vectors α^1, α^4, and α^5. In case (a) these pairs are identical.

In case (b), merging of the equivalent one-dimensional packings provides the set S instead of $\tilde{S}$. At that, nothing changes here from the point of view of the one-dimensional packing. However, to transform $\tilde{S}$ into a two-dimensional packing S one has to carry out topological permutations by a method similar to the algorithm described in [12]. At the same time, it is easy to demonstrate that the optimal solution of the 1CBP problem is the lower bound of the solution of the 2BP problem. Therefore, the 1CBP problem plays an important role in searching the optimum for the two-dimensional packing problem.

Each 1CBP-packing may be identified with its block structure, but not vice versa. The entire set of block structures is wider than the set of block structures with the contiguity condition. Therefore, all aforementioned block structures satisfy the contiguity condition so that the notions of packing and its block structure are synonymic.

A precise algorithm based on the branch-and-bound method with permutations of items is proposed to solve the 1CBP problem. To estimate the promises of subproblems, the LP-relaxation of the problem of one-dimensional cutting with different types of rods is used. A combination of the strategy of the best boundary and the object-oriented branching strategy based on the linear programming with the given types of stock defined by the priority parameter is proposed as the strategy of branching. In what follows, all terms refer to the 1CBP problem.

3. REDUCTION OF PROBLEM DIMENSIONALITY

Branching is preceded by processing the initial information which is independent of the method of solution and invariant to it. Sometimes one succeeds in finding on the set of items of the 2SP problem a subset such that a certain combination of its items necessarily appears in the optimal solution. Therefore, if some parts of the optimal solution are recognized at the initial stage, then they can be disregarded in what follows, thus reducing the problem dimensionality. This procedure if called the *preprocessing*.

3.1. Reduction Technique

Let us consider the first part of packing (Fig. 3). A situation has occurred where the total length of all items of width $W-1$ exceeds the total length of all items of width 1. Obviously, a combination of the items of width 1 located over the items of width $W-1$ is not worse than any other combination of these items. Additionally, they cannot combine with any other items. This combination enters the optimal solution namely in this manner. Therefore, they can be removed before branching. However, in the second part of packing this assertion is not valid for the set of items of the widths $W-2$ and 2. Yet if they are considered together with the items $W-3$ and 3, then a similar situation occurs where the total length of all "wide" items exceeds the total length of all "narrow" items. The combinations of items of the kind of those shown in Fig. 3 enter the optimum and may be disregarded.

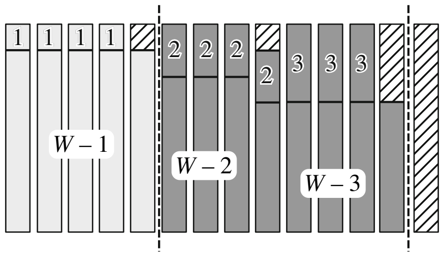

Fig. 3. Preprocessing: Removal of items knowingly included in the optimal solution.

We denote by $I(w) = \{i \in I : \ w_i = w\}$ the set of all items of the width w. If there is $\widehat{W} < W/2$ for which the inequality system

$$\begin{cases} \sum\limits_{w=1}^{\widetilde{w}} \sum\limits_{i \in I(w)} l_i \geqslant \sum\limits_{w=1}^{\widetilde{w}} \sum\limits_{i \in I(W-w)} l_i, & 1 \leqslant \widetilde{w} \leqslant \widehat{W}-1 \\ \sum\limits_{w=1}^{\widehat{W}} \sum\limits_{i \in I(w)} l_i \leqslant \sum\limits_{w=1}^{\widehat{W}} \sum\limits_{i \in I(W-w)} l_i, & \end{cases} \qquad (8)$$

is valid, then the set of items

$$I_1 := \bigcup_{w=1}^{\widehat{W}} (I(w) \cup I(W-w))$$

can be disregarded in the form of a combination of the kind depicted in Fig. 3, it contains H_1 rods:

$$H_1 := \sum_{w=1}^{\widehat{W}} \sum_{i \in I(W-w)} l_i.$$

Therefore, the problem size may be reduced by $|I_1|$ units so as to make the problem optimum $\mathrm{OPT}(I)$ equal to

$$\mathrm{OPT}(I) := H_1 + \mathrm{OPT}(I \backslash I_1). \tag{9}$$

After its reduction, the set $I := I \backslash I_1$ may still have items satisfying the inequality system (8). Therefore, the preprocessing procedure must be repeated as long as there exists $\widehat{W} < W/2$ such that these inequalities are satisfied.

4. BRANCHINGS

In this section we discuss a method for enumeration of a subset of all dense packings where at least one problem optimum is available. This subset is structured as a tree-like scheme whose nodes represent permissible partial packings and leaves, the permissible packings of the entire item set I. In this section we discuss the dominance criteria and the rules for truncation of the nodes that are equivalent and may be disregarded.

Let us consider the priority list of items $\langle i_1, \ldots, i_m \rangle$, $i_l \in I$, $l = 1, \ldots, m$ representing the order of successive placing of each element of I in the bin. The pair i_l and i_{l+1}, where $i_l, i_{l+1} \in I$, means that the item i_{l+1} is placed next after i_l. Therefore, the item i_{l+1} is placed at the lowermost of the left permissible positions in the bin, but not earlier than the preceding placed item i_l. If the item i_{l+1} cannot go into the bin together with the item i_l, then the item i_{l+1} is located in the following bins. At that, if in the current bin there still exists the need for an item, then in the following bins it is downshifted as much as possible. This rule of bin packing is called "next fit" (NF).

The fact of getting a bin packing by applying the NF rule to the priority list $\langle i_1, \ldots, i_m \rangle$, $i_l \in I$, $l = 1, \ldots, m$ is denoted by $\mathrm{NF}(\langle i_1, \ldots, i_m \rangle)$. At that, the set of permutations of the elements on the list $\langle i_1, \ldots, i_m \rangle$ defines the space of problem solution. Each permutation can be transformed in a permissible packing so that at least one of them corresponds to the optimal solution of the 1CBP problem.

4.1. Tree of Branchings

The idea of branching of lies in successive and oriented placement of each element of the set I enabling one to construct the branching tree $T = (V, E)$ where the tree nodes V are connected by the edges from the set E. The leaves of the tree T represent the set of all permutations of the elements of I.

The ordered sequence

$$u = \langle i_1, \ldots, i_d \rangle$$

of the placed items $i_l \in I,\ l = 1, \ldots, d;\ d \leqslant m$ corresponds to each node $u \in V$. Each node $u \in V$ of the tree T is related with its partial packing (block structure [11]):

$$\mathrm{NF}(u) = \langle (\alpha^1, \chi_1), \ldots, (\alpha^{\sigma(u)}, \chi_{\sigma(u)}) \rangle,$$

where the items $\bar{I}(u) = I \backslash \{i_1, \ldots, i_d\}$ must be placed and α is the packing vector. The edge (u, v), where $u, v \in V,\ u \neq v$, belongs to the set E if the subproblem v is obtained from u by placing the item $i \in \bar{I}(u)$, that is, $v = \langle u, i \rangle$. The next item $i \in \bar{I}(u)$ selected for the node $u \in V$ is placed in the blocks

$$\langle (\alpha^{j(u)}, \chi_{j(u)}), \ldots, (\alpha^{\sigma(u)}, \chi_{\sigma(u)}) \rangle \tag{10}$$

according to the NF rule and inequality (1), i_d is the last element placed in u which was placed in the block $(\alpha^{j(u)}, \chi_{j(u)})$, that is,

$$\alpha_{i_d}^{j(u)} = 1 \ \wedge \ \alpha_{i_d}^{j(u)-1} = 0.$$

If $i \in \bar{I}(u)$ cannot be fit in the blocks (10) according to condition (1), then a new block $\sigma(u) + 1$ is initialized to place it.

The root of the tree u_0 lies at the level $d = 0$ and has no placed items $u_0 = \langle \emptyset \rangle$. Therefore, the tree root has no partial packing. Each item $i \in I \backslash \{\emptyset\}$ can be placed. Thus, the root of the tree T has m successors. The partial packing of the nodes at the level $d = 1$ differs from the root u_0 in the placed item i. Each node at the depth $d = 1, \ldots, m-1$ creates $m - d$ successors at the level $d + 1$. Consequently, the number of nodes at the level d is as follows:

$$m \times (m-1) \times \ldots \times (m-d+1) = \frac{1 \times 2 \times \ldots \times (m-d) \times (m-d+1) \times \ldots \times m}{1 \times 2 \times \ldots \times (m-d)} = \frac{m!}{(m-d)!}.$$

4.2. Truncation Rule 1: Elimination of the Vertically Equivalent Packings

Let us consider Fig. 4b. If in the first block items 2 and 3 are interchanged, then the packing shown in Fig. 4a results. They are equivalent from the standpoint of one-dimensional packing; and there is no need to consider both. Yet the branching method based on the priority lists relates them with different nodes of the branching tree $u_a = \langle 1, 2, 3, 4 \rangle$ and $u_b = \langle 1, 3, 2, 4 \rangle$. Consideration is given only to the node u_a.

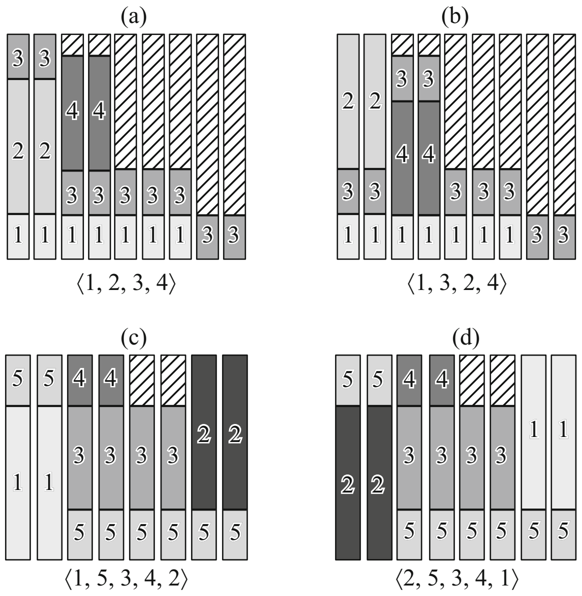

Fig. 4. Packing of items: (a) Dominant packing; (b) item packing equivalent vertically to packing *a*; (c) dominant packing ; (d) item packing equivalent vertically to packing *c*.

Let given be the node $u \in V$ with its partial packing $\mathrm{SS}(u) = \mathrm{NF}(u)$, where each block (α^j, χ_j), $j = 1, \ldots, \sigma(u)$ contains the items

$$I_j = \{i \in I : \ \alpha_i^j = 1\}.$$

We denote by

$$I_j^- = \{i \in I_j : \ i \notin I_{j-1})\}$$

the set of items that were placed for the first time in the block $S_j = (\alpha^j, \chi_j)$. Then, for the node $u \in V$ there are $\mathrm{SM}(u)$ equivalent packings

$$\mathrm{SM}(u) = \prod_{j=1}^{\sigma(u)} \left|I_j^-\right|!.$$

For example, the node $u = \langle 1, 2, 3, 4\rangle$ with the block structure $\mathrm{SS}(u) = \langle(\alpha^1, 2), (\alpha^2, 1), (\alpha^3, 4), (\alpha^4, 2)\rangle$ has $\mathrm{SM}(u) = 3! \times 1! \times 0! \times 0! = 6$ equivalent packings in Figs. 4a and 4b.

The following rule is applied in the course of branching to eliminate from consideration the vertically equivalent packings. Given is the node $u = \langle i_1, \ldots, i_d\rangle \in V$ with its block structure $\mathrm{SS}(u)$ where the item i_d was placed in the block $S_{j(u)} \in \mathrm{SS}(u) : \ i_d \in I_{j(u)}^-$. If for the next placed item $i^* \in \bar{I}(u)$ and block $j = j(u)$ the inequalities

$$w_{i^*} \leqslant W - \sum_{i \in I} w_i \times \alpha_i^{j(u)}, \tag{11}$$

$$i^* < \max\left\{k : \ k \in I_{j(u)}^-\right\} \tag{12}$$

are satisfied, then the node $v = \langle i_1, \ldots, i_d, i^*\rangle$ may be truncated. If inequality (11) is satisfied and (12) is not satisfied, then the item i^* must be placed. If (11) is not satisfied, then go to the next block $j := j + 1$ and repeat the truncation rule. The situation where nevertheless the item i^* gets into the block S_j was or will be considered in other branches of the tree T.

For example, if the first block in Fig. 4b is partially formed $\bar{\alpha}^1 = \{(1, 0), (3, w_1)\}$, then the item $i^* = 2$ satisfies condition (11) but cannot be placed there because inequality (12) is violated. The node u_b may be disregarded.

4.3. Truncation Rule 2: Elimination of the Horizontally Equivalent Packings

Let us consider the packing shown in Fig. 4d. If items 1 and 2 are interchanged, then the packing shown in Fig. 4c results. From the standpoint of one-dimensional packing they are equivalent, and there is no need for considering both. Yet the method of branching based on the priority lists relates these packings with different nodes $u_a = \langle 1, 5, 3, 4, 2\rangle$ and $u_b = \langle 2, 5, 3, 4, 1\rangle$. Consideration is given only to the node u_a.

The following rule is used to avoid studying of additional nodes at branching. First of all, the widest item

$$i^\star := \arg\max_{i \in I}\{w_i\}$$

is selected. If there is more than one widest item, then the set of widest items is generated, and that of the greatest area is taken from them. The following rule is used in the course of branching. Let for the block $u \in V$ there be a block S_c in the packing $\mathrm{SS}(u) = \mathrm{NF}(u)$ such that

$$S_c \in \mathrm{SS}(u) : \ i^\star \in I_c^-,$$

and for it

$$\sum_{j=1}^{c-1} \chi_j > \frac{ub(I)}{2} - \frac{l_{i^\star}}{2}$$

be satisfied, where $ub(I)$ is the upper bound of the problem, then the node u may be truncated.

This empirical rule is based on the assumption that all packings where the item $i^\star$ lies to the left of the vertical line dividing the packing in two have equivalent packings where the item $i^\star$ lies to the right of this line.

4.4. Truncation Rule 3: Elimination of the Equivalent Packings for Identical Items

We consider Fig. 5b. Items j and i are placed in different blocks, but j was placed before i. If items j and i are interchanged, the packing shown in Fig. 5a results. Therefore, these packings are equivalent relative to items i and j and identical. Only the node $u_a = \langle i_1, \ldots, i, \ldots, j, \ldots, i_d \rangle$ is considered instead of the two nodes.

Let for $i \in I$ there be a set of its identical items:

$$I^s(i) = \{j \in I \backslash \{i\} : \ w_i = w_j \wedge l_i = l_j\}.$$

If for the node $u = \{i_1, \ldots, i_d\} \in V$ with its partial packing $\mathrm{SS}(u) = \mathrm{NF}(u)$ and item $i \in I$ there is $j \in I^s(i) : \ i < j$ such that

$$\begin{cases} i \in I^-_{k_2} \\ j \in I^-_{k_1}, \end{cases}$$

$S_{k_1}, S_{k_2} \in \mathrm{SS}(u)$, and at that $k_2 > k_1$, then the node u may be disregarded.

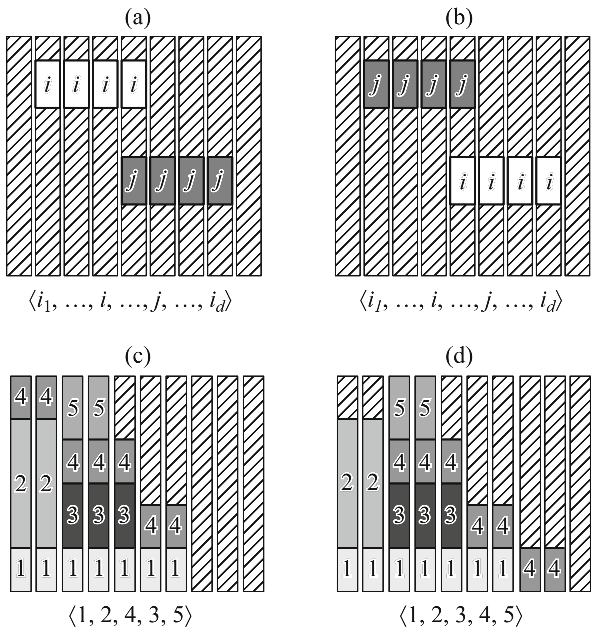

Fig. 5. Item packing: (a) and (b) are equivalent packings relative to the identical items i and j; (c) is the dominating packing; (d) item 4 may be placed in the preceding block.

The following rule is used to disregard the equivalent packings. Let given be the node $u = \langle i_1, \dots, i_d \rangle \in V$ with its partial packing $\mathrm{SS}(u) = \mathrm{NF}(u)$ where item $i \in \bar{I}(u)$ must be placed next on the list. If there is $k \in \{1, \dots, d\}$ such that $i_k \in I^s(i)$ and $i_k > i$, then the block $\langle i_1, \dots, i_d, i \rangle$ may be disregarded.

4.5. Truncation Rule 4: An Item Is Placed in Packing Before Its Placement

Let consideration be given to the block $u = \langle i_1, \dots, i_d \rangle \in V$ having the partial packing $\mathrm{SS}(u) = \mathrm{NF}(u)$. If there exists an item $i^* \in \bar{I}(u)$ and the set $\widetilde{S} \subseteq \mathrm{SS}(u) \backslash \{S_{j(u)}, \dots, S_{\sigma(u)}\} = \{S_{k_1}, \dots, S_{k_2}\}$ of successive blocks in $\mathrm{SS}(u)$ such that the inequalities

$$\begin{cases} l_{i^*} \leqslant \sum\limits_{j=k_1}^{k_2} \chi_j \\ w_{i^*} \leqslant \left(W - \sum\limits_{i \in I_j} w_i \right), \quad \forall j \in \{k_1, \dots, k_2\} \end{cases}$$

are satisfied, then the item i^* fits completely the blocks $\widetilde{S}$ of the partial packing $\mathrm{SS}(u)$, and there is no reason to consider the node u. This relies on the assumption that the packing obtained from placing the item i^* further than $\widetilde{S}$ will not be better than the packing where it is placed in $\widetilde{S}$.

4.6. Truncation Rule 5: Truncation of Nodes According to the First Fit Rule

Let us consider Fig. 5d. In the partial packing of the node u_b, item 3 is placed next after item 2. Yet it does not go to the empty space over items 1 and 2 and is placed in the second block. The next item 4 is placed also in the second block. Although it fits the first block as well. The packing where item 4 is placed in the first block (see Fig. 5c) is not inferior to the packing where item 3 is placed for the first time in the second block. Therefore, there is no need to consider both nodes, but only the node $u_a = \langle 1, 2, 4, 3, 5 \rangle$.

Let the node $u = \langle i_1, \dots, i_d \rangle \in V$ be given together with its partial packing $\mathrm{SS}(u) = \mathrm{NF}(u)$. If the item i_d fits in the empty domain of the preceding block,

$$W - \sum_{i \in I} w_i \alpha_i^{j(u)-1} \geqslant w_{i_d},$$

then the node u may be disregarded.

4.7. Truncation Rule 6: Material Truncation

For each permissible packing of length L, it is always possible to calculate the bound of the residue in the entire packing as

$$\Delta(L) = WL - \sum_i w_i l_i.$$

The lengths of the considered permissible packings lie within the interval $[lb(I), \dots, ub(I)]$, where $lb(I)$ and $ub(I)$ are, respectively, the lower and upper bounds.

Let given be the node $u = \langle i_1, \dots, i_d \rangle \in V$ with its partial packing $\mathrm{SS}(u) = \mathrm{NF}(u)$, where the item i_d is placed in the block $S_{j(u)} : \ i_d \in I^-_{j(u)}$. If the inequality

$$\sum_{j=1}^{j(u)-1} \chi_j \left(W - \sum_{i \in I_j} w_i \right) \geqslant \Delta(ub(I)) \tag{13}$$

is satisfied for the node u, then the packing resulting from the node u has the length at least not smaller than $ub(I)$. The nodes satisfying inequality (13) may be disregarded.

5. LOWER BOUNDS

The present section is devoted to two types of the lower bounds: obvious material lower bound and that relying on the linear programming-based relaxation of the problem of one-dimensional cutting with rods of different types [9]. Both are used despite the fact that the latter dominates the former. For each node, first the material lower bound is calculated and then, if the node is not truncated, the LP bound. Thus, the excessive calculations of the LP bound are avoided.

5.1. Material Lower Bound

All items of the set I must be packed into the strip. Consequently, they may occupy a domain not shorter than

$$lb_{\mathrm{m}} = \left\lceil \frac{\sum\limits_{i\in I} w_i l_i}{W} \right\rceil . \tag{14}$$

The pure form of (14) may be used for the entire set of items,—for example, at the root of the branching tree where no item is placed.

Let given be the node $u = \langle i_1, \ldots, i_d \rangle \in V$, $i_l \in I$, $l = 1, \ldots, d$, $d \leqslant m$ of the block structure $\mathrm{SS}(u) = \langle (\alpha^1, \chi_1), \ldots, (\alpha^{\sigma(u)}, \chi_{\sigma(u)}) \rangle$, where the item i_d is placed in the block $S_{j(u)} \in \mathrm{SS}(u)$: $i_d \in I^-_{j(u)}$. In the partial packing, the already placed items $\{i_1, \ldots, i_d\}$ create unused domains where the remaining items $\bar{I}(u) = I \backslash \{i_1, \ldots, i_d\}$ may be placed. Therefore, the area of the created part of packing must be summed together with the area of the remaining items $\bar{I}(u)$, which should occupy a length not shorter than

$$lb_{\mathrm{m}}(u) = \sum_{j=1}^{j(u)-1} \chi_j + \left\lceil \frac{\sum\limits_{j=j(u)}^{\sigma(u)} \left(\chi_j \sum\limits_{i\in I_j} w_i \right) + \sum\limits_{i\in \bar{I}(u)} w_i l_i}{W} \right\rceil . \tag{15}$$

A formula like (15) may be used in any branch to calculate the material lower bound.

5.2. Linear Programming-based Lower Bound

Figure 6 depicts the block structure $\mathrm{SS}(u) = \mathrm{NF}(u)$ of the node $u = \langle i_1, \ldots, i_d \rangle \in V$ where the items $i_1, \ldots, i_d$ are already placed and the item i_d is in the block $(\alpha^{j(u)}, \chi_{j(u)})$. These items create an empty unused space where the remaining items $\bar{I}(u) = I \backslash \{i_1, \ldots, i_d\}$ can be placed. Beginning from the block $S_{j(u)}$, this space has the following dimensions:

$$\begin{cases} (\lambda_1, u_1) & = \left(W - \sum\limits_{i\in I_{j(u)}} w_i;\ \chi_{j(u)} \right) \\ & \cdots \\ (\lambda_q, u_q) & = \left(W - \sum\limits_{i\in I_{\sigma(u)}} w_i;\ \chi_{\sigma(u)} \right) \\ (\lambda_{q+1}, u_{q+1}) & = (W;\ \infty), \end{cases} \tag{16}$$

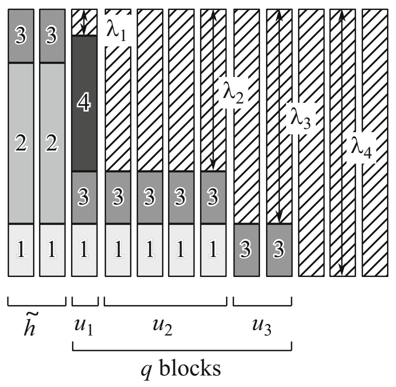

Fig. 6. Partial packing.

where $q = \sigma(u) - j(u) + 1$. According to the problem discreteness, the unused area of the width λ_k and the length u_k may regarded as the set of u_k pieces of one-dimensional rods of the material of the width λ_k and the length 1. Therefore, each type of the rod λ_k is available in the amount of u_k except the last λ_{q+1} which is available in an unlimited amount. The remaining set of items $w_i \times 1$, $i \in \bar{I}(u)$ in number l_i with unit length may be cut from this set of rods according to the following rules. Each item $w_i \times 1$ may be cut only once from one rod. Completeness of each type of rods cannot be exceeded.

Any permissible cutting pattern obeys the following binary vector

$$a^{jk} = (a_{1jk}, \ldots, a_{mjk})^{\mathrm{T}} \in \{0,1\}^m \quad \text{for} \quad \sum_{i \in I} w_i a_{ijk} \leqslant \lambda_k, \tag{17}$$

where $a_{ijk} = 0$, $i \in I(u)$, $j \in J_k$ is the set of patterns for each type of rods $k = 1, \ldots, q(u) + 1$.

We formulate the following problem of optimization 1CSPM(u) for the node $u = \langle i_1, \ldots, i_d \rangle \in V$:

$$z = \sum_j x_{j,q(u)+1} \longrightarrow \min, \tag{18}$$

provided that

$$\sum_{k=1}^{q(u)+1} \sum_j a_{ijk} x_{jk} = l_i, \qquad i \in \bar{I}(u), \tag{19}$$

$$l_i = 0, \qquad i \in I(u), \tag{20}$$

$$\sum_j x_{jk} \leqslant u_k, \qquad k \in K := \{1, \ldots, q(u)\}, \tag{21}$$

$$x_{jk} \in \mathbb{Z}_+, \qquad k \in K \cup \{q+1\}, \quad j \in J_k. \tag{22}$$

This problem is known as the cutting stock problem with multiple stock lengths (1CSPM) [9]. The optimal value z^* of the objective function of the problem 1CSPM(u) is the minimal number of rods required to cut all items from the set of rods. Therefore, the value of z^* may be used as the lower bound for packing the remaining items $\bar{I}(u)$.

The continuous linear programming-based relaxation is used in practice for the problem 1CSPM(u). Instead of the integrality condition for the variables (22), the real variables

$$x_{jk} \in \mathbb{R}_+, \quad k \in K \cup \{q+1\}, \; j \in J_k \tag{23}$$

are used. Therefore, the model LP(u) (18)–(21), (23) is a continuous relaxation of the problem 1CSPM(u) which is solved by the method of column generation with solution of the following problems of the "0–1 knapsack:"

$$\bar{c}_k = -\max\left\{ d_{m+k} + \sum_{i\in I(u)} d_i a_{ijk} : \sum_{i\in I(u)} w_i a_{ijk} \leqslant \lambda_k \right\}, \quad k \in K,$$

$$\bar{c}_{q+1} = 1 - \max\left\{ \sum_{i\in I(u)} d_i a_{ijk} : \sum_{i\in I(u)} w_i a_{ijk} \leqslant \lambda_k \right\},$$

where d_i is the dual estimate of the constraint of i. At each step, $q(u)+1$ problems are solved, and a column $\arg\min_{k\in K}\{\bar{c}_k, \bar{c}_{q+1}\}$ is added.

We denote by z^*_{LP} the value of the optimal solution $X^{\mathrm{LP}} = \{x_{jk} : x_{jk} \geqslant \epsilon, \epsilon > 0\}$ of the problem 1CSPM(u) and accept

$$lb_{\mathrm{LP}}(u) = \sum_{j=1}^{\sigma(u)} \chi_j + \lceil z^*_{\mathrm{LP}}(u) \rceil \tag{24}$$

as the lower bound for the nodes of the branching tree.

5.3. Truncation Rule 7: Verification of Relaxation for Contiguity

After solution of the problem LP(u) for the node $u \in V$, a situation may occur where the set of columns in the optimal solution of the relaxation $A = \{a_{ijk} : x_{jk} \geqslant \epsilon, \epsilon > 0\}$ already features contiguity of selection of the identical items or they can be ordered in compliance with this condition. Then, there is no reason to consider and divide the node u because the optimal integral solution for the given branch of the branching tree T has already been obtained.

To relate the extreme column of the first-type rods with the extreme column of the second-type rods, and so on, it is necessary to add rows of zeros and ones, for example, in the next manner for the three items and three types of rods with one pattern for each:

$$A' = \begin{array}{ccc} \lambda_1 & \lambda_2 & \lambda_3 \\ a_{111} & a_{112} & a_{113} \\ a_{211} & a_{212} & a_{213} \\ a_{311} & a_{312} & a_{313} \\ \hline 1 & 1 & 0 \\ 0 & 1 & 1 \end{array}.$$

The PQ-Trees algorithm of Booth and Leuker [13] was used to verify the conditions for contiguous selection of the identical items for A'. Its complexity is linear in the number of the nonzero elements in the matrix A.

6. PRECISE ALGORITHM

There is no need to consider the nodes whose lower bounds are greater than or equal to the upper bound for the problem 1CBP. Therefore, for some node $u = \{i_1, \ldots, i_d\} \in V$ and its successors $v = \langle i_1, \ldots, i_d, i \rangle$, where $i \in \bar{I}(u)$, the edge $(u, v) \in E$ only if

$$lb(v) < ub(I). \tag{25}$$

Created are not all successors of the node u but only those satisfying condition (25). Therefore, the earlier the upper bound is improved, the smaller number of nodes is possibly created and processed. Consequently, the speed of the branch-and-bound method depends directly on the strategy of determining the upper bound.

The strategies of the "good" upper bound are mostly heuristics underlying the rule of what branches must be ramified first. In the present section, discussed is an heuristic rule based on the linear programming-based solution of the relaxation problem.

6.1. Best Bound Strategy

If all nodes with a certain value of the lower bound are processed, then only the nodes with greater lower bounds remain. Therefore, all subproblems with the lower bounds smaller than this value must be processed to improve the lower bound to a certain value.

Algorithm 1. The best bound strategy.

Input data: the set of nodes V.

(BB1) Construct the set of all nodes with the least lower bound $V^{\min}$:

$$V^{\min} = \left\{ v \in V : \ lb(v) = \min_{u \in V} lb(u) \right\}.$$

If $V^{\min} = \emptyset$, then Exit; otherwise, (BB2).

(BB2) Ramify the highest of the nodes $v^* \in V^{\min}$:

$$v^* = \arg \min_{v \in V^{\min}} |v|.$$

Go to (BB1).

6.2. Domain-oriented Branching Strategy Based on Linear Programming with a Given Priority Parameter

All nodes of the branching tree T are divided into the priority nodes and those without priority.

Definition 3. Let the node $u = \langle i_1, \dots, i_d \rangle$ of the branching tree T be given. Then, the successor $v = \langle u, i \rangle$ of the node u, $i \in \bar{I}(u)$, is called the priority node for the given parameter $r \in \mathbb{Z}_+$ if there exists a column a^{jk} such that

$$x^{\mathrm{LP}}_{jk} > \epsilon, \quad \epsilon > 0, \quad a_{ijk} = 1, \quad k \leqslant r,$$

where X^{LP} is the optimal solution of the problem 1CSPM(u).

It was shown in Section 4.1 that at each step of branching at most $(m-d)$ new nodes are created at the depth d. Each of them theoretically can be a priority one, that is, in the course of branching on the tree T the number of the priority nodes is not constant. New priority nodes can originate in the course of division of the priority and nonpriority nodes.

Algorithm 2. Domain-oriented branching strategy based on linear programming with a given priority parameter.

Input data: the set of nodes V, priority parameter $r \in \{1, \dots, W\}$.

(LP1) Construct the set of all priority nodes $V^+ \subseteq V$. If $V^+ = \emptyset$, then (LP2). Ramify $v^* \in V^+$:

(1) construct the set of deepest nodes
$V^+_{\max} = \{v \in V^+ : \ |v| = \max_{u \in V^+} |u|\}$;

(2) among the deepest nodes, branch those with the least lower bound $v^* = \arg\min_{v\in V^+_{\max}} lb_{\mathrm{LP}}(v)$.

Go to (LP1).

(LP2) Construct the set of all nonpriority nodes V. If $V = \emptyset$, then Exit. Otherwise, ramify $v^* \in V$ according to the best bound strategy $v^* = \mathrm{BestBound}(V)$. Go to (LP1).

7. COMPUTER EXPERIMENT

In view of the fact that the solution of the 1CBP problem is the lower bound of 2SP, the computer experiment was carried out on the 2SP problems of the Bottled problem library [14], and the results were compared with the known lower bounds. The software was realized in *C++* and compiled by *gcc 4.1.2*. The computer experiments were carried out on *AMD Opteron 250* (2.4 GHz).

The library consists of 500 examples that were generated randomly and divided into 10 classes. The problems of the six first classes were given by Berkey and Wang [15]:

- Class 1: $W = 10$, w_i, l_i uniformly distributed on [1, 10];
- Class 2: $W = 30$, w_i, l_i uniformly distributed on [1, 10];
- Class 3: $W = 40$, w_i, l_i uniformly distributed on [1, 35];
- Class 4: $W = 100$, w_i, l_i uniformly distributed on [1, 35];
- Class 5: $W = 100$, w_i, l_i uniformly distributed on [1, 100];
- Class 6: $W = 300$, w_i, l_i uniformly distributed on [1, 100].

In each class, the dimensions of items were generated over identical intervals.

Martello and Vigo [16] gave the following classes of problems that are very close to the real problems. The items were divided into four types:

- Type 1: w_i uniformly distributed on $\left[\frac{2}{3}W, W\right]$, l_i uniformly distributed on $\left[1, \frac{1}{2}W\right]$;
- Type 2: w_i uniformly distributed on $\left[1, \frac{1}{2}W\right]$, l_i uniformly distributed on $\left[\frac{2}{3}W, W\right]$;
- Type 3: w_i uniformly distributed on $\left[\frac{1}{2}W, W\right]$, l_i uniformly distributed on $\left[\frac{1}{2}W, W\right]$;
- Type 4: w_i uniformly distributed on $\left[1, \frac{1}{2}W\right]$, l_i uniformly distributed on $\left[1, \frac{1}{2}W\right]$.

By generating these types of items in various proportions, the four remaining classes of problems were specified:

- Class 7: type 1 with the probability 70%; type 2, 3, 4 with the probability 10%;
- Class 8: type 2 with the probability 70%; type 1, 3, 4 with the probability 10%;
- Class 9: type 3 with the probability 70%; type 1, 2, 4 with the probability 10%;
- Class 10: type 4 with the probability 70%; type 1, 2, 3 with the probability 10%.

For each class, the strip width was $W = 100$. The number of items in each class was, respectively, 20, 40, 60, 80, and 100. Ten problems were generated for each number of items.

For each example, Bortfeld also suggested a lower bound whose mean value is compiled in Table 1 in the `Bortfeld(B)` column. The rounded upward length of the optimal solution of the continuous relaxation of the problem of one-dimensional cutting of rods `LP` and the linear programming-based improved lower bound `Probing(P)` with enumeration of the forbidden pairs of items [17] are presented for reference. The column `1CBP(C)` compiles the mean value of the solutions of 1CBP of which the column `opt` indicates the number of optimal solutions, and for the rest of the problems the minimal value of the lower bound among all unprocessed nodes is presented. The problem runtime was constrained to 3 min.

The cases where one of the bounds exceeds others are compiled in Table 2 from which one can see that in 178, 197, and 184 cases the LP and Probing lower bounds and the precise solution of the 1CBP problem exceed, respectively, the Bortfeld lower bound. Despite the fact that the total

Table 1. Mean values of the lower bounds on the Berkey–Wang and Martello–Vigo problems

Class	N	Bortfeld(B)	LP	Probing(P)	1CBP(C)	opt
cl01	50	187.18	187.68	187.70	187.74	33
cl02	50	60.52	60.52	60.52	60.52	33
cl03	50	504.12	507.62	507.88	507.84	12
cl04	50	193.50	193.50	193.62	193.50	0
cl05	50	1613.16	1630.66	1631.68	1631.72	26
cl06	50	506.40	506.40	506.98	506.40	0
cl07	50	1577.82	1588.76	1590.40	1591.24	46
cl08	50	1397.92	1399.58	1399.90	1399.84	0
cl09	50	3343.10	3344.56	3346.04	3346.16	50
cl10	50	909.16	917.58	917.64	917.70	6
	500	1029.29	1033.69	1034.25	1034.27	206

Table 2. Comparison of the magnitudes of Bortfeld(B), LP, Probing(P), 1CBP(C) lower bounds

Class	LP>B	P>B	P>LP	P>C	C>B	C>LP	C>P
cl01	15	15	1	0	**16**	**3**	**2**
cl02	0	0	0	0	0	0	0
cl03	33	**36**	**7**	**6**	**36**	6	4
cl04	0	**5**	**5**	**5**	0	0	0
cl05	44	**45**	6	1	**45**	**10**	**6**
cl06	0	**8**	**8**	**8**	0	0	0
cl07	27	**27**	7	1	**27**	**16**	**10**
cl08	15	**17**	5	2	16	**6**	**4**
cl09	5	**5**	2	0	**5**	**4**	**2**
cl10	39	**39**	**2**	**2**	**39**	1	1
	178	**197**	43	25	184	**46**	**29**

number of examples where the Probing lower bound proved to be superior to 1CBP, the latter exceeds that of Probing on the four last examples. As can be seen from the columns P>C and C>P, for the library problems the total improvement of the lower bounds was reached by Probing and 1CBP on different examples. Detailed results of solution of the 1CBP problem on the Bortfeld problem are given in Table 3. The examples were decomposed into classes relative to the number of problems.

Results of the numerical experiment:

- No example was solved optimally on the classes cl04, cl06, and cl08. They can be designated as the most difficult ones for the algorithm developed. On the contrary, practically all problems from the classes cl07 and cl09 were solved optimally. Almost a half of the problems were solved optimally, which indicates to the satisfactory performance of the algorithm.
- The lower bound of 1CBP always is not less than the Bortfeld and LP bounds.
- By considering the solution of 1CBP as the lower bound of 2SP, one may conclude that in 197 cases the Probing lower bound proved to be superior to that of Bortfeld, which exceeds by 13 the similar result for 1CBP. Despite this fact, in 46 cases the latter exceeded LP, which exceeds by three the similar result for Probing. Let us compare the bounds of 1CBP and Probing. In 29 cases 1CBP was superior to Probing, and in 25 cases the latter exceeded 1CBP. If one considers separately the results for each problem, then it is possible to conclude that the majority of improvements was attained on different classes, which does not eliminate the results of one of the bounds, especially in the light of the fact that the optimal values of

Table 3. Results of solving 1CBP on the Martello–Vigo and Berkey–Wang problems

Size	Class	Precisely	Mean time, s	Mean number of nodes	Not solved
20	cl01	8	38.1	19 461.2	2
	cl02	10	0	0	0
	cl03	1	162	29 399.3	9
	cl04	0	180.3	2508.1	10
	cl05	6	80.6	6165.8	4
	cl06	0	180.9	500.5	10
	cl07	9	19.5	1705.9	1
	cl08	0	180	7698.9	10
	cl09	10	0	0	0
	cl10	4	108	5671.6	6
	Total:	48	94.94	7311.1	52
40	cl01	7	54	37 945.1	3
	cl02	10	0	0	0
	cl03	4	108.1	18 352.7	6
	cl04	0	181.8	1385.6	10
	cl05	6	72.7	1753.1	4
	cl06	0	189.4	201.7	10
	cl07	10	0	6.9	0
	cl08	0	180.4	2510.7	10
	cl09	10	0.5	135.2	0
	cl10	1	162.3	8573.2	9
	Total:	48	94.92	7086.4	52
60	cl01	7	54	32 694.2	3
	cl02	3	126	3671.4	7
	cl03	2	144	14 336.9	8
	cl04	0	185.2	740.4	10
	cl05	5	92	3214.6	5
	cl06	0	224.3	119.2	10
	cl07	9	18	5008.9	1
	cl08	0	185.9	1162.3	10
	cl09	10	0	0	0
	cl10	1	164.2	3176	9
	Total:	37	119.36	6412.3	63
80	cl01	8	36.1	22 378.5	2
	cl02	5	90.2	1599.2	5
	cl03	3	126.9	12 101.2	7
	cl04	0	187.8	467.8	10
	cl05	6	73.3	4397.9	4
	cl06	0	249.3	79.8	10
	cl07	10	0.2	22.4	0
	cl08	0	187.3	703.3	10
	cl09	10	0	0.3	0
	cl10	0	192.5	1108	10
	Total:	42	114.36	4285.8	58
100	cl01	3	126	67 802.2	7
	cl02	5	91.4	863.9	5
	cl03	2	145.8	6643.4	8
	cl04	0	196.5	291.9	10
	cl05	3	128.1	3055.2	7
	cl06	0	467.7	99.7	10
	cl07	8	36.4	4495.1	2
	cl08	0	198.3	483.8	10
	cl09	10	0	0.3	0
	cl10	0	192.5	668.2	10
	Total:	31	158.27	8440.3	69
	Total:	206	116.37	6707.2	294

the library problems is not known. Therefore, the algorithms to solve the 1CBP and Probing problems may be regarded as alternatives, and both may be used if necessary.

8. CONCLUSIONS

The present paper proposed and realized a precise method for solution of the problem of packing one-dimensional bins with the contiguous choice of identical items based on the branch-and-bound method with permutation of items. Relying on the fact that the solution of the main problem is the lower bound for the problem of two-dimensional packing, efficiency of the proposed algorithm was studied on the two-dimensional packing problems. Despite high (20–100) dimensionality of the problems, 40% of them were solved optimally. The well-known bounds were used to compare the quality of the solution obtained according to the comparison of the lower bounds for the two-dimensional packing. Although improvements were reached on various examples, the best results were obtained by the proposed algorithm to solve the 1CBP problem and by the Probing procedure. At that, the paths to improve the proposed algorithm are visible. For example, it is advisable to complete relaxation of subproblems to the permissible solution by means of a simple heuristic. Another path is represented by using simpler lower bounds such as the dual permissible functions [18, 19].

ACKNOWLEDGMENTS

This work was supported by the Russian Foundation for Basic Research, project no. 10-07-91330-HHUO_a.

REFERENCES

1. Mukhacheva, E.A., Mukhacheva, A.S., and Chiglintsev, A.V., Block-structure Genetic Algorithm in the Problems of Two-dimensional Packing, *Inf. Tekhnol.*, 1999, no. 11, pp. 13–18.

2. Martello, S., Monachi, M., and Vigo, D., An Exact Approach to the Strip-Packing Problem, *INFORMS J. Comput.*, 2003, no. 15(3), pp. 310–319.

3. Mukhacheva, E.A. and Mukhacheva, A.S., The Rectangular Packing Problem: Local Optimum Search Methods Based on Block Structures, *Autom. Remote Control*, 2004, no. 2, pp. 248–257.

4. Kartak, V.M., Mesyagutov, M.A., Mukhacheva, E.A., and Filippova, A.S., Local Search of Orthogonal Packings Using the Lower Bounds, *Autom. Remote Control*, 2009, no. 6, pp. 1054–1066.

5. Zhitnikov, V.P. and Filippova, A.S., Problem of Orthogonal Packing in Half-infinite Strip: Search in the Neighborhood of the Local Lower Bound, *Inf. Tekhnol.*, 2007, no. 5, pp. 55–62.

6. Hartmann, S., *Project Scheduling under Limited Resources*, Berlin: Springer, 2000.

7. Garey, M.L. and Johnson, D.S., *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco: Freeman, 1979. Translated under the title *Vychislitel'nye machiny i trudnoreshaemye zadachi*, Moscow: Mir, 1982.

8. Martello, S. and Toth, P., *Knapsack Problems: Algorithms and Computer Implementations*, Chichester: Wiley, 1990.

9. Belov, G. and Scheithauer, G., A Cutting Plane Algorithm for the One-dimensional Cutting Stock Problem with Multiple Stock Lengths, *Eur. Oper. Res.*, 2002, no. 14(2), pp. 274–294.

10. Hopper, E. and Turton, B., A Review of the Application of Metaheuristic Algorithms to 2D Regular and Irregular Strip Packing Problems, *Artific. Intelligent.*, 2001, no. 16, pp. 257–300.

11. Belov, G., Scheithauer, G., and Mukhacheva, E.A., One-dimensional Heuristics Adapted for Two-dimensional Rectangular Strip Packing, *J. Oper. Res. Soc.*, 2008, no. 59, pp. 823–832.

12. Mukhacheva, E.A. and Nazarov, D.A., Design of Rectangular Packings: Block Structure-based Reconstruction Algorithm, *Autom. Remote Control*, 2008, no. 2, pp. 262–277.

13. Booth, K. and Lueker, G., Testing for the Consecutive Ones Property, Interval Graphs, and Planarity Using PQ-Tree Algorithms, *J. Computat. Sys. Sci.*, 1976, no. 13, pp. 335–379.

14. Bortfeld, A., A Genetic Algorithm for the Two-dimensional Strip Packing Problem with Rectangular Prices, *Eur. J. Oper. Res.*, 2006, no. 172, pp. 814–837.

15. Berkey, J. and Wang, P., Two Dimensional Finite Bin Packing Algorithms, *J. Oper. Res. Soc.*, 1987, no. 38, pp. 423–429.

16. Martello, S. and Vigo, D., Exact Solution of the Two-dimensional Finite Bin Packing Problem, *Manage. Sci.*, 1998, no. 44, pp. 388–399.

17. Kartak, V., Refreshed Lower Bound for the Problem of Packing Rectangulars in Half-infinite Strip, *Vest. UGATU, Ser. Upravlen. Vychislit. Tekhn. Informat.*, 2008, no. 10, 2(27), pp. 154–159.

18. Baldacci, R. and Boschetti, M., A Cutting-plane Approach for the Two-dimensional Orthogonal Non-guillotine Cutting Problem, *Eur. J. Oper. Res.*, 2007, no. 183(3), pp. 1136–1149.

19. Fekete, S. and Schepers, J., A General Framework for Bounds for Higher-dimensional Orthogonal Packing Problems, *Math. Met. Oper. Res.*, 2004, no. 60, pp. 311–329.

This paper was recommended for publication by P.Yu. Chebotarev, a member of the Editorial Board