Als Manuskript gedruckt

Technische Universität Dresden
Herausgeber: Der Rektor

Equivalence and Dominance for Problems of Optimal Packing of Rectangles

Guntram Scheithauer

MATH-NM-04-1995

February 1995

Abstract

In this paper we consider the problem of optimal packing of small rectangles within a larger rectangle.

The topic consists of a comprehensive investigation of equivalence and dominance of packing patterns. Using suitable kinds of equivalence and dominance, the goal is to develop an efficient branch&bound algorithm with which problems of practical interest may be solved exactly.

Contents

1 Introduction

In this paper we consider the problem of optimal packing of small rectangles (so-called pieces) within a larger rectangle (so-called pallet). This problem is investigated in a lot of publications (cf. [5], [11], [12]) because of its large practical relevance in both the two-dimensional and three-dimensional case. Using the classification of DYCKHOFF [4], the considered problem is of type 2/B/O/F.

Since the well-known knapsack problem is involved in the packing problem, the latter also belongs to the class of $\mathcal{NP}$-hard problems. Because the number of different packing patterns increases exponentially with the number of pieces, results for solving problems of medium size are only reported in the literature for heuristic algorithms. These circumstances are founded by a lack of investigations with respect to equivalence and dominance of packing patterns. These investigations can be used for essentially improving exact algorithms.

The main topic of this paper consists of a comprehensive investigation of equivalence and dominance of packing patterns which will be done in Sections 5 and 6. Using suitable kinds of equivalence and dominance, the goal is to develop an efficient branch&bound algorithm with which problems of practical interest may be solved exactly (Section 7). The fundamental principle of the algorithm is similar to the so-called *contour concept* proposed in [10] and [9]. (A somewhat different description of this concept can be found in [8].) In Section 2 several definitions and denotations will be introduced. Fundamental principles to solve the packing problem will be briefly discussed in Sections 3 and 4 in order to motivate the investigations on equivalence and dominance of packing patterns.

Work, dealing with this packing problem, is done e.g. in [1], [2], [3], [7], [13] and [14]. BEASLEY's exact algorithm [2] is based on a 0/1-model which cannot be used in practice because of the rapid increase of the number of 0/1-variables and restrictions.

The algorithm proposed in [1] does not cover all packing patterns since only so-called *5-block-patterns* and guillotine patterns are considered. Finally, only guillotine-packing patterns are considered in [3], [7], [13] and [14].

In order to reduce the number of packing patterns which has to be considered, the so-called *normalized* (bottom-left justified) packing patterns are already used in [7] and [3]. (Normalization is one kind of equivalence.) In [7] and [3] sets of the so-called *raster points* which were introduced in [12] are also used but not the reduced sets of raster points (cf. [12]).

The considerations of symmetry made in [7] are restricted only on the placement of guillotine cuts. A further (restricted) usage of symmetry to reduce computational effort can be observed in [1] where only one of the four symmetric packing patterns (resulting from a reflection on the axises) is considered.

In order to investigate equivalence and dominance of packing patterns and to describe the algorithm we use the following denotations:

L	the length of the pallet,
W	the width of the pallet,
m	the number of (different) types of pieces,
T	$= \{1, \ldots, m\}$ the set of indices of piece types,
l_t	the length of a piece T_t of type t, $t \in T$,
w_t	the width of T_t, $t \in T$,
b_t	the supply of T_t, $t \in T$.

The objective is to find a subset of the pieces which can be packed in a feasible manner within the pallet and which minimizes the unused area. A set of pieces is said to be feasibly packed if the pieces are orthogonally allocated, if they do not overlap each other and if they fit completely within the pallet.

For the sake of simplicity, we consider only the case of waste minimization. The more complex case, such as assigning values to the pieces and maximizing the total value of the packing pattern, is left for future research.

Some or all of the piece types may be rotated by 90 degrees. Let T_t, $t = m+1$, $\ldots, \overline{m}$ $(\overline{m} \geq m)$, denote the rotated pieces. Let $\overline{T} = \{1, \ldots, \overline{m}\}$. $\tau_t \in T$ be the index of the unrotated piece type for T_t, $t \in \overline{T}$. Without loss of generality we assume that all Input-data are positive integers.

2 Definitions and Notations

The **packing** of a piece of type t, $t \in \overline{T}$, can be uniquely described by t and the so-called **allocation point** (x, y) which gives the coordinates of the lower-left corner. For short we will denote in the following such a packing with (t, x, y). The packing (t, x, y) covers the rectangular region

$$R(l_t, w_t, x, y) := \{(r, s) \in \mathcal{R}^2 \ : \ x \leq r \leq x + l_t, \ y \leq s \leq y + w_t\}.$$

A packing pattern of k pieces can be defined as follows: Let $I := \{1, \ldots, k\}$. Then the set

$$A = \{(t_i, x_i, y_i) : i \in I\}$$

is called a **packing pattern** with k packings (with k packed pieces).

The fundamental principle of the algorithm considered here to solve the packing problem consists in a successive allocation of pieces, that is, given a packing pattern we have to look for an unused area and an available piece which fits completely within the free space.

For a more detailed description of the contour concept at first we introduce the so-called **raster sets** which reduce essentially the number of potential allocation points. For $l = (l_1, \ldots, l_{\overline{m}})^T \in Z_+^{\overline{m}}$ and $L \in Z_+$ the set

$$S(l, L) := \{r : r = \sum_{i=1}^{\overline{m}} l_i a_i, \ r \leq L, a_i \in Z_+, \ i = 1, \ldots, \overline{m}\}$$

denotes the set of **raster points** in L-direction. Using

$$\langle s\rangle := \max\{r \in S(l,L) : r \le s\}$$

the reduced set of raster points in L-direction is as follows (cf. [12]):

$$\widetilde{S}(l,L) := \{\langle L - r\rangle : r \in S(l,L)\}.$$

Let $\widetilde{S}(w,W)$ be the corresponding set in W-direction. Furthermore, let

$$\langle l\rangle_L := \max\{r \in \widetilde{S}(l,L) \ : \ r \le l\}, \quad \langle w\rangle_W := \max\{s \in \widetilde{S}(w,W) \ : \ s \le w\}.$$

For any packing (t,x,y) with allocation point (x,y) a generalized "upper right corner" $(\overline{x},\overline{y})$ can be determined by $\widetilde{S}(l,L)$ and $\widetilde{S}(w,W)$ as follows:

$$\overline{x} := L - \langle L - x - l_t\rangle_L, \quad \overline{y} := W - \langle W - y - w_t\rangle_W. \tag{1}$$

Then the area used by the packing (t,x,y) is the following:

$$\{(r,s) \in \mathcal{R}^2 \ : \ x \le r \le \overline{x}, \ y \le s \le \overline{y}\}.$$

For shortness we use the following identifications:

$$(t,x,y) := \{(r,s) \in \mathcal{R}^2 \ : \ x \le r \le \overline{x}, \ y \le s \le \overline{y}\}$$

and

$$A = \{(t_i,x_i,y_i) : i \in I\} = \{(r,s) \in \mathcal{R}^2 \ : \ \exists i \in I \text{ with } (r,s) \in (t_i,x_i,y_i)\}.$$

With respect to the contour concept, the set

$$U(A) = \{(x,y) \in \mathcal{R}^2_+ \ : \ \exists i \in I \text{ with } x \le \overline{x}_i, \ y \le \overline{y}_i\}$$

is called the **covered region of the pattern** A. It is obvious that $A \subset U(A)$ holds. In the contour concept only the area of the pallet not occupied by $U(A)$ is considered for further packings. Hence, the set $U(A) \setminus A$ may be considered as waste.

The polygonal set

$$K(A) := \mathrm{cl}\,(U(A) \cap \{(x,y) : xy > 0\} \setminus \mathrm{int}U(A))$$
$$\bigcup \ (\{(x,y) : xy = 0, \ 0 \le x \le L, \ 0 \le y \le W\} \setminus U(A))$$

is said to be the **contour** of packing pattern A (cf. Fig. 1).

Reversely, given a contour K a covered region U is defined as follows:

$$U = U(K) := \{(x,y) \in \mathcal{R}^2_+ : \exists (r,s) \in K \text{ with } (x,y) \le (r,s)\}.$$

This area will also be called a **contour area**.

A contour K_1 is said to be **above** another contour K_2 if $U(K_2) \subset U(K_1)$ is valid.

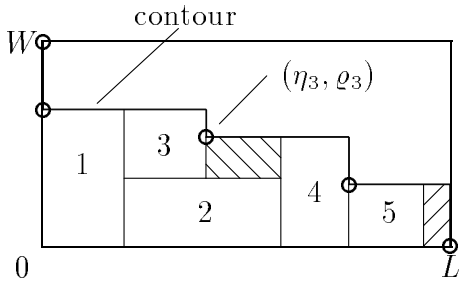

Figure 1: Contour concept

The contour $K(A)$ can be represented by a sequence

$$\{(\eta_i, \varrho_i) : i = 0, \ldots, n+1\}$$

of $n+2$ points, $(n = n(A))$, characterized by

a) $(\eta_0, \varrho_0) = (0, W),\ (\eta_{n+1}, \varrho_{n+1}) = (L, 0)$,

b) $\eta_0 \leq \eta_1 < \ldots < \eta_n \leq \eta_{n+1},\ \varrho_0 \geq \varrho_1 > \ldots > \varrho_n \geq \varrho_{n+1}$,

c) $(\eta_0, \varrho_0) \neq (\eta_1, \varrho_1),\ (\eta_n, \varrho_n) \neq (\eta_{n+1}, \varrho_{n+1})$.

It is easy to see that for a given packing pattern A the set of potential allocation points can be reduced to the points (η_i, ϱ_i), $i = 1, \ldots, n$, because of a simple dominance criterion.

A packing pattern $A = \{(t_i, x_i, y_i) : i = 1, \ldots, k\}$ is called **monotone** if

$$(t_i, x_i, y_i) \cap \mathrm{int}U(A(\{1, \ldots, i-1\})) \ = \ \emptyset, \quad i = 2, \ldots, k,$$

holds. Obviously, only monotone packing patterns will be constructed by the contour concept. A monotone packing pattern $A = \{(t_i, x_i, y_i) : i = 1, \ldots, k\}$ is called **normalized** if integers $r_0 > 0$ and $s_0 > 0$ exist with

a) $x_1 = 0,\ y_1 = 0$,

b) $(x_i + r, y_i) \in K(A(\{1, \ldots, i-1\}))$ for $0 \leq r \leq r_0$, $i = 2, \ldots, k$,

c) $(x_i, y_i + s) \in K(A(\{1, \ldots, i-1\}))$ for $0 \leq s \leq s_0$, $i = 2, \ldots, k$,

that is, any piece is placed bottom-left justified (it is shifted as far as possible in bottom-left direction).

In the following we consider in general only normalized packing patterns (excepting Section 5.5). The investigations with respect to equivalence and dominance of packing patterns will be made in view of the allocation algorithm which is based on the contour concept.

To a packing pattern $A = \{(t_i, x_i, y_i) : i \in I\}$ a **packing vector**

$$\lambda(A) = (\lambda_1, \ldots, \lambda_m)^T \in Z_+^m$$

is assigned with

$$\lambda_t := \mathrm{card}\{i \in I : \tau_{t_i} = t\}, \quad t \in T,$$

where λ_t gives the number of pieces of type t contained within the packing pattern A.

Given a subset $\tilde{I}$ of I, a **sub-pattern**

$$A(\tilde{I}) = \{(t_i, x_i, y_i) : i \in \tilde{I}\}$$

of a packing pattern A is defined.

3 Dynamic Programming Approach

The definition of suitable *states* is very important for the construction of algorithms which are based on dynamic programming.

The packing patterns obtained by a successive allocation of packings lead to states which can be characterized functionally as follows:

$$(\lambda(A), K(A)).$$

For this definition of states the sequence of packings does not have any importance.

Since in general several contours belong to one packing vector λ (i.e. different packing patterns with same packing vector) it is absolutely necessary for constructing efficient algorithms to exclude such packing patterns with help of dominance considerations which cannot lead to optimal solutions. Hence, among the contours having the same packing vector only these packing patterns have to be considered furthermore whose contour is minimal. (A contour $K_1(\lambda)$ is called minimal if no other contour $K_2(\lambda)$ exists to the packing vector λ which is below $K_1(\lambda)$).

Additionally, the set of states which have to be considered can be reduced using symmetry and equivalence considerations of contours.

The number of packed pieces (number of packings) can be used as *stage (index)* in the sense of a stage in an N-stage decision process. The allocation of one more piece represents the transformation from one stage to the next.

Since the number of (different) states increases rapidly with the number of piece types m, the applicability of dynamic programming is strongly restricted to the case of very small m or to instances where the number of pieces which can simultaneously placed on the pallet is very small.

If, in difference to the general case, it is assumed that the desired packing pattern has to be of guillotine-type, then corresponding algorithms which are based on dynamic programming work very efficiently ([6], [12]).

4 Branch & Bound Approach

In difference to dynamic programming where difficulties due to memory restrictions occur, instances with a huge number of feasible solutions (states) often can be solved using the branch&bound principle.

For solving the considered packing problem with a b&b algorithm, it is desirable to use the states introduced in the previous section in order to guarantee that the subproblems

will be investigated at least once. However, it is not known so far how to generate such states systematically within a LIFO strategy.

For that reason we use another approach. One possibility to systematically generate subproblems for a LIFO strategy consists of applying the contour concept (cf. [9]). In this case, however, the packing patterns are used to define subproblems (states) in difference to the states introduced in the previous section. Hence, subproblems will be generated several times also in that case when only normalized packing patterns are considered. In order to exclude a multiple treatment of subproblems, investigations with respect to equivalence and symmetry are necessary. Furthermore, possibilities to exclude subproblems have to be used which are based on dominance considerations.

In the following, we introduce several terms of equivalence of packing patterns:
- equivalence,
- p-equivalence (permutation equivalence),
- vb-, hb-, b-equivalence (vertical, horizontal, block equivalence),
- bs-, cs-, s-equivalence (block symmetry, contour symmetry, symmetry),
- vt-, ht-, t-equivalence (vertical, horizontal, translation equivalence),

and several terms of dominance:
- c-dominance (contour dominance),
- p-dominance (piece dominance),
- vb-, hb-, b-dominance (vertical, horizontal, block dominance),
- f-dominance (fit-in dominance),
- vt-, ht-, t-dominance (vertical, horizontal, translation dominance).

In the following we agree that the sequence of branchings (within the underlying b&b algorithm) corresponds to the sequence of piece types. Therefore, subproblems having the same number of packed pieces (subproblems of equal branching depth) are generated in lexicographic decreasing order with respect to the corresponding packing vectors.

By this we assign simultaneously a sequence of packings to a packing pattern $A = \{(t_i, x_i, y_i) : i = 1, \ldots, k\}$ corresponding to the index i.

5 Equivalence of Packing Patterns

5.1 Equivalence

Definition 1 *The packing patterns* $A = \{(t_i, x_i, y_i) : i = 1, \ldots, k\}$ *and* $\tilde{A} = \{(\tilde{t}_i, \tilde{x}_i, \tilde{y}_i) : i = 1, \ldots, k\}$ *are called* **equivalent** *if*

$$(\lambda(A), K(A)) \;=\; (\lambda(\tilde{A}), K(\tilde{A})).$$

The practical application of this equivalence, which is easy to verify, requires the knowledge of all subproblems that were investigated before. Therefore, all of the generated subproblems have to be stored in the form $(\lambda(A), K(A))$ which cannot be realized in practice.

Since this definition of equivalence of packing patterns cannot be used practically, we will consider other definitions of equivalence which on the one hand lead to smaller classes

of equivalent packing patterns. But on the other hand, with these terms of equivalence and dominance it may be possible to decide whether the current subproblem has already been investigated before or not without any storage requirements on the previously generated subproblems (only the branching sequence will be used).

5.2 p-(Permutation-)Equivalence of Packing Patterns

One possibility of defining another equivalence relation consists in considering the so-called p-equivalence. If two packings of a packing pattern can be exchanged with respect to the allocation sequence without violating the philosophy of the contour concept (both allocation sequences yield monotone packing patterns), then the resulting packing patterns are p-equivalent (Fig. 2).

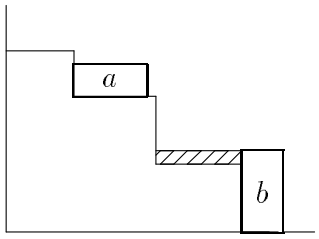

Figure 2: Packings which can be exchanged in their sequence

Definition 2 *The packing patterns* $A = \{(t_i, x_i, y_i) : i = 1, \ldots, k\}$ *and* $\widetilde{A} = \{(\widetilde{t}_i, \widetilde{x}_i, \widetilde{y}_i) : i = 1, \ldots, k\}$ *are called* **p-equivalent** *if a permutation* $\pi = (\pi_1, \ldots, \pi_k)$ *of* $I = \{1, \ldots, k\}$ *exists with* $(t_i, x_i, y_i) = (\widetilde{t}_{\pi_i}, \widetilde{x}_{\pi_i}, \widetilde{y}_{\pi_i})$ *for* $i \in I$.

In order to develop an efficient algorithm it is necessary to consider only one element of a class of p-equivalent packing patterns. In the following we use the **lexicographically smallest normalized (lsn-)** packing pattern as the representative of a class of p-equivalent packing patterns with respect to the natural lexicographic order relation of $\mathcal{R}^{3k}$.

For a (normalized) packing pattern $A = \{(t_i, x_i, y_i) : i = 1, \ldots, k\}$, a p-equivalent lexicographically smaller (with respect to $\mathcal{R}^{\ni\|}$) packing pattern $\widetilde{A} = \{(\widetilde{t}_i, \widetilde{x}_i, \widetilde{y}_i) : i = 1, \ldots, k\}$ can exist only in that case when indices r and s, $0 < r < s \leq k$, exist with

a) $t_r > t_s$, or

b) $t_r = t_s,\ x_r > x_s$, or

c) $t_r = t_s,\ x_r = x_s$ and $y_r > y_s$.

Case c) does not occur if only normalized packing patterns are considered since such a packing pattern is not monotone.

Let $\mu_1(A) := 0$ and

$$\mu_i(A) = \mu_i := \max\{t_j \ : \ j = 1, \ldots, i-1\}, \quad i = 2, \ldots, k.$$

Considering the contour concept after the packing of a new (the kth) piece one has to check whether a lsn-packing pattern (with respect to p-equivalence) is obtained. (We assume that the packing pattern $A(\{1,\ldots,k-1\})$ with $k-1$ pieces is an lsn-packing pattern.)

Hence, a lexicographically smaller p-equivalent packing pattern can exist only if $t_k < \mu_k$ or if $t_k = \mu_k$ and $\exists r < k$ with $t_r = t_k,\ x_r > x_k$ is fulfilled. Therefore, one has to check whether the exchange of the packings (t_k, x_k, y_k) and (t_r, x_r, y_r) leads to a normalized, especially monotone packing pattern.

In order to verify such a situation, the so-called **regions V_r of influence** of the kth piece with respect to packing pattern A and the rth packing ($r := k-1,\ldots,1$) are defined as follows in an algorithmic way (remember the definition of $\overline{x}_k$ in (1)):

0. Set $\overline{x} := \overline{x}_k,\ \overline{y} := \overline{y}_k$.

1. $V_k := \{(x,y)\ :\ x \le \overline{x},\ y \le \overline{y}\}$.

2. For $r := k-1,\ldots,1$:

   a) if $(x_r, y_r) \in \mathrm{int}V_{r+1}$ then:
   $\overline{x} := \max\{\overline{x}, \overline{x}_r\},\ \overline{y} := \max\{\overline{y}, \overline{y}_r\}$,
   $V_r := \{(x,y)\ :\ x \le \overline{x},\ y \le \overline{y}\}$,

   b) if $(x_r, y_r) \notin \mathrm{int}V_{r+1}$ then $V_r := V_{r+1}$.

Assertion 1 *Let $A = A(I) = \{(t_i, x_i, y_i) : i = 1,\ldots,k\}$ be a normalized packing pattern. We assume that $A(I \setminus k) = \{(t_i, x_i, y_i) : i = 1,\ldots,k-1\}$ is an lsn-packing pattern. Then it holds: A represents an lsn-packing pattern if and only if*

$$\nexists\ r \in I \text{ with } r < k,\ t_r > t_k,\ (x_r, y_r) \notin int V_{r+1}$$

and

$$\nexists\ r \in I \text{ with } r < k,\ t_r = t_k,\ x_r > x_k,\ (x_r, y_r) \notin int V_{r+1}.$$

Hence, using this assertion one can decide whether the subproblem $(\lambda(A), K(A))$ is generated the first time (A is an lsn-packing pattern) or not.

5.3 b-(Block-)Equivalence of Packing Patterns

Definition 3 *A sub-pattern $A(\widetilde{I}) = \{(t_i, x_i, y_i) : i \in \widetilde{I}\}$ of A is called* **block-wise** *or* **block pattern** *if there exists a rectangle B with*
a) $(t_i, x_i, y_i) \subset B \quad \forall\ i \in \widetilde{I}$,
b) $int(t_i, x_i, y_i) \cap B = \emptyset \quad \forall\ i \notin \widetilde{I}$.

The smallest such rectangle $B(A(\widetilde{I}))$, if existing, is the following:

$$B(A(\widetilde{I})) := \{(x,y)\ :\ \min_{i\in\widetilde{I}} x_i \le x \le \max_{i\in\widetilde{I}} \overline{x}_i,\ \min_{i\in\widetilde{I}} y_i \le y \le \max_{i\in\widetilde{I}} \overline{y}_i\}.$$

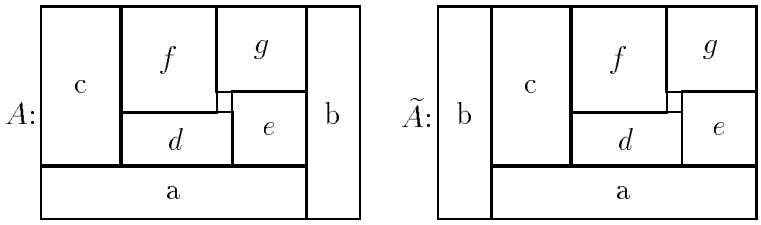

Figure 3: b-equivalent packing patterns

The block length $L(A(\widetilde{I}))$ and the block width $W(A(\widetilde{I}))$ are defined as follows:

$$L(A(\widetilde{I})) := \max_{i\in\widetilde{I}} \overline{x}_i - \min_{i\in\widetilde{I}} x_i, \qquad W(A(\widetilde{I})) := \max_{i\in\widetilde{I}} \overline{y}_i - \min_{i\in\widetilde{I}} y_i.$$

A block pattern $A(\widetilde{I})$which contains more than one packing is called **proper** block pattern. A proper block pattern is said to be **horizontal** if there does not exist any proper subset $\widehat{I}$ of $\widetilde{I}$ (that is $\widehat{I} \subset \widetilde{I}$, $\widehat{I} \neq \emptyset$, $\widehat{I} \neq \widetilde{I}$) which forms by itself a block pattern with equal block width. Analogously, a proper block pattern is said to be **vertical** if there does not exist any proper subset of $\widetilde{I}$ which is by itself a block pattern with equal block length. A block pattern is called a **minimal** block pattern if it is simultaneously a horizontal and a vertical block pattern.

The sub-pattern in Fig. 3 which is formed by the packings $\{a, c, d, e, f, g\}$ is a horizontal block pattern. The packings $\{c, d, e, f, g\}$ define a vertical block pattern. Proper block patterns are also formed by $\{a, b, c, d, e, f, g\}$ and $\{d, e, f, g\}$. The latter is also a minimal block pattern.

Definition 4 *Given equivalent packing patterns* $A = \{(t_i, x_i, y_i) : i = 1, \ldots, k\}$ *and* $\widetilde{A} = \{(\widetilde{t}_i, \widetilde{x}_i, \widetilde{y}_i) : i = 1, \ldots, k\}$. *We suppose there exist disjunctive block patterns* $A(I_1) = \{(t_i, x_i, y_i) : i \in I_1\}$ *and* $A(I_2) = \{(t_i, x_i, y_i) : i \in I_2\}$ *of* A
and disjunctive block patterns
$\widetilde{A}(\widetilde{I}_1) = \{(\widetilde{t}_i, \widetilde{x}_i, \widetilde{y}_i) : i \in \widetilde{I}_1\}$ *and* $\widetilde{A}(\widetilde{I}_2) = \{(\widetilde{t}_i, \widetilde{x}_i, \widetilde{y}_i) : i \in \widetilde{I}_2\}$ *of* $\widetilde{A}$
which forms exchangeable blocks
$B_1 := B(A(I_1)) = R(a, b, x_1, y_1)$ *and* $B_2 := B(A(I_2)) = R(c, d, x_2, y_2)$ *and*
$\widetilde{B}_1 := B(\widetilde{A}(\widetilde{I}_1)) = R(\widetilde{a}, \widetilde{b}, \widetilde{x}_1, \widetilde{y}_1)$ *and* $\widetilde{B}_2 := B(\widetilde{A}(\widetilde{I}_2)) = R(\widetilde{c}, \widetilde{d}, \widetilde{x}_2, \widetilde{y}_2)$
i.e. for which hold:
a) $a = c,\ x_1 = x_2,\ y_1 = y_2 + d,$
b) $\widetilde{a} = \widetilde{c},\ \widetilde{x}_1 = \widetilde{x}_2,\ \widetilde{y}_1 = \widetilde{y}_2 + \widetilde{d},$
c) $a = \widetilde{a},\ b = \widetilde{d},\ d = \widetilde{b}.$
Then A *and* $\widetilde{A}$ *are called* **vb-equivalent** *(vertical block equivalent).*

Analogously, **hb-equivalence** of packing patterns can be defined (see Fig. 4).

Definition 5 *The packing patterns* A *and* $\widetilde{A}$ *are called* **b-equivalent** *(block equivalent) if both are vb- or hb-equivalent.*

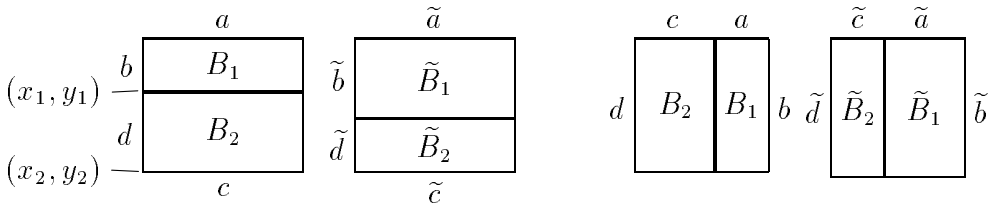

Figure 4: vb-equivalent and hb-equivalent packing patterns

Fig. 4 shows also a b-equivalent packing pattern.

Again, in order to obtain efficient algorithms only one representative of a class of b-equivalent packing patterns should be inspected. This can be reached e.g. if it is demanded that block B_2 contains a lexicographically non-greater packing vector in comparison with block B_1, i.e.[1].

$$\lambda(A(I_1)) \geq_L \lambda(A(I_2)).$$

Hence, in the b&b algorithm when a packing pattern A is generated having two neighboured blocks B_1 and B_2 with block patterns $A(I_1)$ and $A(I_2)$, one can easily decide whether the subproblem $(\lambda(A), K(A))$ has already been investigated ($\lambda(A(I_1)) \not\geq_L \lambda(A(I_2))$) or not.

An efficient realization of such a block-equivalence test is a still open problem. It should be noted that it is not sufficient to consider only minimal blocks as Fig. 3 shows.

5.4 s-Equivalence / Symmetries

In order to develop efficient packing algorithms it is necessary to recognize any form of symmetry of packing patterns to keep down the computational amount.

Symmetries occur especially within block patterns. In the sequel we assume, for the sake of simplicity, that the packing pattern A represents a block pattern with more than one piece, which does not contain any proper block pattern containing (t_k, x_k, y_k).

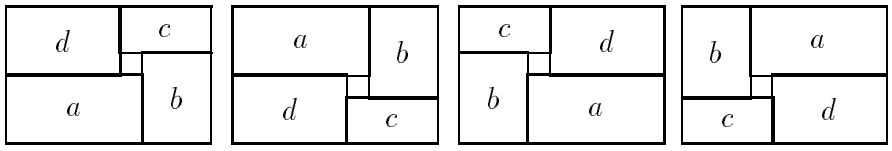

Figure 5: Symmetric (non-quadratic) block patterns with 4 pieces

If a block pattern A results when a piece is allocated, then it should be checked whether any symmetric packing pattern of A has already been considered. The following tests may be used as a criterion in the case of block patterns with 4 packed pieces to identify a

[1] $\geq_L$ denotes the common lexicographic order relation.

representative of symmetric packing patterns (cf. Fig. 5; the letters a - d represent the type of the allocated piece):

1. $a < b$ for symmetry with respect to a vertical line,

2. $a < d$ for symmetry with respect to a horizontal line,

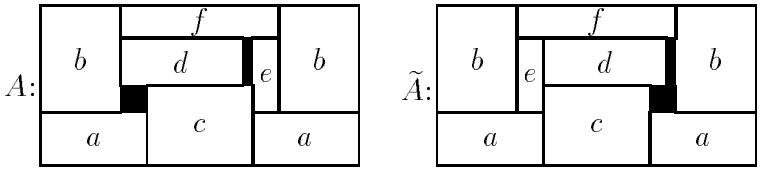

Figure 6: Symmetric block patterns

In the general case, symmetry of packing patterns can be handled by considering the sequence of packings within the block pattern. In the example in Fig. 6 where we assume $a < b < \ldots$), a, b, c, a, d, e, f, b is the lexicographically smallest sequence of the types of pieces belonging to A. Considering the block pattern $\tilde{A}$ which is a reflection of A with respect to a vertical line, then a, b, e, c, a, d, f, b is the corresponding sequence. Because of $c < e$, when the packing pattern $\tilde{A}$ is generated in the algorithm, the block pattern A has already been obtained and investigated.

The following definition gives a generalized form of symmetry of packing patterns.

Definition 6 *Given equivalent packing patterns* $A = \{(t_i, x_i, y_i) : i = 1, \ldots, k\}$ *and* $\tilde{A} = \{(\tilde{t}_i, \tilde{x}_i, \tilde{y}_i) : i = 1, \ldots, k\}$.
We assume there exist proper block patterns
$A(I_1) = \{(t_i, x_i, y_i) : i \in I_1\}$ *of* A *and* $\tilde{A}(\tilde{I}_1) = \{(\tilde{t}_i, \tilde{x}_i, \tilde{y}_i) : i \in \tilde{I}_1\}$ *of* $\tilde{A}$
whose blocks $B_1 := B(A(I_1)) = R(a, b, x_1, y_1)$ *and* $\tilde{B}_1 := B(\tilde{A}(\tilde{I}_1)) = R(\tilde{a}, \tilde{b}, \tilde{x}_1, \tilde{y}_1)$
coincide (same surrounding rectangle) and in addition have the same packing vector, i.e. for which hold:

a) $a = \tilde{a},\ b = \tilde{b},\ x_1 = \tilde{x}_1,\ y_1 = \tilde{y}_1,$

b) $\lambda(A(I_1)) = \lambda(\tilde{A}(\tilde{I}_1))$.

Then A *and* $\tilde{A}$ *are called* **bs-equivalent** *(block symmetric equivalent).*

Again, only one representative of a class of bs-equivalent packing patterns have to be investigated. In order to test whether after the allocation of the kth piece a packing pattern results for which a bs-equivalent packing pattern has already been considered, all block patterns containing (t_k, x_k, y_k) have to be inspected with respect to symmetry. Furthermore, reflections (horizontal, vertical and central) also have to be considered.

In the case where a quadratic block pattern is present and all packed pieces may be rotated by 90^o, then two further possibilities of symmetry have to be taken into account (s. Fig. 7).

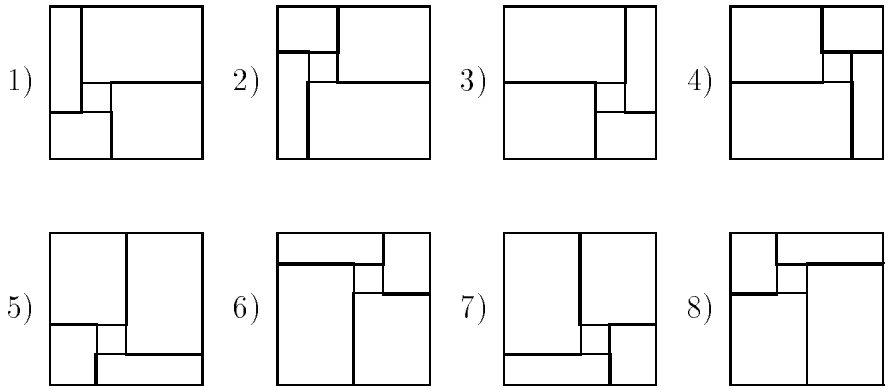

Figure 7: Symmetric quadratic block patterns: reflection on horizontal, vertical, horizontal and vertical axis, on diagonals, rotation by 90^o, 270^o.

A further form of symmetry (not restricted to block patterns) arises from the symmetry of the corresponding contour (cf. Fig. 8).

Definition 7 *A packing pattern A is called* **c-symmetric** *if all allocated pieces may be rotated by 90^o and if there exists an $r \in \{-1, 0, 1\}$ for the corresponding contour $(0, W)$, $(\eta_1, \varrho_1), \ldots, (\eta_n, \varrho_n), (L, 0)$ such that hold:*

$$\eta_{i+r} = \varrho_{n+1-i}, \quad \varrho_{i+r} = \eta_{n+1-i}, \qquad i = 1, \ldots, n.$$

The c-symmetry of packing patterns may be considered in a more general form if the c-symmetry for a block pattern is used (cf. Fig. 8).

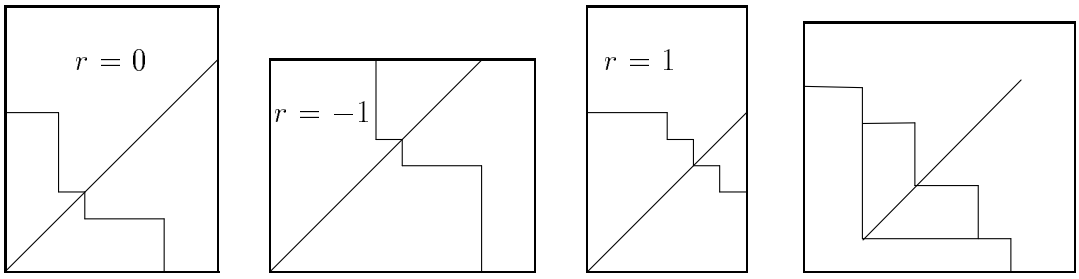

Figure 8: c-symmetric packing patterns

Packing patterns which are c-symmetric and coincide by reflection are called **cs-equivalent** (contour symmetric). Two packing patterns which are bs- or cs-equivalent are called **s-equivalent (symmetric)**.

5.5 t-(Translation-)Equivalence

For the investigations in this section we consider not only normalized packing patterns.

Definition 8 *Given equivalent packing patterns* $A = \{(t_i, x_i, y_i) : i = 1, \ldots, k\}$ *and* $\widetilde{A} = \{(\widetilde{t}_i, \widetilde{x}_i, \widetilde{y}_i) : i = 1, \ldots, k\}$ *for which* $t_i = \widetilde{t}_i$ *for* $i \in I = \{1, \ldots, k\}$ *is valid.*
a) A and $\widetilde{A}$ *are called* **ht-equivalent** *(horizontal translation equivalent) if there exists an index* $i_0 \in I$ *such that* $x_i = \widetilde{x}_i,\ i \in I \setminus i_0$, *and* $y_i = \widetilde{y}_i,\ i \in I$.
b) A and $\widetilde{A}$ *are called* **vt-equivalent** *(vertical translation equivalent) if there exists an index* $i_0 \in I$ *such that* $x_i = \widetilde{x}_i,\ i \in I$, *and* $y_i = \widetilde{y}_i,\ i \in I \setminus i_0$.
c) A and $\widetilde{A}$ *are called* **t-equivalent** *(translation equivalent) if there exists a sequence* $A = A_1, \ldots, A_p = \widetilde{A}$ *of equivalent packing patterns such that* A_j *and* A_{j+1} *are ht- or vt-equivalent for* $j = 1, \ldots, p-1$.

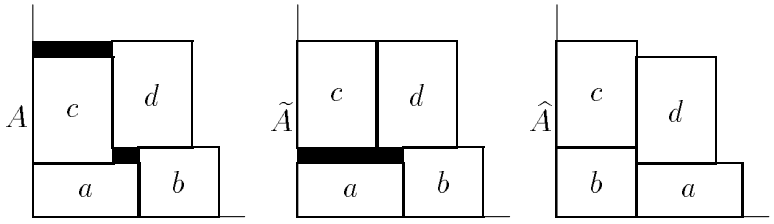

Figure 9: t-equivalent packing patterns

The packing pattern A in Fig. 9 is normalized. The packing pattern $\widetilde{A}$ which is t-equivalent to A however is not normalized. As will be shown in Section 6.2 the packing pattern $\widetilde{A}$ will be dominated by another packing pattern $\widehat{A}$. Hence, both packing patterns A and $\widetilde{A}$ need not to be considered in a b&b algorithm.

A well-known result (used already in [7]) is the following:

Assertion 2 *Let A be a monotone packing pattern. Then there exists a packing pattern* $\widetilde{A}$ *which is t-equivalent to A and is normalized.*

The packing pattern $\widetilde{A}$ may be obtained by a step-wise translation of packings in "bottom-left" or "left-bottom" direction. Thus in general A is not unique.

The packing patterns $\widetilde{A}$ and $\overline{A}$ in Fig. 10 are t-equivalent. But, the sub-patterns of $\widetilde{A}$ and $\overline{A}$ which contain only the pieces a, b, c, d, are not equivalent.

6 Dominance

6.1 c-(Contour-), p-(Piece-) and Dominance of Packing Patterns

Dominance of packing patterns may be defined in a natural way as follows:

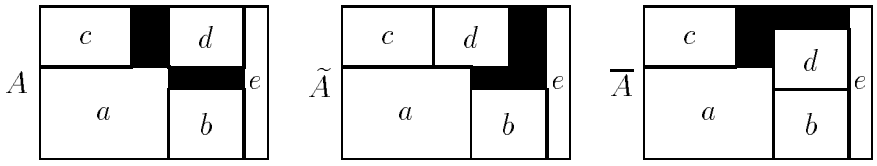

Figure 10: t-equivalent normalized packing patterns $\widetilde{A}$ and $\overline{A}$ of A

Definition 9 *Let* $A = \{(t_i, x_i, y_i) : i = 1, \ldots, k\}$ *and* $\widetilde{A} = \{(\widetilde{t}_i, \widetilde{x}_i, \widetilde{y}_i) : i = 1, \ldots, \widetilde{k}\}$.
a) If $\lambda(A) = \lambda(\widetilde{A})$ *and* $U(A) \subset U(\widetilde{A})$ *hold then* A *is said to be* **c-dominant** *(contour dominant) compared with* $\widetilde{A}$.
b) If $\lambda(A) \geq \lambda(\widetilde{A})$ *and* $U(A) = U(\widetilde{A})$ *hold then* A *is said to be* **p-dominant** *(piece dominant) compared with* $\widetilde{A}$.
c) The packing pattern A *is called* **dominant** *compared with the packing pattern* $\widetilde{A}$*, or* A **dominates** $\widetilde{A}$*, if* A *is c-dominant or p-dominant to* $\widetilde{A}$.

Since the c-dominant packing pattern A occupies fewer area in comparison with a c-dominated packing pattern $\widetilde{A}$, it is possible to generate new packing patterns by allocating further packings to A at least as good as it is possible when $\widetilde{A}$ is used.

Since p-dominant packing patterns occupy the same contour area it is not possible to generate a better packing pattern from a p-dominated packing pattern in comparison with a p-dominant packing pattern. Hence, it is sufficient to investigate only dominant packing patterns in order to find an optimal one.

The application of this dominance relation within a b&b algorithm is difficult since in general the knowledge of all subproblems (packing patterns) already considered is required (at least for the c-dominance). This should not be realistic in practice.

In the following some more special dominance relations will be considered which possibly can be realized in a more easier way.

6.2 b-(Block-)Dominance

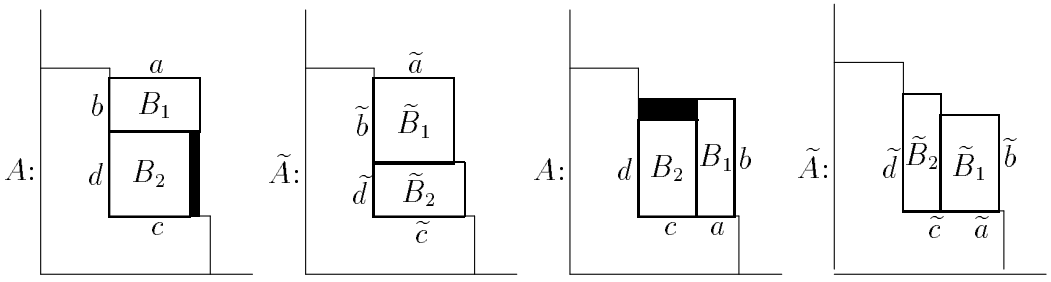

Figure 11: vb- and hb-dominant packing patterns

A useful tool results from the consideration of the so-called block-dominance (Fig. 11).

Definition 10 *Let*

$A = \{(t_i, x_i, y_i) : i = 1, \ldots, k\}$ *and* $\widetilde{A} = \{(\widetilde{t}_i, \widetilde{x}_i, \widetilde{y}_i) : i = 1, \ldots, k\}$

be packing patterns such that

$\lambda(A) = \lambda(\widetilde{A})$.

Furthermore let

$A(I_1) = \{(t_i, x_i, y_i) : i \in I_1\}$ *and* $A(I_2) = \{(t_i, x_i, y_i) : i \in I_2\}$

be disjunctive block patterns of A *and let*

$\widetilde{A}(\widetilde{I}_1) = \{(\widetilde{t}_i, \widetilde{x}_i, \widetilde{y}_i) : i \in \widetilde{I}_1\}$ *and* $\widetilde{A}(\widetilde{I}_2) = \{(\widetilde{t}_i, \widetilde{x}_i, \widetilde{y}_i) : i \in \widetilde{I}_2\}$ *of* $\widetilde{A}$

be disjunctive block patterns of $\widetilde{A}$ *with the blocks*

$B_1 := B(A(I_1)) = R(a, b, x_1, y_1)$, $B_2 := B(A(I_2)) = R(c, d, x_2, y_2)$, *and*

$\widetilde{B}_1 := B(\widetilde{A}(\widetilde{I}_1)) = R(\widetilde{a}, \widetilde{b}, \widetilde{x}_1, \widetilde{y}_1)$, $\widetilde{B}_2 := B(\widetilde{A}(\widetilde{I}_2)) = R(\widetilde{c}, \widetilde{d}, \widetilde{x}_2, \widetilde{y}_2)$.

Furthermore, suppose:

a) $a > c,\ x_1 = x_2,\ y_1 = y_2 + d$,

b) $\widetilde{a} < \widetilde{c},\ \widetilde{x}_1 = \widetilde{x}_2,\ \widetilde{y}_1 = \widetilde{y}_2 + \widetilde{d}$,

c) $a = \widetilde{c},\ c = \widetilde{a}$.

Then $\widetilde{A}$ *is called* **vb-dominant** *(vertical block dominant) compared with* A *(cf. Fig. 11).*

If $\widetilde{A}$ is vb-dominant to A then $\widetilde{A}$ dominates the packing pattern A.

In a similar way, the so-called **hb-dominance** (horizontal block dominance) of packing patterns can also be defined (cf. Fig. 11). If $\widetilde{A}$ is hb-dominant to A then $\widetilde{A}$ dominates the packing pattern A.

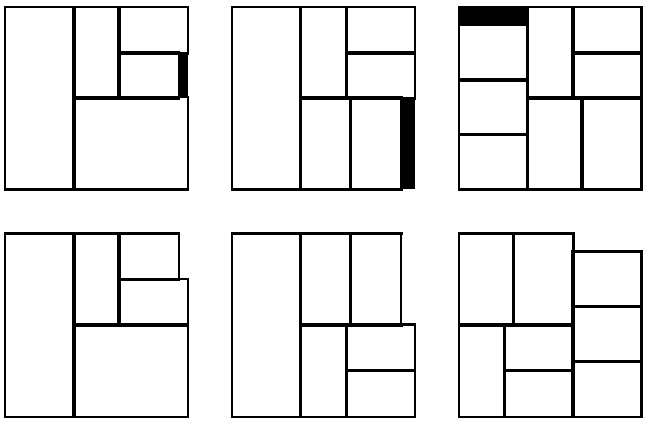

Figure 12: Examples of vb- and hb-dominant packing patterns

Definition 11 *A packing pattern* A *is said to be* **b-dominated** *(block dominated) by a packing pattern* $\overline{A}$ *if* A *is vb-dominated or hb-dominated by* $\overline{A}$.

An important particularity occurs in the case when the block pattern $A(I_1)$ contains only one packing, since then the computation of the corresponding block is not necessary.

As can be seen in the examples in Fig. 12, all combinations of neighboured blocks have to be investigated. The packing patterns shown on the top are dominated by the packing patterns below them.

The packing pattern $\widetilde{A}$ (Fig. 9) which is t-equivalent to A, is b-dominated by $\widehat{A}$. Hence, A is also dominated by $\widehat{A}$.

The following is then questioned: How can one recognize (in an efficient way) that there exists a t-equivalent packing pattern to a packing pattern A for which (e.g.) b-dominance can be proven?

6.3 f-(Fit in-)Dominance

A further, easy realizable tool is the so-called f-dominance (Fig. 13) which is based on the usage of areas of waste.

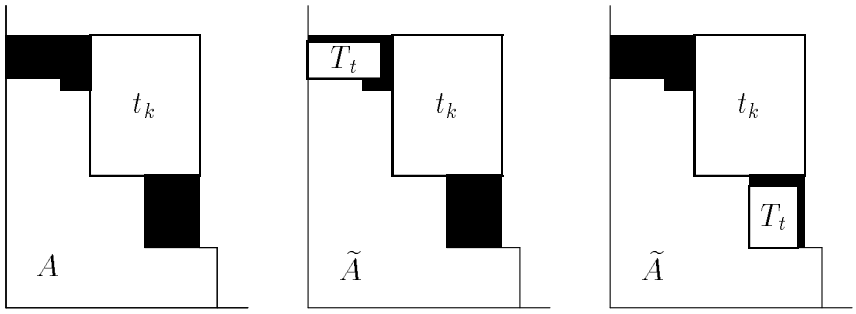

Figure 13: f-dominant packing patterns

Definition 12 *If the packing (t_k, x_k, y_k) produces a region of waste which contains a rectangle of length l and width w and if there exists a piece T_t with $l_t \leq l$ and $w_t \leq w$ (still available), then the packing pattern A is said to be* **f-dominated** *(fit in-dominated) by the packing pattern $\widetilde{A}$ which results from A by using the additional packing of piece T_t.*

Note, with respect to the contour concept it is sufficient to consider only these parts of waste which are determined by the packing of the actual (kth) piece.

If $\widetilde{A}$ is f-dominant to A, then $\widetilde{A}$ also is p-dominant to A. Therefore $\widetilde{A}$ is dominant to A. As the following example shows, equivalent packing patterns do not be t-equivalent in general. Furthermore, p-dominance does not imply f-dominance.

Example Let piece types be given with

$l_1 = 40,\ w_1 = 30,\ b_1 = 8,\ l_2 = 50,\ w_2 = 20,\ b_2 = 8,\ l_3 = 20,\ w_3 = 20,\ b_3 = 1,$

and further types with $100 < l_i \cdot w_i < 400$.

As Fig. 14 shows, the packing pattern A is p-dominant to $\widehat{A}$ and $\widetilde{A}$. But $\widehat{A}$ is not t-equivalent with $\widetilde{A}$. The packing pattern A is f-dominant to $\widehat{A}$ but A is not f-dominant compared with $\widetilde{A}$.

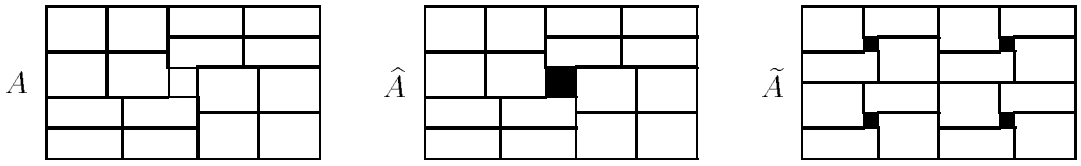

Figure 14: Example to the f-dominance

The f-dominance may be generalized in accordance with Fig. 15. A c-dominant packing pattern $\widetilde{A}$ may be obtained if a piece T_{i^*} (which is already packed) is placed within a region of waste of A and which determines the contour of A.

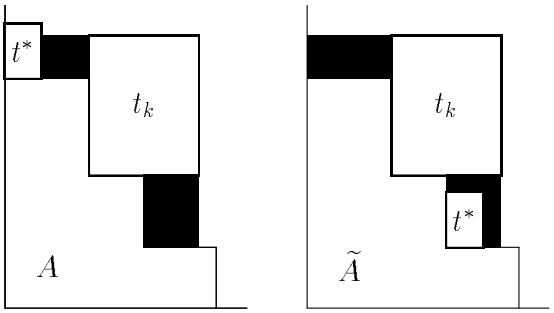

Figure 15: Generalization of f-dominance

6.4 t-(Translation-)Dominance

A further form of dominance is t-(translation-)dominance. As the Fig. 16 and 17 show, the translation of the packing k_2 induces a packing pattern with a smaller contour area. (The translation of piece k_2 meets an exchange within the allocation sequence.)

Definition 13 *Let*

$A = \{(t_i, x_i, y_i) : i = 1, \ldots, k\}$ *and* $\widetilde{A} = \{(\widetilde{t}_i, \widetilde{x}_i, \widetilde{y}_i) : i = 1, \ldots, k\}$
be packing patterns such that

$\lambda(A) = \lambda(\widetilde{A})$.

Furthermore, suppose there exist indices $k_1, k_2, \widetilde{k}_1, \widetilde{k}_2 \in \{1, \ldots, k\}$ *with* $k_1 < k_2$ *and* $\widetilde{k}_2 < \widetilde{k}_1$ *such that for the subpatterns of* A *and* $\widetilde{A}$*, defined by the index sets* $I = \{1, \ldots, k\} \backslash \{k_1, k_2\}$ *and* $\widetilde{I} = \{1, \ldots, k\} \backslash \{\widetilde{k}_1, \widetilde{k}_2\}$, $A(I) = \widetilde{A}(\widetilde{I})$ *hold. Furthermore, let*

$$(t_{k_1}, x_{k_1}, y_{k_1}) = (\widetilde{t}_{\widetilde{k}_1}, \widetilde{x}_{\widetilde{k}_1}, \widetilde{y}_{\widetilde{k}_1}) \quad \text{and } t_{k_2} = \widetilde{t}_{\widetilde{k}_2},\ x_{k_2} = \widetilde{x}_{\widetilde{k}_2},\ \widetilde{y}_{\widetilde{k}_2} < y_{k_2} = y_{k_1} + w_{t_{k_1}}.$$

Then the packing pattern A *is said to be* **vt-dominated** *(vertical translation dominated) by* $\widetilde{A}$.

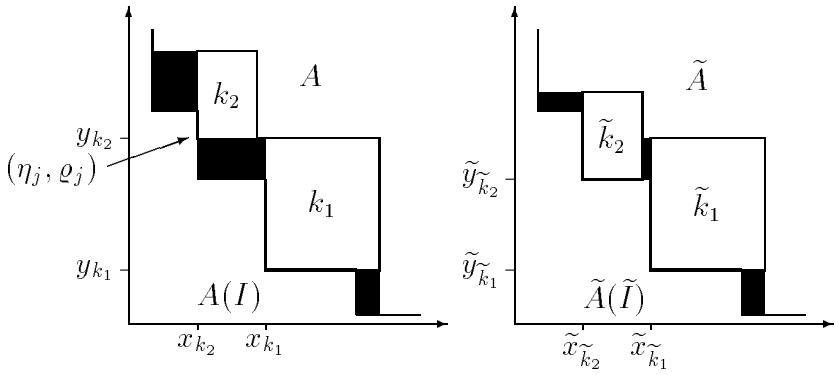

Figure 16: vt-dominance

The definition of vt-dominance makes use of the fact that translation may lead to a packing pattern with smaller contour area. Hence, if A is vt-dominated by $\widetilde{A}$ then $\widetilde{A}$ c-dominates A.

In a similar way, the so-called **ht-dominance** (horizontal translation dominance) of packing patterns may be defined (cf. Fig. 17).

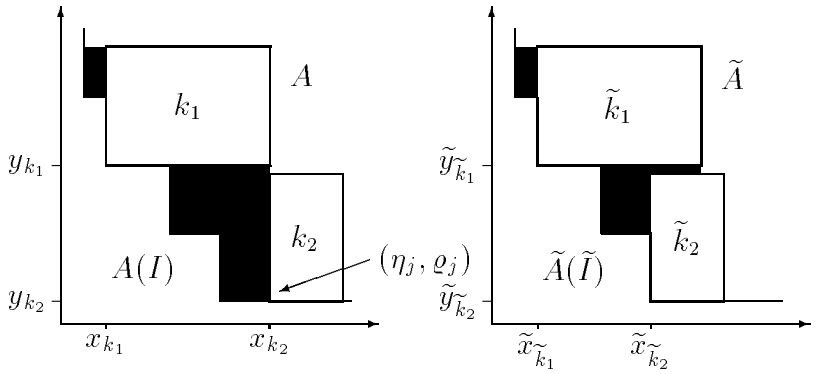

Figure 17: ht-dominance

Definition 14 *A packing pattern A is said to be* **t-dominated** *(translation-dominated) by a packing pattern $\widetilde{A}$ if A is vt- or ht-dominated by $\widetilde{A}$.*

It holds: if A is t-dominated by $\widetilde{A}$, then A is also c-dominated by $\widetilde{A}$. In the following the connections between f- and t-dominance will be investigated.

The t-dominance is neither necessary nor sufficient for f-dominance. To see this, the examples in Fig. 19 need to be considered. The packing (t_k, x_k, y_k) of a packing pattern A leads to waste which is sufficiently large enough to place a piece of type f in it. In this case, A will be f-dominated but not t-dominated. The t- and f-dominance may occur also simultaneously as can be seen in Fig. 19.

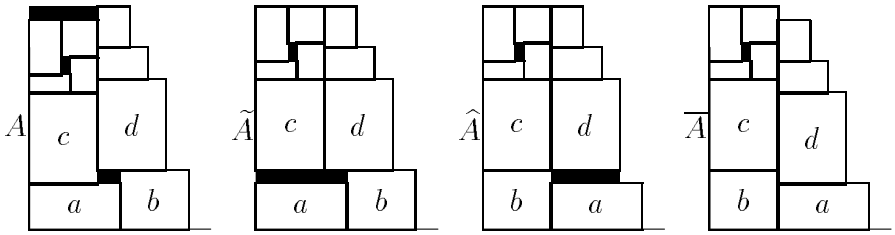

Figure 18: t-dominance of packing patterns

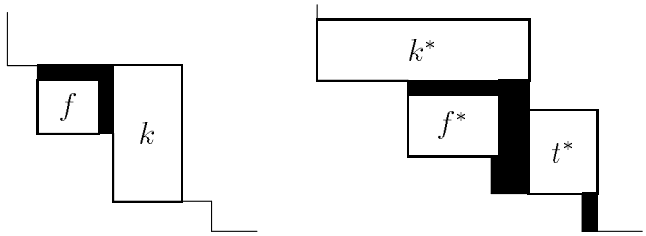

Figure 19: f- and t-dominance

The essential difference between f- and t-dominance consists in the following facts: f-dominant packing patterns are p-dominant and t-dominant packing patterns are c-dominant.

7 Branch & Bound Algorithm

The b&b algorithm given below computes an optimal packing pattern. Since the minimization of waste is the objective the potential usage of the remaining area, denoted with F, may be used to obtain an upper bound. Let

$$v(F) := \max\left\{\sum_{i=1}^{m} l_i w_i a_i \;:\; \sum_{i=1}^{m} l_i w_i a_i \le F,\ 0 \le a_i \le b_i,\ a_i \in Z_+,\ i = 1,\ldots,m\right\},$$

$$0 \le F \le LW.$$

For the description of the b&b algorithm the following denotations are used:

- k branching stage, number of packed pieces,
- j_k number of the allocation point used in stage k,
- t_k index of piece type packed in stage k,
- a_k number of allocation points in stage k.

Branch and Bound Algorithm

S0: Initialization:

S0.1: Initialization of the contour:
$(\eta_0, \varrho_0) := (0, W),\ (\eta_1, \varrho_1) := (0, 0),\ (\eta_2, \varrho_2) := (L, 0),\ n := 1;$

S0.2: Initialization of the branching tree:
$k := 0,\ z := 0,\ z^* := 0,\ a_0 := 1;$

S1: Branching with respect to allocation points (After the packing of k pieces, choose an allocation point for the $(k+1)$st piece.):

S1.1: Save the allocation points $(\eta_j, \varrho_j),\ j = 1, \ldots, a_k$ which belong to stage k.

S1.2: Set $j_k := 1,\ x := \eta_1,\ y := \varrho_1$ and go to S2.

S1.3: Set $j_k := j_k + 1$. If $j_k > a_k$, then go to S1.4. Set $x := \eta_{j_k},\ y := \varrho_{j_k}$, go to S2.

S1.4: If $k = 0$, then STOP. Go to S2.9 (next piece type).

S2: Branching with respect to piece types (After the packing of k pieces and the fixing of an allocation point choose a piecetype.):
$i := 1;$

S2.1: If $b_{\tau_i} = 0$ then go to S2.2;
If $x + l_i > L$ or $y + w_i > W$, then go to S2.2;
Go to S2.3;

S2.2: $i := i + 1$; If $i \leq \overline{m}$ then go to S2.1; Go to S1.3 (next allocation point);

S2.3: Set $\overline{x} := L - \langle L - x - l_i \rangle_L,\ \overline{y} := W - \langle W - y - w_i \rangle_W$;

S2.4: Test whether the packing pattern with the new packing (i, x, y) is an lsn-packing pattern. If not, then go to S2.2;

S2.5: Equivalence- and dominance tests: If one of the equivalence and dominance tests leads to the cancellation of the subproblem, then go to S2.2.

S2.6: Set $k := k + 1$ (new branching stage),
$t_k := i,\ x_k := x,\ y_k := y,\ \overline{x}_k := \overline{x},\ \overline{y}_k := \overline{y},$
$z := z + l_i w_i,\ b_{\tau_i} := b_{\tau_i} - 1;$

S2.7: Update the current contour. If an improved packing pattern is found then save it.

S2.8: Compute the current remaining area F.
If $z + v(F) > z^*$, then go to S1.

S2.9: Set $z := z - l_i w_i,\ b_{\tau_i} := b_{\tau_i} + 1,\ k := k - 1$, update the contour belonging to stage k and go to S2.2.

The algorithm is based on the concept of contours and realizes a stepwise allocation of pieces. Some or all of the proposed tests with respect to equivalent and/or dominant subproblems may be included in the algorithm.

8 Conclusional Remarks

In this paper the equivalence and dominance of packing patterns is investigated. Since the natural definitions of equivalence and dominance are difficult to be used practically, due to the huge memory requirements, several special forms of equivalence and dominance are proposed which can be used in a b&b algorithm.

In order to show the efficiency of these tests, extensive computational experiments are required. This will be left for future work.

The investigations done in this work are also of importance for developing algorithms for solving three-dimensional packing problems.

Acknowledgement

The author wish to thank Johannes Terno, Uta Sommerweiß, Jan Riehme and Jürgen Rietz for fruitful discussions and helpful remarks.

References

[1] Arenales, M.N., Orlandi, M.A., Salomao, S., Morabito, R.N., A new approach to the solution of two-dimensional non-guillotine cutting problems, Paper presented on the Workshop on Cutting and Packing Problems, Lisbon, July 1993.

[2] Beasley, J.E., An exact two-dimensional non-guillotine cutting tree search procedure, Oper. Res. 33 (1985) 49-65.

[3] Christofides, N., Whitlock, C., An algorithm for two-dimensional cutting problems, Oper. Res. 25 1977) 30-44.

[4] Dyckhoff, H.: A typology of cutting and packing problems. Europ. J. OR 44 (1990) 145-160.

[5] H.Dyckhoff, and U.Finke, Cutting and packing in production and distribution, Physica Verlag, Heidelberg, 1992.

[6] Gilmore, P.C., R.E. Gomory, Multistage cutting stock problems of two and more dimensions, Oper. Res. 13 (1965) 94-120.

[7] Herz, J., Recursive computational procedure for two-dimensional cutting, IBM J. Res. Develop. 16 (1972) 462-469.

[8] Lipovetskij, A.J., Algorithms for non-guillotine rectangular cutting. Reports of the Ufa Aviation Institute (in Russian), 1988.

[9] Scheithauer, G., Algorithms for the container loading problem, Operations Research Proceedings 1991, Springer-Verlag Berlin Heidelberg, 445-452, 1992.

[10] Scheithauer, G., and J. Terno, A heuristic procedure for the container loading problem, Preprint 07-19-90. TU Dresden, 1990.

[11] Sweeney, P.E., and E.I. Ridenour, Cutting and packing problems: A categorized application-oriented research bibliography, Working Paper 610, School of Business Administration, University of Michigan, 1989.

[12] Terno, J., R. Lindemann, G. Scheithauer, Zuschnittprobleme and ihre praktische Lösung, Verlag Harry Deutsch, Thun und Frankfurt/Main, 1987.

[13] Viswanathan, K.V., A Bagchi, An exact best-first search procedure for the constrained rectangular guillotine knapsack problem.

[14] Wang, P.Y., Two algorithms for constrained two-dimensional cutting stock problems, Oper. Res. 31 (1983) 573-586.

Author

Guntram Scheithauer

Institute of Numerical Mathematics

Technical University Dresden

D - 01062 Dresden, Mommsenstr. 13, Germany