The Two-Dimensional, Rectangular, Guillotineable-Layout Cutting Problem with a Single Defect

Vera Neidlein • Andréa C. G. Vianna •
Marcos N. Arenales • Gerhard Wäscher

FEMM Working Paper No. 35, December 2008

Vera Neidlein[1] • Andréa C. G. Vianna[2]
Marcos N. Arenales[3] • Gerhard Wäscher[1]

The Two-Dimensional, Rectangular, Guillotineable-Layout Cutting Problem with a Single Defect – An AND/OR-Graph Approach

December 2008

Abstract: In this paper, a two-dimensional cutting problem is considered in which a single plate (large object) has to be cut down into a set of small items of maximal value. As opposed to standard cutting problems, the large object contains a defect, which must not be covered by a small item. The problem is represented by means of an AND/OR-graph, and a Branch & Bound procedure (including heuristic modifications for speeding up the search process) is introduced for its exact solution. The proposed method is evaluated in a series of numerical experiments that are run on problem instances taken from the literature, as well as on randomly generated instances.

Keywords: Two-dimensional cutting, defect, AND/OR-graph, Branch & Bound

[1] Otto-von-Guericke-University Magdeburg, Germany, Faculty of Economics and Management
[2] Universidade Estadual Paulista, Brazil, Faculdade de Ciências
[3] Universidade de Sao Paulo, Brazil, Instituto de Ciências Matemáticas e de Computaçao

Corresponding author:

Dipl.-Math. oec. Vera Neidlein

Otto-von-Guericke-University Magdeburg
Faculty of Economics and Management
- Management Science -
Postfach 4120
39016 Magdeburg
{vera.neidlein@ww.uni-magdeburg.de}

Table of Contents

1 Introduction

From the late 80's onwards, the number of publications in the area of cutting and packing has grown rapidly. However, in the first place, authors tend to concentrate on developing models and methods for standard problems, while important real-world conditions are only rarely taken into consideration. A typical, often neglected aspect is that the material, which has to be cut down, contains a defect. By simply ignoring the defect, standard methods will only provide sub-optimal solutions.

Therefore, in this paper, a solution approach will be presented that will allow for taking into account material which contains a defect. More precisely, a two-dimensional cutting problem is considered, in which a single plate (large object) has to be cut down into a (weakly heterogeneous) set of small items, which are in unlimited demand. The large plate contains a defect, all cuts are of the guillotine type, and the goal is to maximize the value of the small items provided.

The remaining part of this paper is organized as follows: In section 2, the problem under consideration will be defined and described in detail; furthermore, a literature review about related cutting problems with and also without defects will be given. In section 3, the respective two-dimensional cutting problem without a defect will be discussed. An AND/OR-graph representation, as well as a Branch&Bound procedure, will be described which have been presented earlier in the literature for the solution of this problem. In section 4, this approach will be extended to the case in which the plate contains a defect. In order to evaluate the proposed approach, a series of numerical experiments have been carried out, in which it was run on problem data taken from the literature and data generated randomly. The set-up of the experiments, as well as the implementation of the solution method will be outlined in section 5. Section 6 presents and discusses the results from the experiments. The paper concludes with an outlook on future work in section 7.

2 Fundamentals

2.1 Problem Definition and Characterization

According to the typology of Wäscher, Haußner, and Schumann (2007) the cutting problem to be discussed in this paper can be categorized as a variant of the unconstrained, two-dimensional, rectangular, guillotineable-layout Single Large Object Placement Problem (SLOPP).

The basic underlying cutting problem is characterized by a set of small items (required rectangles), which have to be laid out on a single large object (stock rectangle, stock plate) of given dimensions in a way that the small items do not overlap and lie entirely within the large object. Any description of such a layout is called a cutting pattern.

The small items have to be laid out orthogonally, i.e. in the cutting pattern their edges must be parallel to the edges of the large object, and their orientation is fixed (they cannot be rotated). Furthermore, we impose a guillotine constraint on the cutting pattern, i.e., all small items must be obtainable by a sequence of cuts, each of which dissecting a rectangle (the original large object or a rectangle resulting from a previous cut) into two new rectangles (guillotineable layout). The number of stages that is necessary to cut all items is not restricted.

The assortment of the small items is weakly heterogeneous; the items can be grouped into relatively few classes (types) in which the items are of identical size. The number of times each item type is duplicated in the cutting pattern is not limited (unconstrained problem), and it may happen that an item type does not appear in the pattern at all. In addition, the orientation of each item type is fixed, i.e. it may not be rotated by 90° in order to be laid out on the large object. Each small item type has a particular value, and – since it is not possible to accommodate all small items in the large object – one wants to maximize the total value of the small items in the cutting pattern.

In addition to these basic features which define the unconstrained, two-dimensional, rectangular, guillotineable-layout SLOPP (2D_UG_SLOPP), we now assume that the stock has a (single) defect, i.e. there is a specific region (defined in two dimensions) of the plate, to which no small item must be assigned. More precisely, in the cutting pattern to be determined, no small item must overlap with this defective region. Even though the defect may be of arbitrary shape, we assume that it can be represented by a rectangle, whose edges run in parallel to the edges of the stock plate (see Figure 1).

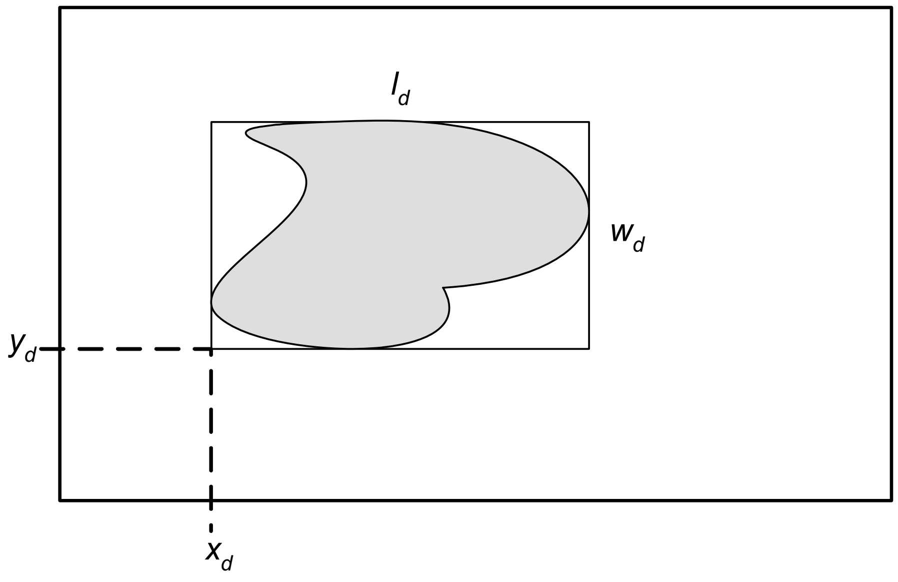


Figure 1: *Representation of the defect*

The two-dimensional SLOPP is a NP-hard problem since it is a generalization of the classic (one-dimensional) Single Knapsack Problem, which is known to be NP-hard (Karp 1972). Furthermore, also the 2D_UG_SLOPP with a single defect is NP-hard since it is a generalization of the two-dimensional SLOPP.

2.2 Formal Representation

Let v_i denote the value and (l_i, w_i) the dimensions of an item of type i (i = 1, …, m), and a_i the number of times an item of type i is assigned to the stock plate, then the unconstrained two-dimensional, rectangular, guillotineable-layout SLOPP can be represented as follows (see Morabito, Arenales, and Arcaro 1992):

$$\begin{aligned} &\max \quad v = \sum_{i=1}^{m} v_i \cdot a_i \\ &\text{s. t.} \quad (a_1,\ldots,a_m) \text{ corresponds to a feasible cutting pattern.} \end{aligned} \tag{1}$$

$(a_1,\ldots,a_m)$ is said to correspond to a feasible cutting pattern if item type i (i = 1, …, m) can be laid out a_i times an on the stock plate in a way that all the above-mentioned constraints are satisfied. A more precise, quite complex representation of these constraints (in particular the guillotineable-layout constraint) has been given by Messaoud, Chu, and Espinouse (2008); we will not go into details here because such a formal representation is not necessary for the presentation of our approach.

2.3 Literature Review

The 2D_UG_SLOPP has been studied extensively, in particular over the last thirty years. Gilmore and Gomory (1966) describe an exact dynamic programming approach for two-dimensional knapsack functions, which are used as a basis to determine solutions for the2D_UG_SLOPP. An exact recursive procedure using discretization sets of all necessary locations of cuts is given by Herz (1972). Beasley (1985a) presents an exact recursive approach using dynamic programming, based on the method of Gilmore and Gomory (1966), and heuristic modification which considers only a subset of the discretization sets. In the same article, a similar approach is used for staged problems. An exact algorithm performing a tree search on AND/OR-graphs, as well as a heuristic search strategy to improve computability, is described by Morabito, Arenales, and Arcaro (1992). A heuristic approach for the 2D_UG_SLOPP as well as for other versions of the 2D_SLOPP can be found in the paper by Fayard, Hifi, and Zissimopoulos (1998); they reduce the problem to a series of one-dimensional knapsack problems which are solved by dynamic programming. For the same problem types, Alvarez-Valdés, Parajón, and Tamarit (2002) describe a tabu search algorithm including GRASP. G and Kang (2002) present an upper bound for the 2D_UG_SLOPP and also for the non-guillotineable layout type, which is calculated by solving two one-dimensional knapsack problems.

Other versions of the 2D_SLOPP have also been widely studied. Christofides and Whitlock (1977) present a Branch & Bound strategy to solve a guillotineable-layout problem exactly with upper bounds for the number of times an item type can be cut (constrained problem). They combine dynamic programming and a transportation routine for the determination of upper bounds. For the same problem type, Wang (1983) presents two algorithms that successively put together horizontal and vertical builds of item types to fill the large object. Christofides and Hadjiconstantinou (1995) present an improvement of the algorithm of Christofides and Whitlock (1977); they use a state-space relaxation of a dynamic programming formulation of the problem to calculate an upper bound required for limiting the Branch & Bound search. Morabito and Arenales (1996) extend the AND/OR-graph approach to solve constrained and staged problems, exactly and heuristically. Parada, Pradenas, Solar, and Palma (2002) develop a combination of a genetic algorithm and a search procedure on an inverted AND/OR-graph. In the article by Alvarez-Valdés, Parreño, and Tamarit (2005), a GRASP algorithm and a reasonable choice of its parameters are described.

Hifi and M’Hallah (2005) present an exact Branch & Bound algorithm including a new upper bound for the constrained guillotineable-layout 2D_SLOPP with two cutting stages. Three different upper bounds for any Branch & Bound algorithm, derived by LP, Knapsack and Lagrangean relaxation, are introduced in Beasley (1985c). The constrained, non-guillotineable layout 2D_SLOPP is considered by Beasley (1985b), who provides a 0-1 linear programming model and corresponding exact depth-first tree search (Branch & Bound) procedure.

In general, cutting problems with defects have only been studied rather limitedly. Gilmore and Gomory (1965) use linear programming on the one-dimensional and the two-dimensional Single Stock Size Cutting Stock Problem, and for both cases, they describe a recursive formula for the value of an item type where the mentioned value depends on its position on the large object, i.e. the large object consists of different qualities, one of which may be a non-usable defect. The three-staged 2D_UG_SLOPP with multiple defects and a non-linear value function for the item types is considered by Hahn (1968), who describes an adaptation of the dynamic programming approach by Gilmore and Gomory (1966). Scheithauer and Terno (1988) present an improved dynamic programming solution method for the same type of problem with a non-rectangular large object. Herz (1972) mentions without giving any details that it is possible to adapt his recursive algorithm for the 2D_UG_SLOPP with multiple defects.

More extensive or specialized, less standardized problem variants have also been a field of research. The combined process of cutting a felled tree first into logs and then into lumber, considering quality and shape variations (defects) of the tree by use of a profile scanner, is presented by Faaland and Briggs (1984). They use a staged dynamic programming model to solve this problem taken from practice. A special kind of one-dimensional cutting problem is solved by Sarker (1988) using dynamic programming. Here, a single large object with several punctual defects is considered, and the goal is to maximize the total value that can be achieved by cutting only through defects, under the condition that the value of a piece cut depends on its length and the number of contained defects. Aboudi and Barcia (1998) consider a one-dimensional cutting problem that occurs in paper mills – a roll of paper containing one defect is to be cut vertically into a given set of sheets; one wishes to determine a permutation of the sheets which minimizes the length of those sheets containing a defect. (The authors give a 0-1 integer programming model which can be relaxed (surrogate relaxation) to achieve strong bounds for a Branch & Bound method including several heuristics. A complex one-dimensional cutting and wrapping problem in the textile industry is described by Özdamar (2000); the cutting of fabric lengths into shorter pieces and the sorting of those pieces into different quality grades depending on contained defects is solved by a simulated annealing approach with occasional mutations.

An auxiliary problem, namely the one of finding all usable rectangles on a large rectangle containing several pairwise disjoint defects, is solved by a constructive algorithm by Twisselmann (1999).

The 2D_UG_SLOPP with a single defect, as it is discussed in this paper, has been introduced into the literature by Carnieri, Mendoza, and Luppold (1993). They present a heuristic solution based on dynamic programming, which extends the classic approach by Gilmore and Gomory (1963), and including a Branch & Bound search where the necessary bounds are calculated assuming the large object had no defect. The same problem type has also been studied by Vianna and Arenales (2006); we present an extension of their approach here.

3 The AND/OR-Graph Approach to the 2D_UG_SLOPP

In this section, a summary of the AND/OR-graph approach (Morabito, Arenales and Arcaro 1992) to the unconstrained, two-dimensional, rectangular, guillotineable-layout SLOPP (without defects) is presented, which, in short, consists of representing all cutting patterns as complete paths in a specific graph and enumerating them implicitly in order to find an optimal solution. This approach will be extended to problems with a single defect in Section 4.

3.1 Guillotine Cuts and Intermediate Plates

Applying a guillotine cut to a plate results in two new rectangles, called intermediate plates, which can be cut down further. The process of cutting terminates when a "final" (required) item or a piece of waste is obtained. Consequently, any guillotineable-layout cutting pattern can be generated just by examining the different guillotine cuts, which can be applied to each intermediate plate. An intermediate plate (say, plate *N*) has a certain length L_N and width W_N , and will be denoted by $N = (L_N, W_N)$. The set $M(N)$ of small items that can still be cut from *N* is given by

$$M(N) := \{i : l_i \le L_N \text{ and } w_i \le W_N,\ i \in \{1,...,m\}\}. \tag{2}$$

Herz (1972) has shown that, without loss of generality, the positions of the cuts on the large object can be taken as non-negative integer combinations of the dimensions of the small items, i.e. from the set

$$X = \left\{x:\ x = \sum_{i=1}^{m} \alpha_i l_i,\ l_0 \le x \le L - l_0,\ \alpha_i \ge 0 \text{ and integer},\ i = 1,...,m\right\} \tag{3}$$

for the vertical cuts and from the set

$$Y = \left\{y:\ y = \sum_{i=1}^{m} \beta_i w_i,\ w_0 \le y \le W - w_0,\ \beta_i \ge 0 \text{ and integer},\ i = 1,...,m\right\} \tag{4}$$

for the horizontal cuts, where $l_0 = \min\{l_i,\ i = 1,...,m\}$, and $w_0 = \min\{w_i,\ i = 1,...,m\}$. *X* and *Y* are called discretization sets. Likewise, the discretization sets for an intermediate plate $N = (L_N, W_N)$ are given by

$$X(N) = \left\{x:\ x = \sum_{i \in M(N)} \alpha_i l_i,\ l_0 \le x \le L_N - l_0,\ \alpha_i \ge 0 \text{ and integer},\ i \in M(N)\right\} \tag{5}$$

and

$$Y(N) = \left\{y:\ y = \sum_{i \in M(N)} \beta_i w_i,\ w_0 \le y \le W_N - w_0,\ \beta_i \ge 0 \text{ and integer},\ i \in M(N)\right\} \tag{6}$$

which can be determined by means of the recursive formula of Christofides and Whitlock (1972), or by its revised version as presented in Morabito and Arenales (1996).

Therefore, given an intermediate plate $N = (L_N, W_N)$, a vertical cut at position $x \in X(N)$ produces two new plates: $N_1 = (x, W_N)$ and $N_2 = (L_N - x, W_N)$; a corresponding formula applies to a horizontal cut $y \in Y(N)$. Although the number of intermediate plates is finite when only cuts in X and Y are considered, it can be enormous for a practical problem instance.

3.2 AND/OR-Graphs

An AND/OR-graph $G = (V, E)$ is a special type of graph, where V is a set of nodes and $E = \{e_1,...,e_s\}$ a set of directed arcs. Each arc e_{ju} links a node j to a set S_u of nodes (in an ordinary graph the set S_u consists of just one node):

$$e_{ju} = (j, S_u),\ j \in V,\ S_u \subseteq V.$$

The nodes in S_u are called successors of j, and j is called predecessor of nodes in the set S_u. In this paper, S_u will always consist of a pair $\{p,q\}$ of nodes (in case of an AND-arc, see below), or a unit set $\{p\}$ (ordinary arc leading to a final node, see below).

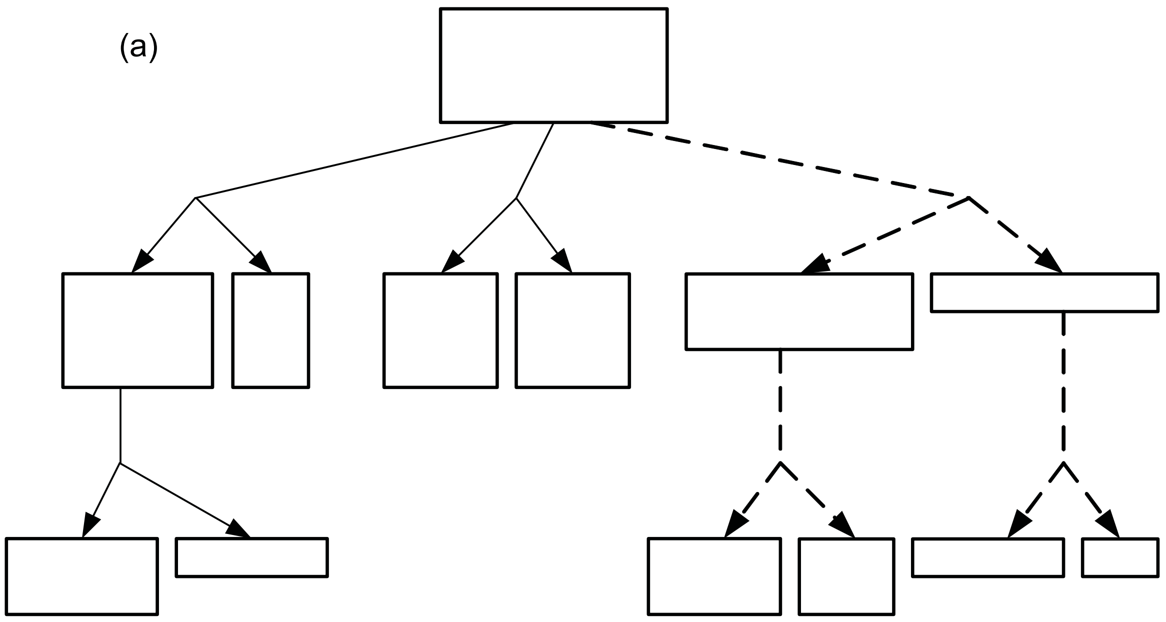
When following a path through the graph, one can choose between several arcs that emerge from a node (OR-arcs), but one has to follow both branches of the chosen arc (AND-arc). An example can be seen in Figure 2 (a). This type of graph provides an appropriate tool for the representation of a cutting process – each node stands for a plate (stock or intermediate), each arc $e_{ju} = (j,\{p,q\})$ for a guillotine cut that separates a plate/node j (start node of e_{ju}), into a pair of new plates $\{p,q\}$ (end nodes of e_{ju}).

At some point throughout the cutting process, one may decide not to perform any further cuts on a particular plate, e.g. because the dimensions of this plate indicate that it has to be considered as waste, or it has the exact dimensions of a small item, or because the optimal cutting pattern for that plate is known. This situation can be represented in the AND/OR-graph by introducing an ordinary arc, called 0-cut, for each node j (not depicted in Figure 2 (a)), i.e. an arc $e_{ju} = (j, p)$ that copies exactly the plate of node j into node p. If such an arc has been chosen in the path, no further cut is made on plate in node p, and p is called a final node of the path. The value of a final node equals 0 in case it represents waste. If the final node represents a specific small item i, than its value is identical with the value of i, and if the final node represents a plate for which the optimal cutting pattern is known, then its value is identical with the value of this pattern.

Any cutting pattern for the large object (stock plate) can be determined from the AND/OR-graph as follows: Starting from the root node (stock plate), choose one and only one arc (AND-arc, or 0-cut arc), and from each node pointed to by this chosen arc choose again one and only one arc, and so on, until all visited nodes are final nodes. This sequence of paths is called a complete path of the AND/OR-graph, and it corresponds to a cutting pattern. The broken lines of Figure 2 (a) indicate a complete path, which corresponds to the cutting pattern depicted in Figure 2 (b).

The value of a complete path (or of its corresponding cutting pattern) is the sum of the values of all its final nodes. Therefore, problem (1) can be reformulated as the problem of finding a most valuable complete path in the AND/OR-graph.

(a)



(b)

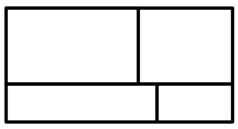


Figure 2: *AND/OR-graph with a particular highlighted path (a) and the corresponding cutting pattern (b)*

3.3 Upper and Lower Bounds

In order to describe a scheme for the implicit enumeration of the complete paths (i.e., cutting patterns), we define upper and lower bounds for the value of the optimal cutting pattern for a given plate $N = (L_N, W_N)$. The area provides a straightforward way to compute an upper bound $\mathcal{UB}(N)$ for the objective function value which can still be generated by cutting down N:

$$\begin{aligned} \mathcal{UB}(N) = \mathcal{UB}(L_N, W_N) = \ & \max \sum_{i \in M(N)} v_i \cdot a_i \\ & \text{s.t.} \sum_{i \in M(N)} a_i (l_i w_i) \le L_N \cdot W_N \quad \text{(area utilization)} \\ & \qquad a_i \ge 0, \quad i \in M(N) \end{aligned} \tag{7}$$

Trivially, the optimal objective function value of problem (2) is obtained by

$$\mathcal{UB}(L_N, W_N) = L_N \cdot W_N \cdot \max \left\{ \frac{v_i}{l_i w_i} : \ i \in M(N) \right\}. \tag{8}$$

Computation of a lower bound $\mathcal{LB}(N)$ for the objective function value related to a plate $N = (L_N, W_N)$ can be based on homogeneous cutting patterns, which contain only small items of a single type (cf. Figure 3).

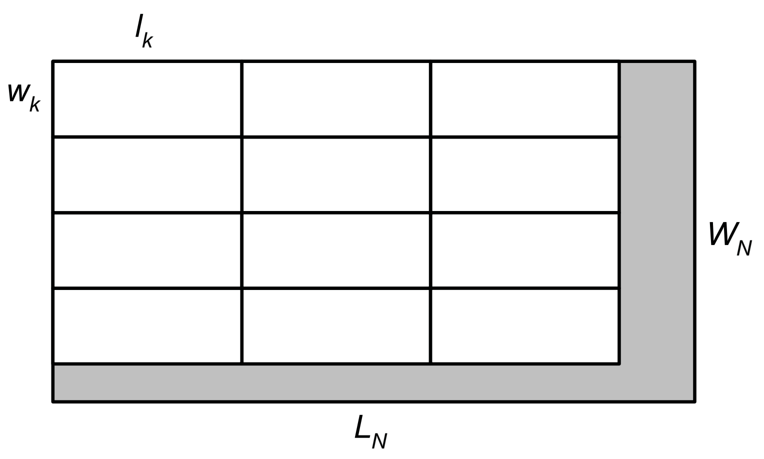


Figure 3: *Homogeneous cutting pattern for plate (L_N,W_N), consisting of item type k*

Given a plate $N = (L_N, W_N)$, the maximum number of times that item type $k = (l_k, w_k)$ appears in a homogeneous pattern is $\left\lfloor \frac{L_N}{l_k} \right\rfloor \cdot \left\lfloor \frac{W_N}{w_k} \right\rfloor$. Thus the corresponding objective function value is $v_k \cdot \left\lfloor \frac{L_N}{l_k} \right\rfloor \cdot \left\lfloor \frac{W_N}{w_k} \right\rfloor$, and the best homogeneous cutting pattern gives a lower bound for the objective function value for plate $N = (L_N, W_N)$:

$$\mathcal{LB}(N) = \mathcal{LB}(L_N, W_N) = \max\left\{ v_i \cdot \left\lfloor \frac{L_N}{l_i} \right\rfloor \cdot \left\lfloor \frac{W_N}{w_i} \right\rfloor : \ i \in M(N) \right\} \tag{9}$$

Computation of both upper and lower bound requires very little time, the number of necessary operations depends linearly on the number of small item types. In section 4.2, we give more specific formulas to deal with defective plates.

3.4 A Branch & Bound Algorithm

Let $\mathcal{LB}(N)$ denote a lower bound and $\mathcal{UB}(N)$ an upper bound, respectively, for the objective function value of node/plate N, and let $\mathcal{V}(N)$ be the currently known best objective function value for node N. Of course, $\mathcal{LB}(N) \le \mathcal{V}(N) \le \mathcal{UB}(N)$ holds. Furthermore, let S be the set of nodes (original or intermediate plates) still to be dealt with. Then the following Branch & Bound procedure can be used for the determination of an optimal solution for the 2D_UG_SLOPP:

1. Set $S := \{(L, W)\}$.
2. Choose a node $N = (L_N, W_N)$ from S and delete it from S. Set $\mathcal{V}(N) := \mathcal{LB}(N)$. For all possible successors N_1 and N_2 that can result from a vertical cut on N chosen from the discretization set *X(N)* or from a horizontal cut on N chosen from the discretization set *Y(N)*, do:
    a) If N_1 represents waste, i.e. $M(N_1)=\emptyset$, then make N_1 a final node; otherwise include N_1 in S. Repeat this step for N_2.

b) If $\mathcal{V}(N) < \mathcal{LB}(N_1) + \mathcal{LB}(N_2)$ (i.e., an improved objective function value can be obtained by this cut), then update $\mathcal{V}(N) := \mathcal{LB}(N_1) + \mathcal{LB}(N_2)$; also update recursively the currently known best objective function values of all predecessors of *N* up to the original plate (root node)

c) If $\mathcal{V}(N) \geq \mathcal{UB}(N_1) + \mathcal{UB}(N_2)$ (i.e., the cut can not result in an improved objective function value), then remove N_1 and N_2 from *S*.

3. If S is not yet empty, go back to step 2.
4. STOP! The currently known best value for the root node is the optimal objective function value. The corresponding cutting pattern can be determined by identifying the complete path that provides this objective function value.

3.5 Heuristic Modifications

Obviously, computing times for the Branch & Bound procedure depend very much on the number of nodes that have to be considered, which can be enormous. Therefore, Morabito, Arenales, and Arcaro (1992) developed several heuristics, which are modifications of the original procedure, that reduce the computing times drastically, but still lead to reasonably good solutions.

Heuristic 1: Use of promising cuts only

A cut is considered to be promising if the sum of the upper bounds of the resulting plates N_1 and N_2 is substantially larger than the currently known best value of *N*, that is, $\mathcal{UB}(N_1) + \mathcal{UB}(N_2) > (1+\alpha)\mathcal{V}(N)$, and if the sum of the lower bounds of the successors is only slightly smaller or even larger than the lower bound of *N*, that is, $\mathcal{LB}(N_1) + \mathcal{LB}(N_2) > (1-\beta)\mathcal{LB}(N)$, where α and β are parameters to be chosen empirically.

Heuristic 2: Heuristic search strategy

Morabito, Arenales and Arcaro (1992) combine depth-first and hill climbing strategies. They arbitrarily choose a depth limit (a maximum number of successive cuts, or the length of the path) and compute from the root node (initial plate) the best (exact or heuristic) complete path not exceeding this limit, discarding all others paths (a pure hill-climbing search would be obtained for a depth limit of 1). Then, for the given depth limit, all nodes are considered as root nodes and chosen for further investigation, that is, from each one the best complete path up to the depth limit is calculated, and so on. For details, see Morabito, Arenales, and Arcaro (1992) or Arenales and Morabito (1995).

4 Dealing with a Single Defect

In this section, the AND/OR-graph approach will be modified in such a way that a single defect on the plate can be considered explicitly. As has been noted above (see Figure 1), we assume that the defect *d* can be represented by a rectangle of length l_d and width w_d: $d = (l_d, w_d)$. The location of the defect on the stock plate is indicated by the coordinates of its lower left corner (x_d, y_d).

4.1 Dimensions and Location of a Defect on an Intermediate Plate

If a cut is performed on a defective (stock or intermediate) plate $N = (L_N, W_N)$, we need to know what the dimensions of the defects are on the resulting plates and where they are located. The following line of argumentation will be presented for a vertical cut only, but holds likewise for a horizontal cut. Let the cut (chosen from the discretization set *X*) be at position *z*, then the dimensions of the plates N_1 and N_2 to be generated are (z, W_N) and $(L_N - z, W_N)$, respectively, i.e. $N_1 = (z, W_N)$ and $N_2 = (L_N - z, W_N)$. Figure 4 shows the possible positions of the vertical cut *z* in relation to the defect.

Figure 4 (a) depicts a case in which only plate N_2 will contain a defect; likewise in Figure 4 (c) only plate N_1 will have a defect. Both plate N_1 and N_2 will have a defect in the case depicted in Figure 4 (b). Table 1 presents the size of the defects and their locations (indicated by the coordinates of the lower left corner) for these three cases.

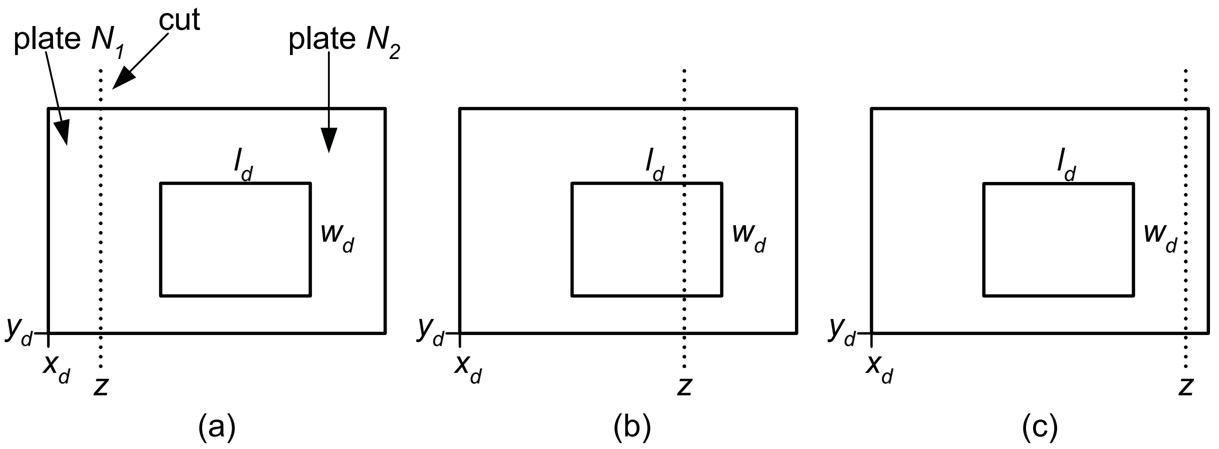


Figure 4: *Possible positions of a vertical cut*

	plate N_1		plate N_2	
	dimensions of defect	location of defect	dimensions of defect	location of defect
case (a)	-	-	(l_d, w_d)	$(x_d - z, y_d)$
case (b)	$(z - x_d, w_d)$	(x_d, y_d)	$(x_d + l_d - z, w_d)$	$(0, y_d)$.
case (c)	(l_d, w_d)	(x_d, y_d)	-	-

Table 1: *Size and location of defects on plates N_1 and N_2*

4.2 Upper and Lower Bounds for a Defective Plate

An upper bound for a node representing a plate $N = (L_N, W_N)$ with a defect of size (l_d, w_d) can be obtained by a simple modification of the area-utilization bound (4). Since the useable plate area now is $L_N W_N - l_d w_d$, we obtain

$$\mathcal{UB}(N) = \mathcal{UB}(L_N, W_N) = (L_N W_N - l_d w_d) \cdot \max\left\{\frac{v_i}{l_i w_i} : \ i \in M(N)\right\}. \qquad (10)$$

The modification for the determination of a lower bound for a node representing a defective plate is less straightforward. Figure 5 and Figure 6 demonstrate why the objective function value of the simple homogeneous cutting pattern cannot be used as a reasonable lower bound any more.

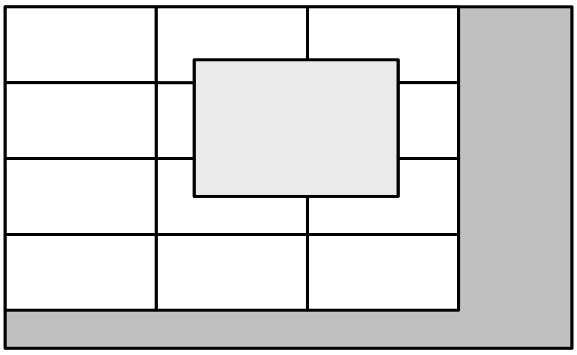


Figure 5: *Simple homogeneous cutting pattern with items overlapping the defect*

If the calculation of the lower bound would be carried out by ignoring the items overlapping the defect, the bound corresponding to Figure 5 would be based on six items. Figure 6, on the other hand, presents a solution obtained by shifting a block of small items as far as possible to the right, allowing the computation of the bound to be based on having assigned nine items.

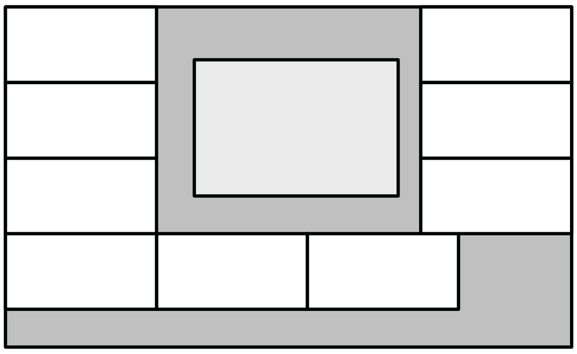


Figure 6: *Modified homogeneous cutting pattern without items overlapping the defect*

Consequently, the determination of a lower bound will be based on the identification of (rectangular) “non-defective regions” of maximal size. A region is called non-defective if it can be obtained from *N* by a series of successive guillotine cuts and does not contain a defect. It is of maximal size if each of the guillotine cuts cannot be moved further towards the defect without creating a defect on this region.

As long as the defect is not located directly adjacent to an edge of plate *N*, there exist 14 different ways in which *N* can be partitioned into non-defective regions of maximal size by a series of guillotine cuts, and there are always four such regions in any partition (cf. Figure 7; the regions “on the left” of the defect are denoted by A, the ones “on top” by B, those “on the bottom” by C and the ones “on the right” by D.). As becomes evident, each of the four (maximal) regions can be rectangles of four different sizes (types). It goes without mentioning that some of the partitions will be degenerated if the defect is located adjacent to an edge of the plate.

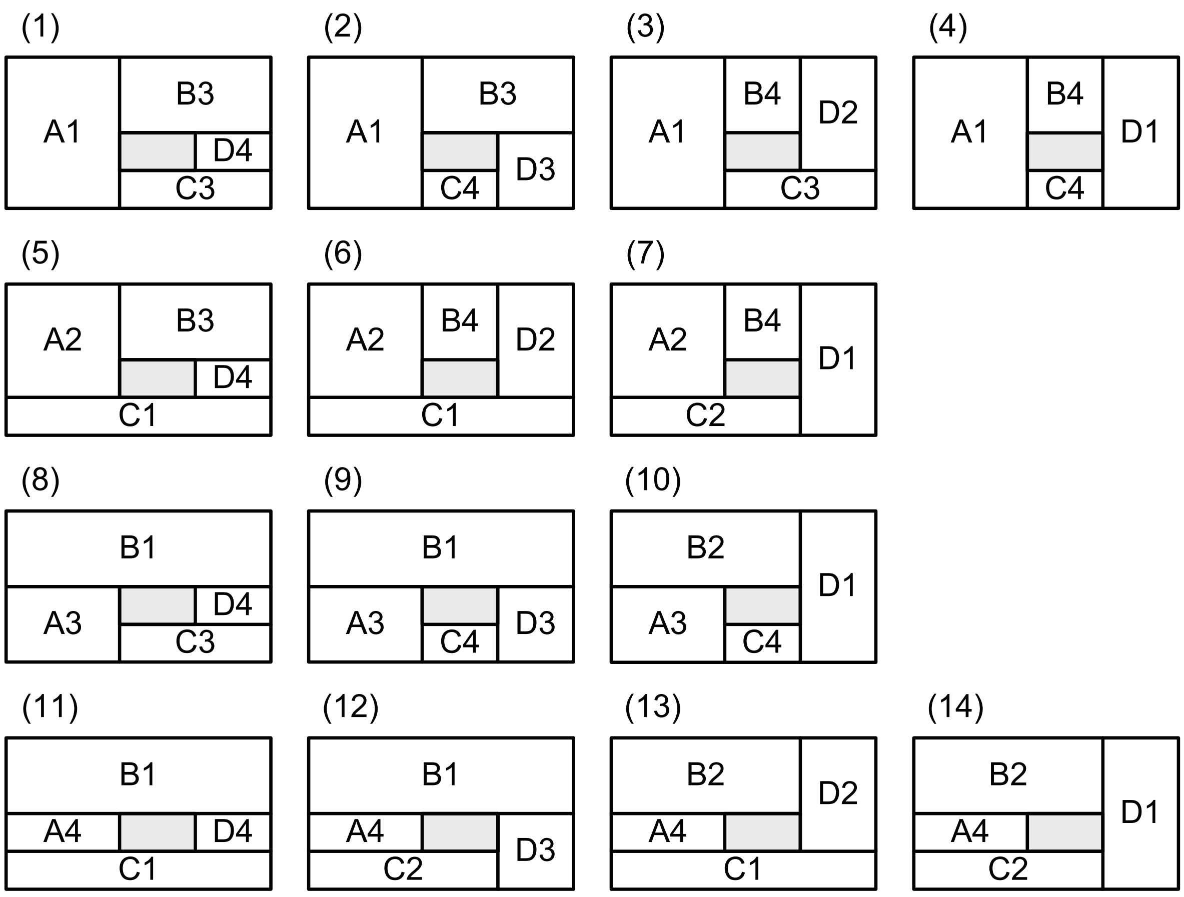


Figure 7: *The 14 different ways to divide a plate with one defect into four non-defective regions of maximal size*

A lower bound for a plate *N* with a defect can now be computed as follows: For each type of non-defective region, a homogeneous cutting pattern is determined which provides – according to (5) – the best objective function value (across all item types which can be accommodated at least once by the region). This gives a lower bound for each type of non-defective region. Then, for each partition (cf. Figure 7), the lower bounds for the respective non-defective regions are added up, resulting in a lower bound for each partition. Finally, the lower bound for *N* can simply be computed as the maximum of the lower bounds of all 14 partitions.

In order to determine the (maximal) homogeneous cutting patterns on the (maximal) non-defective regions, the dimensions of these regions have to be known. For a plate $N=(L_N,W_N)$, on which the (lower left corner of the) defect of size (l_d,w_d) is located in position (x_d,y_d), they can be taken from Table 2.

Regarding the computing times, it can be noted that the number of nodes in the AND/OR-graph may grow exponentially in the number of small items. However, the number of calculations needed for the determination of the lower bound at a single node is only linear in the number of the small items (as it is the case for the 2D_SLOPP without a defect). Therefore, there will be no significant increase in the computing times if compared to the times needed for solving the problem without any defect.

type of non-defective region	horizontal dimension	vertical dimension
A1	x_d	W_N
A2	x_d	$W_N - y_d$
A3	x_d	$y_d + w_d$
A4	x_d	w_d
B1	L_N	$W_N - (y_d + w_d)$
B2	$x_d + l_d$	$W_N - (y_d + w_d)$
B3	$L_N - x_d$	$W_N - (y_d + w_d)$
B4	l_d	$W_N - (y_d + w_d)$
C1	L_N	y_d
C2	$x_d + l_d$	y_d
C3	$L_N - x_d$	y_d
C4	l_d	y_d
D1	$L_N - (x_d + l_d)$	W_N
D2	$L_N - (x_d + l_d)$	$W_N - y_d$
D3	$L_N - (x_d + l_d)$	$y_d + w_d$
D4	$L_N - (x_d + l_d)$	w_d

Table 2: *Dimensions of the 16 (maximal) non-defective regions*

5 Numerical Experiments

In order to determine the performance of the above-described algorithm, a series of numerical experiments has been carried out. A first set of problem instances, which has also been used by Vianna and Arenales (2006) for benchmarking purposes, has been taken from the paper by Carnieri, Mendoza, and Luppold (1993). Even though this set is of rather limited size, it will be considered here since – according to the best of the authors' knowledge – this is the only one related to the 2D_UG_SLOPP with a single defect that has been presented in the literature so far. A second, more substantial set of problem instances has been generated randomly.

5.1 Data Sets

5.1.1 The Data Set of Carnieri, Mendoza, and Luppold

Carnieri, Mendoza and Luppold (1993) consider eight problem instances: one set of item types (see Table 3) combined with 8 different defects, differing with respect to location and size (see Table 4). For each item type and defect, the dimensions as well as the relative size (percentage of area of the large object) are given. The dimensions of the large object are (L, W) = (200, 100).

item type i	length l_i	width w_i	value v_i	relative size [%]
1	40	30	10	6.00
2	68	26	12	8.84
3	50	20	8	5.00
4	60	35	18	10.50
5	45	22	9	4.95

Table 3: *Item types of the data set of Carnieri, Mendoza, and Luppold (1993)*

defect no.	position of lower left corner (x_d,y_d)	dimensions (l_d,w_d)	relative size [%]
1	(50,100)	(4,5)	0.10
2	(40,100)	(4,5)	0.10
3	(60,100)	(6,5)	0.15
4	(20,125)	(10,7)	0.35
5	(71,125)	(8,7)	0.28
6	(30,30)	(10,10)	0.50
7	(40,80)	(18,30)	2.70
8	(40,80)	(18,38)	3.42

Table 4: *Defects of the data set of Carnieri, Mendoza, and Luppold (1993)*

It should be noted that the values of the item types are not proportional to their area (weighted version of the 2D_UG_SLOPP) and that the item types are quite large w.r.t. the size of the large object. The defects, on the other hand, appear to be rather small.

5.1.2 Randomly Generated Data Sets

Identification of Problem Parameters

The second set of problem instances refers to the unweighted version of the 2D_UG_SLOPP, i.e. it has been assumed that the values (in monetary units) of all item types are proportional to their sizes (in square units). Under this assumption, any instance of the 2D_UG_SLOPP with a single defect is completely characterized by the vector

$$((L,W),\ (l_1,w_1),\ (l_2,w_2),\ldots,\ (l_m,w_m),\ (l_d,w_d),\ (x_d,y_d)).$$

In other words, in order to obtain a test problem instance,

- the dimensions of the large object (L,W),
- the dimensions of the item types (l_i,w_i), i = 1,..,m,
- the number m of different item types to be placed,
- the dimensions of the defect (l_d,w_d), and
- the location of the defect (x_d,y_d),

(problem parameters) have to be specified. Since each instance contains information about the dimensions of the large object, about the dimensions of the defect and about its location, it seems reasonable to refer to m, the number of small item types, as the problem size.

Parameter Specification

In all experiments, the size of the large object has been fixed to 360,000 square units (sq. u.), for which three different combinations of length and width have been considered (see Table 5).

(L,W)	(600,600)	(900,400)	(1200,300)
length-to-width ratio	1:1	2.25:1	4:1

Table 5: *Dimensions of the large object*

With respect to the size of the item types in relation to the large object, three classes have been distinguished: item types of "small" size (max. 1% of the area of the large object, i.e. 1-3,600 square units), "large" size (3-5% of the area of the large object, i.e. 10,800-18,000 square units), and "both small and large" size (max. 5% of the area of the large object, i.e. 1-18,000 square units).

The actual size a_i (in sq. u.) of each item type i was considered as the realization $\hat{a}_i$ of a random variable, which is generated from the respective size range. $\hat{a}_i$ provides the basis for the determination of the corresponding dimensions (length l_i, width w_i) of i. In order to avoid "degenerated" item sets (i.e. sets which contain very long and narrow items in the first place), the following procedure has been chosen (for details see Neidlein and Wäscher 2008): At first, from the interval $[0.1, 0.9]$ a realization $\hat{b}_i$ of a random variable b_i for the relative length of i is determined, which is defined in the following way:

$$b_i = \frac{l_i}{l_i + w_i}. \tag{11}$$

Then approximate values $\hat{l}_i$ and $\hat{w}_i$ for l_i and w_i, respectively, are calculated as follows:

$$\hat{l}_i = \sqrt{\frac{\hat{a}_i \hat{b}_i}{1-\hat{b}_i}} \text{ and } \hat{w}_i = \frac{\hat{a}_i}{\hat{l}_i}. \tag{12}$$

In order to obtain l_i and w_i, these values are rounded down:

$$l_i = \lfloor \hat{l}_i \rfloor, w_i = \lfloor \hat{w}_i \rfloor \tag{13}$$

This provides an item type of size $a_i = l_i \cdot w_i$, close to the value $\hat{a}_i$ originally generated. By limiting b_i to the interval

$$\frac{\hat{a}_i}{\hat{a}_i + W^2} \le b_i \le \frac{L^2}{\hat{a}_i + L^2} \tag{14}$$

it is guaranteed that every item type fits into the large object, i.e. that l_i will not exceed *L* and w_i will not exceed *W*.

small (~3,600 sq. u.)	(l_d, w_d)	(60,60)	(105,35)	(35,105)
	area [sq. u.]	3,600	3,675	3,675
medium (~14,400 sq. u.)	(l_d, w_d)	(120,120)	(210,70)	(70,210)
	area [sq. u.]	14,400	14,700	14,700
large (~28,800 sq. u.)	(l_d, w_d)	(170,170)	(285,95)	(95,285)
	area [sq. u.]	28,900	27,075	27,075
	length-to-width ratio	1:1	3:1	1:3

***Table 6:** Types of the defect*

The number of different item types to be placed on the large object was chosen to be 5 and 10.

The size of the defect was fixed to (approximately) 1% (3,600 sq. u.), 4% (14,400 sq. u.) and 8% (28,800 sq. u.) of the area of the large object. Small deviations from the exact percentage values have been permitted in order to allow for a better numerical manageability of the length-to-width ratios. For each of these size classes, three different combinations of lengths and widths have been introduced, which shape the defect as a square, as a "lying" (i.e. horizontal) rectangle, and as a "standing" (i.e. vertical) rectangle. The exact size and dimensions can be taken from Table 6.

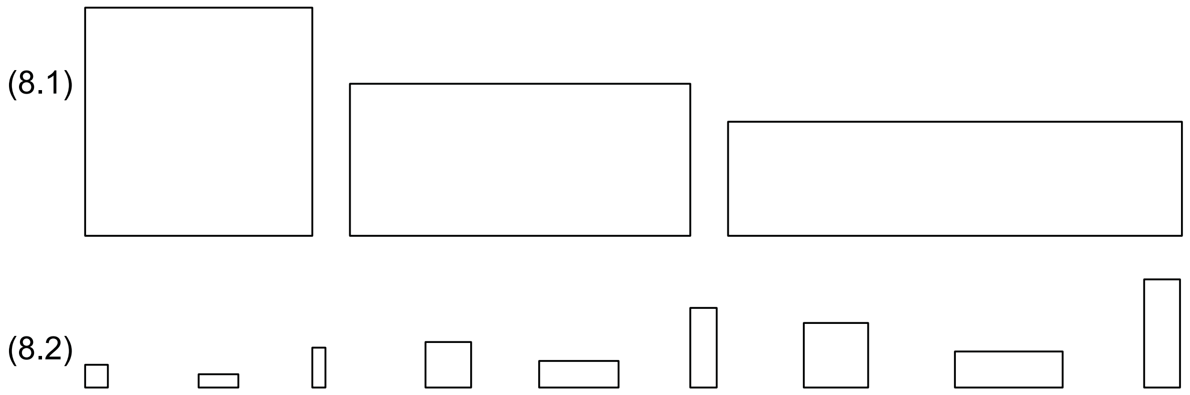


***Figure 8:** Sizes and shapes of large objects (8.1) and defects (8.2)*

The position (x_d, y_d) of the lower left corner of the defect on the large object was generated randomly in the intervals $[0, L - l_d]$ for x_d and $[0, W - w_d]$ for y_d. Figure 8 gives an illustration of the sizes and shapes of the large objects and the defects.

Basic Problem Classes and Problem Instances

The properties introduced above for the dimensions of the large object (3), for the size of the small item types (3), and the size of the problem (2) define (3 x 3 x 2 =) 18 classes of problem data (basic problem classes). For each of these classes, 30 instances have been generated by means of an adaptation of the problem generator of Neidlein and Wäscher (2008).

Each instance of a basic problem class has been combined with each of the nine different defect dimensions defined in Table 6. Thus, in total (18 x 30 x 9 =) 4860 problem instances have been included in the numerical experiments. The instances are available at www.ovgu.de/mansci/materials.

5.2 Implementation of the Algorithm

As has been described in section 3.5, in order to speed up the algorithm, a heuristic which considers promising cuts only is used. The values of the parameters α and β have both been set to 0.1. The depth-first / hill-climbing strategy is used with a depth bound of three. The algorithm was encoded using Borland Pascal Version 7. All experiments were run under Windows XP on a microcomputer with 1.7 GHz core memory clock speed and 512 MB RAM.

6 Computational Results

6.1 Instances of Carnieri, Mendoza, and Luppold

Since this data set contains instances of the weighted version of the 2D_UG_SLOPP, the quality of a solution can be measured by the total value of the small items cut from the large object. Table 7 presents the results obtained by Carnieri, Mendoza, and Luppold (1993, p. 71), as well as the results and computing times of the AND/OR-graph approach presented in this paper. For all instances, the computing times of the AND/OR-graph approach turned out to be very reasonable. Please note that Carnieri, Mendoza, and Luppold (1993) do not report any computing times in their paper.

The AND/OR-graph approach provided better solutions for two instances, while equally good solutions were found for the remaining ones. In additional tests we modified the heuristic search parameters to allow for a deeper AND/OR-graph search – and thus allocated more computing time to the algorithm – in order to prove that the obtained solutions were optimal. However, neither did the algorithm terminate at an optimal solution, nor was any further improvement of the objective function value achieved for any of the problem instances. We believe that the obtained solutions are optimal, even though we are not able to prove it.

instance no.	Carnieri et al. (1993)	AND/OR-graph approach	
	total value of items cut	total value of items cut	computing time [sec]
1	166	166	0.52
2	160	160	0.77
3	162	162	1.77
4	158	160	0.27
5	164	164	4.11
6	164	164	1.44
7	157	158	1.07
8	154	154	0.50

Table 7: *Total values of items cut and computing times for the instances of Carnieri, Mendoza, and Luppold (1993)*

6.2 Randomly Generated Instances

We note that – under the assumptions made – the size of the (non-defective) area of the large object which is not covered by small items (waste, trim loss) can be used as a measure for the quality of each solution. Since the optimal solutions are not known, the waste is reported as a percentage of the useable area of the large object, i.e. of $LW - l_d w_d$.

Across all problem instances, an average percentage of waste of 5.0333% was observed; the average computing time per instance amounted to 10.25 sec.

Table 8 provides waste and computing times for the 18 basic problem classes (containing 270 instances each) in greater detail. It can already be seen that the waste decreases (i.e. the solution quality improves) and computing times increase with an increase in the number of item types. This observation will be confirmed by the subsequent analysis.

In Table 9, results from Table 8 have been aggregated with respect to the number of item types and the size of the item types. Additionally, the number of instances is depicted for which zero-waste solutions have been obtained. 2,673 instances are contained in each of the two problem classes of Table 9.1, and 1,782 instances in each of the three problem classes of Table 9.2. Computing times increase drastically with an increasing number of item types, while the solution quality (both in terms of waste per instance and the number of zero-waste instances per class) improves (cf. Table 9.1). The latter observation can be explained easily by the fact that a larger number of item types allows for a larger number of combinations of item types (cutting patterns), which – on the other hand – take more time to evaluate. The same reasoning can be put forward with respect to the observation that computing times are longer and solution quality is better for small item types instead of larger ones (cf. Table 9.2).

main problem class no.	dimensions of large object (*L*,*W*)	no. of item types *n*	size of item types	waste [%]	computing time [sec]
1	(600,600)	5	small	1,472	9,63
2	(600,600)	5	small/large	5,042	7,48
3	(600,600)	5	large	9,117	3,88
4	(600,600)	10	small	0,713	18,93
5	(600,600)	10	small/large	3,661	17,92
6	(600,600)	10	large	7,752	10,59
7	(900,400)	5	small	1,600	9,49
8	(900,400)	5	small/large	4,967	6,65
9	(900,400)	5	large	10,588	3,84
10	(900,400)	10	small	0,974	19,85
11	(900,400)	10	small/large	4,045	14,33
12	(900,400)	10	large	7,732	9,58
13	(1200,300)	5	small	1,811	8,17
14	(1200,300)	5	small/large	7,636	4,74
15	(1200,300)	5	large	10,839	2,82
16	(1200,300)	10	small	0,991	18,17
17	(1200,300)	10	small/large	4,219	11,61
18	(1200,300)	10	large	7,442	6,84

Table 8: *Waste and computing times (averages per instance) for the basic problem classes*

(9.1)

no. of item types *n*	waste [%]	computing time [sec]	no. of zero-waste instances
5	5,897	6,30	2
10	4,170	14,20	19

(9.2)

size of item types	waste [%]	computing time [sec]	no. of zero-waste instances
small	1,260	14,04	18
small/large	4,928	10,46	3
large	8,912	6,26	0

Table 9: *Waste and computing times (averages per instance) for different numbers of item types (9.1) and different sizes of item types (9.2)*

We note that – in contrast to the results reported so far – the dimensions of the large object appear to have only very influence. Waste increases slightly as the shape of the large object becomes less quadratic, computing times are almost constant (cf. Table 10).

dimensions of large object (*L*,*W*)	**waste [%]**	**computing time [sec]**
(600,600)	4,626	10,25
(900,400)	4,984	11,41
(1200,300)	5,490	10,62

Table 10: *Waste and computing times (averages per instance) for the different dimensions of the large objects*

The picture looks slightly different if the data (concerning the waste) is further differentiated (see Table 11). The amount of waste is affected by the problem size and the size of the item types in the first place, but it also tends to grow slightly with a more rectangular, less quadratic shape of the large object. It has to be mentioned, however, that the outliers represented by the problem classes “number of item types: 5; dimensions of large object: (1,200, 300)” and “size of item types: small/large; dimensions of large object: (1,200, 300)” are due to a few single instances only which yield an extraordinarily large amount of waste.

(11.1)

number of item types *n*	**dimensions of large object (*L*,*W*)**		
	(600,600)	(900,400)	(1200,300)
5	5,210	5,718	6,762
10	4,042	4,250	4,218

(11.2)

size of item types	**dimensions of large object (*L*,*W*)**		
	(600,600)	(900,400)	(1200,300)
small	1,092	1,287	1,401
small/large	4,352	4,506	5,928
large	8,434	9,160	9,140

Table 11: *Percentage of waste (averages per instance) dependent on the dimensions of the large object and the number of item types (11.1) and the size of the item types (11.2)*

Table 12 and Table 13 investigate the influence of the defect on computing times and solution quality. Table 12 shows the detailed results for the nine different defect types, whereas Table 13 (in which the defects are sorted in the same sequence as in

Figure 8) gives an aggregated view regarding the sizes (Table 13.1)) and shapes (Table 13.2)) of the defect.

defect	**waste [%]**	**computing time [sec]**
small quadratic	3,929	10,72
small horizontal	4,368	10,34
small vertical	3,704	10,75
medium quadratic	4,994	10,57
medium horizontal	6,016	10,38
medium vertical	4,578	10,27
large quadratic	5,879	10,23
large horizontal	7,211	9,65
large vertical	4,622	9,35

Table 12: *Waste and computing times (averages per instance) for the different dimensions of the defect*

The computing times are hardly affected by the different sizes and shapes of the defect, even though they tend to be slightly smaller for large defects because fewer possible cutting patterns have to be investigated. As could have been expected, a small defect yields less waste than a large defect (cf. Table 13.1) since it allows for a larger number of cutting patterns. The small amount of waste related to a vertical defect (cf. Table 13.2) is due to the fact that all large objects which are considered here are either quadratic or of horizontal shape. This implies – except for the quadratic large object – that a vertical defect virtually divides the large object into two defect-free new plates (see Figure 9), for which a large number of feasible combinations of small items (i.e. cutting patterns) exists. A horizontal defect, on the other hand, leaves narrow defect-free regions above and below the defect and only narrow defect-free regions on the sides of the defect, for all of which the number of feasible cutting patterns is significantly smaller. Both effects compensate each other on a quadratic large object.

(13.1)

size of defect	**waste [%]**	**computing time [sec]**
small	4,000	10,60
medium	5,196	10,41
large	5,904	9,74

(13.2)

shape of defect	**waste [%]**	**computing time [sec]**
quadratic	4,934	10,51
horizontal	5,865	10,13
vertical	4,301	10,12

Table 13: *Waste and computing times (averages per instance) for the different sizes (13.1) and the different shapes (13.2) of the defect*

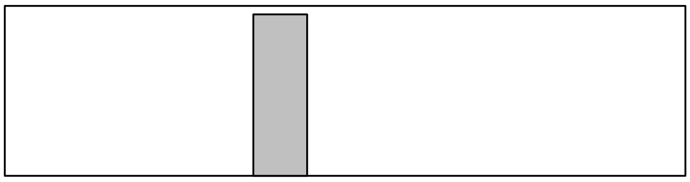


Figure 9: *Large object of size (1200,300) combined with defect of size (95,285)*

Table 14 differentiates the results concerning the waste further with respect to the size (Table 14.1) and the shape (Table 14.2) of the defect on one hand, and with respect to the dimensions of the large object on the other. The amount of waste is affected by the size and the shape of the defect in the first place, but varies only slightly the more rectangular, less quadratic the large object gets.

(14.1)

size of defect	**dimensions of large object (*L*,*W*)**		
	(600,600)	(900,400)	(1200,300)
small	3,686	3,837	4,477
medium	4,708	5,007	5,873
large	5,484	6,108	6,119

(14.2)

shape of defect	**dimensions of large object (*L*,*W*)**		
	(600,600)	(900,400)	(1200,300)
quadratic	4,369	4,930	5,502
horizontal	5,345	5,566	6,684
vertical	4,165	4,456	4,283

Table 14: *Percentage of waste (averages per instance) dependent on the dimensions of the large object and the size (14.1) and the shape (14.2) of the defect*

(15.1)

size of defect	**number of item types *n***	
	5	10
small	4,987	3,214
medium	6,106	4,232
large	6,597	5,064

(15.2)

shape of defect	**number of item types *n***	
	5	10
quadratic	5,801	4,053
horizontal	6,441	5,026
vertical	5,449	3,431

Table 15: *Percentage of waste (averages per instance) dependent on the number of item types and the size (15.1) and the shape (15.2) of the defect*

In Table 15, the percentage of waste for different problem sizes (number of item types) is depicted against the size (Table 15.1) and the shape (Table 15.2) of the defect. The waste increases with an increasing size of the defect as well as with a decreasing number of item types. It can again be observed that the percentage of waste is highest for a horizontal defect and lowest for a vertical defect.

Finally, in Table 16 the percentage of waste for different sizes of the item types is demonstrated against the size (Table 16.1) and the shape (Table 16.2) of the defect. Similar to what has been said about Table 11, it can be observed here that the solution quality differs significantly with the size of the item types. Thus, it becomes evident that – with respect to the solution quality – the size of the item types is the most important parameter.

(16.1)

size of defect	**size of item types**		
	small	small/large	large
small	0,962	3,881	7,459
medium	1,272	5,076	9,159
large	1,546	5,829	10,117

(16.2)

shape of defect	**size of item types**		
	small	small/large	large
quadratic	1,256	4,883	8,642
horizontal	1,366	5,788	10,046
vertical	1,158	4,114	8,047

Table 16: *Percentage of waste (averages per instance) dependent on the size of item types and the size (16.1) and the shape (16.2) of the defect*

7 Conclusions and Outlook

In this paper, an exact solution approach and some heuristic modifications have been presented for the solution of the two-dimensional, guillotineable-layout Single Large Object Placement Problem, in which the large object contains a single defect. It could be shown that the proposed (heuristically modified) method provides solutions of excellent quality in reasonable computing time. Future work will concentrate on extending the proposed method to problems with more than one defect; similar to that, the approach could be extended to non-rectangular defects (represented by more than one rectangle).

References

Aboudi, R.; Barcia, P. (1998):
Determining Cutting Stock Patterns when Defects are Present. *Annals of Operations Research* **82**, 343-354.

Arenales, M.; Morabito, R. (1995):
And AND/OR-Graph Approach to the Solution of Two-Dimensional Non-Guillotine Cutting Problems. *European Journal of Operational Research* **84**, 599-617.

Beasley, J.E. (1985a):
Algorithms for Unconstrained Two-Dimensional Guillotine Cutting. *Journal of the Operational Research Society* **36**, 297-306.

Beasley, J.E. (1985b):
An Exact Two-Dimensional Non-Guillotine Cutting Tree Search Procedure. *Operations Research* **33**, 49-64.

Beasley, J.E. (1985c):
Bounds for Two-Dimensional Cutting. *Journal of the Operational Research Society* **36**, 71-74.

Beasley, J.E. (2004):
A Population Heuristic for Constrained Two-Dimensional Non-Guillotine Cutting. *European Journal of Operational Research* **156**, 601-627.

Carnieri, C.; Mendoza, G.A.; Luppold, W.G. (1993):
Optimal Cutting of Dimension Parts from Lumber with a Defect: a Heuristic Solution Procedure. *Forest Products Journal* **43** (9), 66-72.

Christofides, N.; Whitlock, C. (1977):
An Algorithm for Two-Dimensional Cutting Problems. *Operations Research* **25**, 30-44.

Christofides, N.; Hadjiconstantinou, E. (1995):
An Exact Algorithm for Orthogonal 2-D Cutting Problems Using Guillotine Cuts. *European Journal of Operational Research* **83**, 21-38.

Faaland, B.; Briggs, D. (1984):
Log Bucking and Lumber Manufacturing Using Dynamic Programming. *Management Science* **30**, 245-257.

Fayard, D.; Hifi, M.; Zissimopoulos, V. (1998):
An Efficient Approach for Large-Scale Two-Dimensional Guillotine Cutting Stock Problems. *Journal of the Operational Research Society* **49**, 1270-1277.

G, Y.-G.; Kang, M.-K. (2002):
A New Upper Bound for Unconstrained Two-Dimensional Cutting and Packing. *Journal of the Operational Research Society* **53**, 587-591.

Gilmore, P.C.; Gomory, R.E. (1961):
A Linear Programming Approach to the Cutting-Stock Problem. *Operations Research* **9**, 849-859.

Gilmore, P.C.; Gomory, R.E. (1963):
A Linear Programming Approach to the Cutting-Stock Problem – Part II. *Operations Research* **11**, 863-888.

Gilmore, P.C.; Gomory, R.E. (1965):
Multistage Cutting Stock Problems of Two and More Dimensions. *Operations Research* **13**, 94-120.

Gilmore, P.C.; Gomory, R.E. (1966):
The Theory and Computation of Knapsack Functions. *Operations Research* **14**, 1045-1074.

Hahn, S.G. (1968):
On the Optimal Cutting of Defective Sheets. *Operations Research* **16**, 1100-1114.

Herz, J.C. (1972):
Recursive Computational Procedure for Two-Dimensional Stock Cutting. *IBM Journal of Research and Development* **16**, 462-469.

Karp, R.M. (1972):
Reducibility Among Combinatorial Problems. In: Miller, R.E.; Thatcher, J.W. (eds): Complexity of Computer Computations. New York: Plenum, 85-103.

Massoud, S.B.; Chu, C.; Espinouse, M.-L. (2008):
Characterization and Modelling of Guillotine Constraints. *European Journal of Operational Research* **191**, 112-126.

Morabito, R.N.; Arenales, M.N.; Arcaro, V.F. (1992):
An And-Or-Graph Approach for Two-Dimensional Cutting Problems. *European Journal of Operational Research* **58**, 263-271.

Morabito, R.; Arenales, M.N. (1996):
Staged and Constrained Two-Dimensional Guillotine Cutting Problems: An AND-OR-Graph Approach. *European Journal of Operational Research* **94**, 548-560.

Neidlein, V.; Wäscher, G. (2008):
SLOPPGEN: A Problem Generator for the Two-Dimensional Rectangular Single Large Object Placement Problem. Working Paper No. 15/2008, Faculty of Economics and Management, Otto-von-Guericke-University Magdeburg.

Özdamar, L. (2000):
The Cutting-Wrapping Problem in the Textile Industry: Optimal Overlap of Fabric Lengths and Defects for Maximizing Return Based on Quality. *International Journal of Production Research* **38**, 1287-1309.

Parada, V.; Pradenas, L.; Solar, M.; Palma, R. (2002):
A Hybrid Algorithm for the Non-Guillotine Cutting Problem. *Annals of Operations Research* **117**, 151-163.

Sarker, B.R. (1988):
An Optimum Solution for One-Dimensional Slitting Problems: A Dynamic Programming Approach. *Journal of the Operational Research Society* **39**, 749-755.

Scheithauer, G.; Terno, J. (1988):
Guillotine Cutting of Defective Boards. *Optimization* **19**, 111-121.

Twisselmann, U. (1999):
Cutting Rectangles Avoiding Rectangular Defects. *Applied Mathematics Letters* **12**, 135-138.

Vianna, A.C.G.; Arenales, M.N. (2006):
O Problema de Corte de Placas Defeituosas. *Pesquisa Operacional* **26**, 185-202.

Wang, P.Y. (1983):
Two Algorithms for Constrained Two-Dimensional Cutting Stock Problems. *Operations Research* **31**, 573-586.

Wäscher, G.; Haußner, H.; Schumann, H. (2007):
An Improved Typology of Cutting and Packing Problems. *European Journal of Operational Research* **183**, 1109-1130.