ELSEVIER

European Journal of Operational Research 84 (1995) 618–632

EUROPEAN JOURNAL OF OPERATIONAL RESEARCH

Theory and Methodology

An approximation algorithm for solving unconstrained two-dimensional knapsack problems

Didier Fayard, Vassilis Zissimopoulos *

LRI, Université de Paris XI, Centre d'Orsay, 91405 Orsay, France

Received August 1992; revised November 1992

Abstract

An efficient heuristic for solving two-dimensional knapsack problems is proposed. The algorithm selects an *optimal subset of optimal generated strips* by solving a sequence of *one-dimensional knapsack problems*. We show that the number of these knapsacks can be reduced to only four knapsacks. The algorithm gives an excellent worst-case experimental approximation ratio (0.98), and a high percentage of optimal solutions (91%). From this heuristic, we derive an approximation algorithm for which we prove some refined bounds and we show that its approximation ratio is $\frac{4}{9}$. Our numerical study on large size instances shows the efficiency of these algorithms for solving real-world problems which are hardly handled by other known methods, which are often limited by computer storage facilities.

Keywords: Two-dimensional knapsack; Two-space knapsack; Cutting stock problems; Knapsack; Approximation algorithms

1. Introduction

Given quantities of goods of different shapes to be cut from a material that comes in various sizes, such as lengths and heights, a number of possible efficient cutting patterns are considered. How much of each pattern should be cut? The numbers of each pattern cut represent the decision variables. The constraints are given by the required quantities and by the amount of available material of each size. The objective may be to minimize the cost of the used material or the amount of produced waste. Such kind of problems, known as *Trim problems* [16,14] or *Cutting Stock problems* [1,3,6,11,20,21] are very often approached by *Generalized Linear Programming methods* or by tailored heuristics. Whatever the approach, one has to confront the problem of cutting a stock unit with respect to the quantities requirements of goods and the optimizing objective. In a linear programming approach this problem appears when we generate the column vector which enters the basis each time we look for improving the

* Corresponding author.

0377-2217/95/$09.50 © 1995 Elsevier Science B.V. All rights reserved
SSDI 0377-2217(93)E0221-I

current basic solution [12,10,13]. The cutting stock problem in a linear programming formulation is defined as follows:

$$\text{Min} \sum_{j=1}^{q} c_j x_j$$

$$\text{subject to} \quad \sum_{j=1}^{q} a_{ij} x_j \geq b_i, \quad x_j \geq 0, \quad i = 1, \ldots, m,$$

where b_i denotes the quantities requirements of goods, q is the number of feasible cutting patterns, x_j is the number of times the j-th cutting pattern is used, a_{ij} is the number of good of type i cut in the pattern j and c_j is the cost of stock item j. The usually huge number of cutting patterns q constitutes a major difficulty in solving this problem. This difficulty is circumvented by employing a column generation scheme, where each column to enter the basis is generated by solving an auxiliary problem. In particular, in order to produce the next column, we have to find that column whose scalar product with the current linear programming dual evaluators is maximum. Since any set of nonnegative integers a_i, $i = 1, \ldots, m$, is a column (or cutting pattern) provided that it presents a feasible cutting pattern, a column is generated by solving a generalized knapsack problem [11].

Next, we deal with this problem which is NP-complete and we consider the case where goods and material to be cut are rectangular forms and we refer to it as the *Two-space Knapsack Problem* [16], or as the *Two-dimensional Knapsack Problem* (TDK) [11].

Cutting problems have been studied first by Kantorovich [17] and later by Gilmore and Gomory [11,12] for industrial or commercial uses (textile, paper, sheet metal, or plastic industries). Later these problems have been studied in multiprogrammed computer systems by Codd [5] and in multiprocessor systems, by Garey and Graham [9]. Since these problems occur in a wide variety of domains there is a large economic incentive to find more efficient solutions.

In this paper we propose a new efficient heuristic for solving the TDK problem. The algorithm uses a serie of One-Dimensional Knapsack (ODK) problems for generating a set of optimal strips and then fills them in the plate in an optimal way using still another ODK [2,7,8,19]. For reducing the number of involved ODK problems we use dynamic programming methods and we show that only four knapsacks are sufficient in our algorithm. Certainly, for solving exactly these problems by dynamic programming the complexity is $O(m \cdot \max(L, H))$, where m is the number of pieces to cut and L, H the length and the height respectively of the plate to cut. But, the values m, L and H, even for large size instances of the TDK, are small in practice for the ODK and thus the dynamic programming methods turn out to be quite efficient.

We show that the algorithm is very efficient in practice by giving a large number of experimental results on randomly generated problem instances.

Another particular feature of the algorithm is that it deals with different versions of the TDK problem. In fact, it applies to versions with *bounded variables* [4,23] or *unbounded variables* [11,15], to versions where the *profit associated to each piece to cut is proportional to its surface* [15,23] as well as to versions with *surface-independent profit* [4,11,24,25]. When we deal with *bounded variables* each piece to be cut can appear in the cutting pattern no more than a fixed number of times. Whereas, when *unbounded variables* are assumed, each piece can appear any number of times in the pattern. If the profit associated to each piece is proportional to its surface the objective is to minimize the waste, otherwise the objective is to maximize a profit function associated with each cutting pattern.

In a natural way we derive from this algorithm an approximation algorithm which still uses the ODK problem. In this new algorithm the ODK problems are solved by a polynomial algorithm within $1 - \varepsilon$, for every real number $\varepsilon > 0$.

We study several performance bounds for the approximation algorithm and we prove that in the case of unbounded variables and profit proportional to the pieces surface the approximation ratio (fraction obtained solution/optimal solution) is $\frac{4}{9}$, which is the first known approximation result for this problem.

The paper is organized as follows: In the next section we define the problem and its different versions. In Section 3 we describe the procedures that generate the optimal strips and a procedure which fills these strips in a plate. We show how to reduce the number of ODK problems and we present the main steps of our algorithm. In Section 4 we discuss the complexity of the algorithm. In Section 5 we present the derived approximation algorithm. Next, we study performance bounds for the approximation algorithm and finally, we report on several numerical experiments on large size instances which are solved efficiently by the algorithms with solutions very close to the optimal ones.

2. The Unconstrained Two-Dimensional Knapsack Problem

An instance of the Unconstrained Two-Dimensional Knapsack Problem (UTDK) consists of a finite set of rectangular pieces of given dimensions: $S = \{(l_1, h_1), (l_2, h_2), \ldots, (l_m, h_m)\}$ and a rectangular initial plate of length L and height H. With each piece is associated a profit π_i. The objective is to find a cutting pattern of the plate without limits on the number of repetitions of each type of piece and *maximizing* the profit function:

$$F(L, H) = \sum_{i=1}^{m} a_i \pi_i$$

where a_i (to be determined) denotes the number of times that the piece i appears in the pattern and m is the number of pieces in S. If the number of appearances of each type of piece is not allowed to exceed a fixed number in the cutting pattern then the problem is known as the *Constrained Two-Space Knapsack problem* or *Constrained Two-Dimensional Cutting problem*. Furthermore, for each type of the problem, two versions are considered depending on the values π_i attributed to each piece. If π_i is the area of the i-th piece the problem consists in *minimizing* the total waste. If the values π_i are attributed following criteria which take into account other characteristics than pieces surface the problem consists in *maximizing* the total profit given by the cutting pattern. For example the value may reflect, the desirability that the next cutting pattern in a sequence should include pieces of a size appeared in the previous cutting pattern. A full cutting, i.e. without any wastage, is not necessarily in this case an optimal cutting pattern.

Next, we make two assumptions which reflect frequent real-world situations. The cutting tool is of *guillotine type* (each cut is made orthogonally from one edge to the parallel one of the plate or of some sub-plate produced by previous cuts) and all pieces to cut are of *fixed orientation* ($(l_i, h_i) \neq (h_i, l_i)$).

3. The algorithm

The algorithm we propose, called the Best Strips Cutting Algorithm (BSC), is derived from the analysis of the structure of the optimal cutting patterns. In general, these patterns are composed by some *vertical and (or) horizontal strips* cut either in the initial plate or in a few sub-plates produced by previous cuts made on the initial plate. Furthermore, an optimal pattern is very often equivalent to another one which has a such structure. Fig. 1 illustrates two optimal cutting patterns for a set of three pieces, $p_1 = (3, 3)$, $p_2 = (4, 3)$, $p_3 = (2, 4)$ with $\pi_i = l_i h_i$, cut into a plate of dimensions $(L, H) = (9, 11)$. One observes a strips composition structure of the second cutting pattern. It can be also seen that the equivalent optimal pattern (a) is composed by two sub-plates (3, 11) and (6, 11) each one having a strips

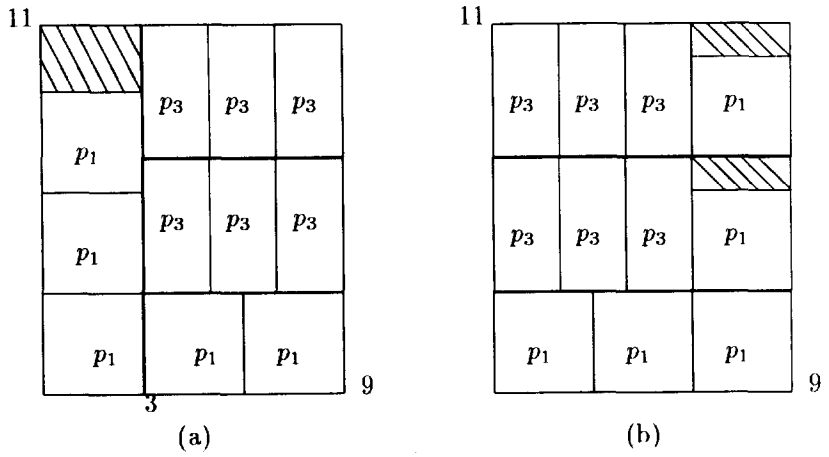


Fig. 1. (a) An optimal cutting pattern of the oriented pieces p_i. (b) An equivalent strips structure cutting pattern. The optimal cutting pattern in (a) is composed by two sub-plates (3, 11) and (6, 11), each one optimal dissected and having a strips structure. Dimensions for pieces $p_i = (l_i, h_i)$, $1 \le i \le 3$: 3×3, 4×3, 2×4. Values for p_i: 9, 12, 8.

structure. The first sub-plate includes a vertical strip whereas the second one three horizontal strips. Intuitively, one expects that a method which creates optimal strips and fill them into the initial plate or into some sub-plates, should be able to produce satisfactory solutions.

Next we discuss what kind of sub-plates to consider, how we can create a set of optimal strips and then, how we can efficiently fill strips in a given sub-plate.

3.1. Dealing with a finite number of sub-plates

In an attempt to fill efficiently the initial plate by strips, we cut it in a finite number of couples of sub-plates. Then, each sub-plate is filled by horizontal or vertical sub-strips. In this way, the obtained structures including combinations of horizontal-vertical sub-strips give efficient solutions for the initial plate. However, the division of the plate into sub-plates should be made only on abscissas which enable our algorithm to provide better solutions. We consider two sets of abscissas. One for the horizontal size denoted by P_L, and another one for the vertical size of the initial plate, denoted by P_H. These sets, slightly modified, have been used also by Christofides and Whitlock [4] in order to define *normalized* guillotine patterns.

The precise definition of the sets P_L and P_H is as follows:

$$P_L = \left\{ x \mid x = \sum_{i=1}^{m} z_i l_i \le \tfrac{1}{2}L,\ z_i \ge 0,\ z_i \text{ integer} \right\},$$

$$P_H = \left\{ x \mid x = \sum_{i=1}^{m} z_i h_i \le \tfrac{1}{2}H,\ z_i \ge 0,\ z_i \text{ integer} \right\}.$$

The set P_L (and similarly P_H) can be generated as follows: Consider the function

$$g_i(x) = \begin{cases} 0 & \text{if } i = 0 \wedge x = 0, \\ \infty & \text{if } i = 0 \wedge x \ne 0, \\ g_{i-1}(x) & \text{if } x < l_i, \quad i \in I, \\ \min\{g_{i-1}(x), \max\{1, \min_k\{g_{i-1}(x - kl_i),\ 1 \le k \le \lfloor x/l_i \rfloor, k \in N\}\}\}, & i \in I, \quad x \in X, \end{cases}$$

where $I = \{1, 2, \ldots, m\}$ and $X = \{0, 1, 2, \ldots, \lfloor \frac{1}{2}L \rfloor\}$. Hence, a point x is in P_L if $g_m(x) = 1$ and it does not if $g_m(x) = \infty$.

We show later that we can take into account only the elements of these sets for plate's dissection, since only these elements can improve our solution.

3.2. Strips generation

We propose now a procedure which creates either horizontal or vertical (sub-)strips which are optimal with respect to a given length and height. We call these strips *general type of strips*. Let α be the length and β the height of a (sub-)plate.

a) *Horizontal strips.* Let $S = \{(l_1, h_1), (l_2, h_2), \ldots, (l_m, h_m)\}$ be the set of rectangular pieces to cut. We reorder the elements of the set S in an order of non-decreasing heights such that $h_1 \leq h_2 \leq \cdots \leq h_m$. Let r denote the number of distinct heights. Let k_i denote the greatest index of a piece of height h_i, $i = 1, \ldots, r$, and

$$S_{k_i} = \{l_j : (l_j, h_j) \in S,\ h_j \leq h_{k_i}\}.$$

Obviously, we have

$$S_{k_1} \subset S_{k_2} \subset \cdots \subset S_{k_r}.$$

For each S_{k_i} we associate an One-Dimensional Knapsack Problem defined as follows:

(ODK(i))

$$\text{Max} \sum_{l_\mu \in S_{k_i}} \pi_\mu x_\mu$$

$$\text{subject to} \sum_{l_\mu \in S_{k_i}} l_\mu x_\mu \leq \alpha, \quad x_\mu \geq 0, \quad x_\mu \text{ integer},$$

where α is the length of the plate to be cut, π_μ is the profit associated to the piece (l_μ, h_μ) (here the surface) and x_μ is the number of times the piece (l_μ, h_μ) appears into the i-th strip.

By solving the ODK(i), $i = 1, 2, \ldots, r$, we create r horizontal strips with length α, height h_{k_i} and value $F_i(\alpha)$ (here the occupied surface inside the rectangle (α, h_{k_i})). In the sequel, for purposes of simplicity we replace h_{k_i} by β_i.

This procedure is easily generalized to create also *uniform* strips consisting each one of pieces with the same height.

Also it is able to construct *almost uniform* strips consisting of all type of pieces with heights h_k in the interval $(h_i - \delta, h_i + \delta)$ where δ a 'convenient' real parameter. Notice that the *proposed general strips are better than the uniform or almost uniform strips*, since within a given strip's height the value $F_i(\alpha)$ of the corresponding knapsack is always at least equal to the value of the knapsack corresponding to a uniform or almost uniform strip.

b) *Vertical strips.* By the same process we can construct vertical strips. We simply reverse the roles of the lengths and the heights in the previous procedure. The pieces of the set S are reorganized in non-decreasing lengths. We denote by v the number of different lengths and the sub-sets S_{k_i}, $i = 1, \ldots, v$, include the heights of the pieces with lengths bounded by l_{k_i} (next replaced by α_i). By solving v knapsack problems we create a set of optimal vertical strips, which have values $F_i(\beta)$ and lengths α_i.

3.3. Filling the plate

We have shown that we can generate a number of optimal horizontal or vertical strips. Each such strip is characterized by its profit value $F_i(\alpha)$ and its height β_i (for horizontal strips), and $F_i(\beta)$ and its length α_i (for vertical strips). We shall fill the best of them in the plate (α, β) with respect to the height (respectively to the length for vertical strips) of the disposed plate, by solving another knapsack problem. For the horizontal strips this knapsack is defined as follows:

(ODK$(r+1)$)

$$F_{r+1}(\beta) = \text{Max} \sum_{i=1}^{r} F_i(\alpha)\, y_i$$

$$\text{subject to} \quad \sum_{i=1}^{r} \beta_i y_i \le \beta, \quad y_i \ge 0, \quad y_i \text{ integer},$$

where β is the height of the plate to be cut and y_i, $i = 1, \ldots, r$, denotes the number of repetitions of the i-th strip. For filling the plate with vertical strips just define the ODK$(v+1)$ problems as previously and replace α by β, β by α, β_i by α_i and r by v. Therefore, two cutting patterns are generated having values $V^{\text{hor}} = F_{r+1}(\beta)$ and $V^{\text{ver}} = F_{v+1}(\alpha)$.

We have shown how to fill a (sub-)plate with horizontal or vertical strips. The procedure we use and this one which creates the strips require to solve $r + v + 2$ $(r, v < m)$ knapsack problems when all pieces of the set S enter to the (sub-)plate. However, using dynamic programming methods [22], for solving the involved knapsack problems this number can be reduced dramatically.

In fact, let (α, β) be the initial plate (L, H). When we solve the largest knapsack r, we create the highest horizontal strip. Then, the solutions of the other knapsacks $1, 2, \ldots, r-1$ are available and thus, all the strips of intermediate height. Clearly, one only knapsack provides all the horizontal strips. Next, all values required by the filling procedure for formulating the $(r+1)$-th knapsack which selects strips for filling in the plate are available. Similarly, the v-th knapsack creates the longest vertical strip, and thus all strips of intermediate length. Finally, the $r + v + 2$ *knapsacks required to fill the plate by horizontal or vertical strips are reduced to four.*

Furthermore, all the optimal sub-strips for the sub-plates generated after dissections on the points of the sets P_L and P_H are also available. The only requirement when we deal with each sub-plate is to solve two knapsacks which select the best sub-strips for filling the sub-plate.

The main steps of the algorithm are:

Step 1. *Discretize* the length $\frac{1}{2}L$ and the height $\frac{1}{2}H$ of the initial plate into two sets P_L and P_H respectively, where P_L denotes the set of linear combinations with non negative integer coefficients of the lengths and P_H denotes the set of linear combinations with non negative integer coefficients of heights of the pieces to cut.

Step 2. *Generate* the highest horizontal strip and the longest vertical strip.

Step 3. At each element $x_j \in P_L \cup P_H$ make a *vertical guillotine cut* if $x_j \in P_L$ or a *horizontal guillotine cut* if $x_j \in P_H$. Let $P_1 = (x_j, H)$ and $P_2 = (L - x_j, H)$ denote the produced sub-plates, when $x_j \in P_L$ and $P_1 = (L, x_j)$ and $P_2 = (L, H - x_j)$ when $x_j \in P_H$.

Step 3.1. *Fill* horizontal strips in the sub-plates P_i, $i = 1, 2$. Let V_i^{hor} denote the two corresponding values of the generated patterns.

Step 3.2. *Repeat the step 3.1* for filling vertical strips in the sub-plates. Let V_i^{ver}, $i = 1, 2$ denote the values of the new patterns.

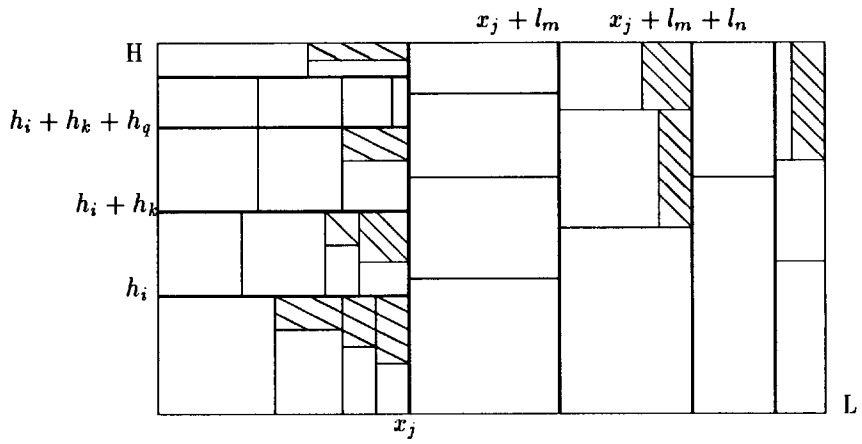


Fig. 2. A cutting pattern generated by the BSC algorithm. Horizontal strips of heights h_i, h_k, h_q,... are filled into the left sub-plate and vertical strips of lengths l_m, l_n,... are filled into the right sub-plate.

Step 3.3. Record for each *sub-plate* P_i, $i = 1, 2$ the better pattern. Let $V_i = \max\{V_i^{\text{hor}}, V_i^{\text{ver}}\}$ denote the value of the chosen pattern.

Step 3.4. Record the produced cutting pattern of the *plate* and let $V^{x_j} = V_1 + V_2$ denote its value.

Step 4. Choose the cutting pattern having the maximum value V^{x_j} for all $x_j \in P_L \cup P_H$.

Fig. 2 depicts a possible cutting pattern generated by the algorithm.

3.4. The 0-cut phase

When dealing with the zero element of the set P_L, the second sub-plate is exactly the initial plate (L, H). In the sequel we shall refer to this case as the *0-cut phase* of the algorithm. This phase includes the Steps 2 and 3 in the algorithm and only for $x_j = 0$. It provides the solution $V^0 = \max\{V_2^{\text{hor}}, V_2^{\text{ver}}\}$ which is obtained by solving only four knapsacks.

4. On the complexity of algorithm BSC

By considering only the elements of the sets P_L and P_H for making dissections on the initial plate, obviously the computational effort is significantly reduced. Moreover, solutions improvement is not possible from other dissections which are not included in the sets P_L and P_H. In fact, it is easy to prove the following result: *There is a solution* V^x *for some* $x \in P_L$ *or some* $x \in P_H$ *which is not worse than any solution* V^p *with* $p \notin P_L$ *and* $p \notin P_H$.

As we have already mentioned, the *0-cut phase* of the algorithm requires only four knapsacks. Additionally, for each generated sub-plate we require to solve two knapsacks. Thereby, the BSC algorithm executed over all elements of the sets P_L and P_H requires for each element four knapsacks. The total required number of knapsacks for the BSC algorithm being $4(|P_L \cup P_H|)$.

Among these knapsacks the larger ones are those solved at Step 2. The first one with size m (number of pieces to cut), and second member L, and the second one with sizes m and second member H. To solve, however, these problems exactly using dynamic programming, the complexity is $O(m \cdot \max(L, H))$

which is not polynomial in the size of the problem. But, in our case large size instances of the TDK in practice have values m, L and H which are rather small for the one-dimensional knapsack and the dynamic programming algorithms turn out to be quite efficient.

We point out that TDK problems of length and height equal to 150 and with 50 pieces to cut can be considered as medium or even large size instances. But, the generated one-dimensional knapsack problems in our algorithm having sizes 50 and second member 150 are small size instances. Therefore, the algorithm is able to deal efficiently with instances which are hardly handled by other known methods [11,10,15].

5. The derived approximation algorithm

In this section we present a polynomial approximation algorithm which is naturally derived from the *0-cut phase* of the BSC algorithm. This algorithm is shortly described in Fig. 3 by putting $\alpha = L$ and $\beta = H$.

We assume here that the involved ODK problems are all separately solved by a polynomial approximation algorithm. Obviously, the algorithm needs to solve $r < m$ knapsacks with maximum size m (number of pieces to cut) and second member the size of the initial plate L. Also it needs another knapsack with size r (number of different heights) and second member the height of the initial plate H. Moreover, it needs $v < m$ knapsacks with maximum size m and second member H and one knapsack with size v (number of different lengths) and second member L. Thereby, the algorithm needs to solve $r + v + 2$ knapsacks. Of course, the ODK problem is NP-complete. However, it can be solved approximately using a polynomial approximation algorithm [18].

6. Performance bounds for the approximation algorithm

In this section we study performance bounds for the approximation algorithm which deals with *minimizing waste* UTDK problem. In this case the value π_i attributed to each piece is the surface $l_i h_i$ and thereby maximizing the profit is equivalent to minimizing the waste.

Lemma 1. *The waste in each strip generated by the strips generation procedure is less than* $(1/(k+1))S_i$, *where* S_i *is the surface of the strip,* $i = 1, \ldots, r$ *and* $k = \lfloor L/l \rfloor$ *with* l *the length of the tallest piece that can be fit in the strip for the horizontal strips and* $k = \lfloor H/h \rfloor$ *with* h *the height of the longest piece that can be fit in the strip for the vertical strips.*

```
procedure Filling-plate (α, β, l_i, h_i, π_i)
    procedure knapsack (weights, values)
        begin
            ⟨strips generation⟩
            knapsack (weights, values)
        end
    begin
        sort(h_i)
        knapsack (weights, values)
        ⟨pattern generation⟩
        Filling-plate (β, α, h_i, l_i, π_i)
    end
```

Fig. 3. An approximation algorithm which provides cutting patterns with a horizontal strips or vertical strips structure.

Proof. We have $L/(k+1) < l \le \lfloor L/k \rfloor$ and thus, $k(L/(k+1)) < kl \le k\lfloor L/k \rfloor \le L$. The strip of height h_i has a value (or occupied area) V_i which satisfies: $V_i \ge klh_i$ since there is a construction which includes at least k pieces of length l. Thus, $V_i > k(L/(k+1))h_i = (k/(k+1))S_i$. For the vertical strips the proof is similar. □

Immediately from the lemma, we conclude that each created strip has a value (occupied area by the included pieces) greater than the half of its area. As a consequence we have the following corollary.

Corollary 1. *The approximation ratio of the algorithm is asymptotically equal to* $\frac{1}{2}$.

Proof. By the previous lemma we have for each strip that $V_i > \frac{1}{2}S_i$. Moreover, we have $V = \sum_{i \in E} V_i$, where E is the set of indices of the selected strips by the last knapsack. Thereby, $V > \sum_{i \in E} \frac{1}{2}S_i = \frac{1}{2}L\sum_{i \in E} h_i$. Since $H - \sum_{i \in E} h_i < h^*$, where h^* is the minimum height of the pieces, we obtain $V > \frac{1}{2}L(H - h^*)$. Also we have $V^* \le LH$. Consequently, $V/V^* > \frac{1}{2}(1 - h^*/H)$, and obviously, this ratio tends to $\frac{1}{2}$ for small values of h^*. The same result can be obtained of course for small lengths. □

Lemma 2. (i) *If* $l_i > \lfloor \frac{1}{2}L \rfloor$ *for all* $i = 1, \ldots, m$, *then the optimal solution is equivalent to a solution composed by a vertical strip.*

(ii) *If* $h_i > \lfloor \frac{1}{2}H \rfloor$ *for all* $i = 1, \ldots, m$, *then the optimal solution is composed by a horizontal strip.*

(iii) *If* $k = \max\{\lfloor L/l_i \rfloor, \lfloor H/h_i \rfloor, i = 1, \ldots, m\}$ *is equal to* 1 *or* 2, *then the optimal solution of the* UTDK *is (or is equivalent to a solution) composed by horizontal or vertical strips.*

(iv) *The approximation algorithm finds these solutions with* $1 - \varepsilon$, *for every* $\varepsilon > 0$.

Proof. (i): Obviously, no more than one piece can enter horizontally in the plate. Fig. 4a illustrates such an instance.

(ii): Similarly, no more than one piece can enter vertically in the plate (Fig. 4b).

(iii): The case of $k = 1$ is trivial. For $k = 2$ there exists an index w such that either

$$\tfrac{1}{3}L < l_w \le \tfrac{1}{2}L, \quad \tfrac{1}{3}L < l_i \le L \quad \forall i \ne w, \text{ and } \tfrac{1}{3}H < h_i \le H \quad \forall i = 1, \ldots, m$$

(at least one piece enters exactly two times in the length of the plate)
or

$$\tfrac{1}{3}H < h_w \le \tfrac{1}{2}H, \quad \tfrac{1}{3}H < h_i \le H \quad \forall i \ne w, \text{ and } \tfrac{1}{3}L < l_i \le L \quad \forall i = 1, \ldots, m$$

(at least one piece enters exactly two times in the height of the plate).

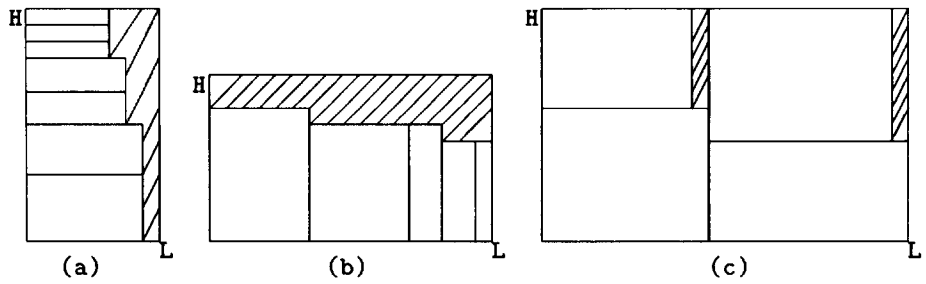


Fig. 4. Three instances for the two-dimensional cutting problem which are solved within $1 - \varepsilon$ by the approximation algorithm: a) a vertical strip structure b) a horizontal strip structure and c) two vertical strips structure.

Assume that $k = 2$ is induced by a length l_w. Then, an optimal cutting pattern includes necessarily a guillotine cut (say vertical cut), that divides the initial plate into two sub-plates which are also optimal solved. But, it is obvious that each one of them does not contain more than one piece in its length. Therefore, the optimal solution for each sub-plate is exactly a vertical strip. Consequently, the optimal solution of the initial plate is composed by two vertical strips (Fig. 4c). If the guillotine cut is a horizontal cut we conclude that the optimal solution is composed either by a horizontal strip or two horizontal strips. The conclusion is similar when $k = 2$ is induced by a height h_w.

(iv): Just solve the ODK problems which create the horizontal strips of length L and the vertical strips of height H by a polynomial algorithm within $1 - \frac{1}{2}\varepsilon$, and the two last ODK problems that select the best of the strips to fill the plate, also within $1 - \frac{1}{2}\varepsilon$. □

Theorem 1. *If V denotes the value of a solution provided by the approximation algorithm and V^* is the value of an optimal solution then*

$$V > \frac{k}{2(k+1)} V^*, \quad \text{where } k = \max\left\{\left\lfloor \frac{L}{l_i} \right\rfloor, \left\lfloor \frac{H}{h_i} \right\rfloor, i = 1, \ldots, m\right\}.$$

Proof. Let w be a piece with $k = \max\{\lfloor L/l_i \rfloor, \lfloor H/h_i \rfloor, i = 1, \ldots, m\}$ and assume that k corresponds to the length l_w. Let us consider the horizontal strip having height h_w. By Lemma 1 the strip's value is such that $V_w > (k/(k+1))Lh_w$. Consequently, the algorithm gives

$$V \geq \left\lfloor \frac{H}{h_w} \right\rfloor V_w > \frac{k}{k+1} L h_w \left\lfloor \frac{H}{h_w} \right\rfloor$$

and since $V^* \leq LH$ we obtain

$$\frac{V}{V^*} > \frac{k}{k+1} \frac{h_w}{H} \left\lfloor \frac{H}{h_w} \right\rfloor \geq \frac{k}{k+1} \frac{\lfloor H/h_w \rfloor}{\lfloor H/h_w \rfloor + 1} \tag{1}$$

and thus, $V > (\frac{1}{2}k/(k+1))V^*$. If k corresponds to the height h_w the proof is similar by considering the vertical strip of length l_w. □

As discussed by Christofides and Whitlock [4], every guillotine pattern has an equivalent *normalized* guillotine form in which all pieces are left-justified at the lowest possible position in the initial plate and the cuts are made on elements belonging in the sets P_L and P_H. Next, we consider optimal solutions having this property. Eventually, the strips discussed below can be trivial strips including only one piece. These strips are indifferrently referred as vertical or horizontal strips.

Theorem 2. *The approximation ratio of the algorithm is $\frac{4}{9}$.*

Proof. By considering a normal optimal cutting pattern we prove that, for $k \geq 3$, either there exists a piece w that can fit twice in the length of the plate and in the height of the plate or the optimal solution is composed by only horizontal or only vertical strips.

Let us firstly consider the case where $k = 3$ and assume that this value is induced by a length l_w. Clearly, we have that

$$\tfrac{1}{4}L < l_w \leq \tfrac{1}{3}L \tag{2}$$

whereas

$$\tfrac{1}{4}L < l_i \leq L \quad \forall i \neq w, \text{ and } \tfrac{1}{4}H < h_i \leq H \quad \forall i = 1, 2, \ldots, m$$

(similarly if $k = 3$ is induced by a height h_w). Let $x_j \le \frac{1}{2}L$, $x_j \in P_L$ be the first guillotine cut, say vertical, in an optimal pattern, that divides the initial plate into two sub-plates each one of them being naturally optimally dissected. The two sub-plates contain in their length only one piece or the former contains one piece and the latter two pieces (or inversely).

In the first case the optimal solution is composed by two vertical strips (or it is equivalent to a such solution). By applying Lemma 2 we conclude that the algorithm finds this solution within $1 - \varepsilon$, for every $\varepsilon > 0$. In the second case we consider the following two sub-cases.

Firstly, suppose that

$$\tfrac{1}{3}H < h_i \le H \quad \forall i = 1, 2, \ldots, m, \tag{3}$$

i.e. there is no piece that can be fit vertically more than two times in the plate. Then, Lemma 2 implies that the optimal solution of the second sub-plate is composed either by two vertical strips or two horizontal strips (these are the more interesting cases). Thereby, the optimal solution of the initial plate includes either one vertical strip (first sub-plate) and two horizontal strips (second sub-plate) or three vertical strips. The last solution composed by three vertical strips is found within $1 - \varepsilon$ by the algorithm. For the other solution the algorithm gives in the worst case approximation ratio $\frac{4}{9}$. In fact, such a solution will have a structure as in Fig. 5a. The optimal pattern includes three consecutive guillotine cuts made at the point x_j vertically, at the point y_j horizontally, and at the point x_p vertically. Since we consider normal guillotine patterns with respect of the effects of symmetry, necessarily we have that

$$x_j \le \left\lfloor \tfrac{1}{2}L \right\rfloor, \qquad y_j \le \left\lfloor \tfrac{1}{2}H \right\rfloor, \qquad x_p \le \left\lfloor \tfrac{1}{2}(L - x_j) \right\rfloor. \tag{4}$$

The strips including pieces w and q can be eventually trivial strips. If $h_p > \frac{1}{2}H$ we have by Lemma 1 and considering the two horizontal strips generated by the pieces p and q that $V_p > \frac{2}{3}Lh_p > \frac{1}{3}LH$. Also we have by using (3) that $V_q > \frac{1}{2}Lh_q > \frac{1}{6}LH$.

Consequently, our solution verifies $V \ge V_p + V_q > \frac{1}{2}LH$ and thus, $V > \frac{1}{2}V^*$.

If $h_p \le \frac{1}{2}H$, then piece p can fit two times in the length and 2 times in the height. Thus, by Lemma 1 and relation (1) we have $V \ge V_p > \frac{4}{9}LH$ and so, $V > \frac{4}{9}V^*$.

Secondly, suppose that there is a piece p such that

$$\tfrac{1}{4}H < h_p \le \tfrac{1}{3}H,$$

i.e. there exists a piece of height h_p that can fit vertically in the plate three times. If $p = w$ (where w is the piece having the smallest length) then the algorithm shall give a solution greater than $\frac{9}{16}$ of the

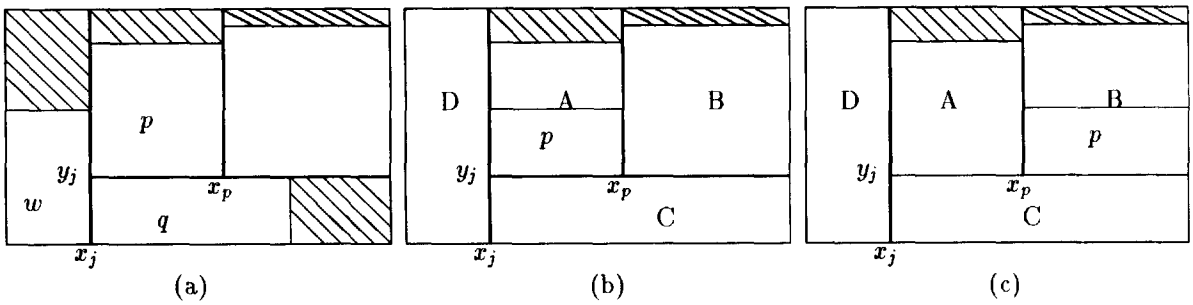


Fig. 5. Three optimal cutting patterns when a piece w having the smallest length l_w can fit into a horizontal strip exactly 3 times. (a) The pattern is obtained by the BSC algorithm. (b) The region A is sufficient to produce two pieces. The pattern can not be obtained by the BSC algorithm. (c) The region B is sufficient to produce two pieces, whereas the region A produces only one piece. The pattern can not be obtained by the BSC algorithm.

optimal solution (using (1)). Otherwise, one of the two situations illustrated by Figs. 5b and 5c will appear. If one of the regions A or B produces two pieces, the induced structure can not be obtained neither by the BSC algorithm. By (2) we conclude that region D is a vertical strip, and the region C is a horizontal strip. Furthermore, we deduce that one of the two pieces produced in region A (Fig. 5b) or B (Fig. 5c) has dimensions l_p and h_p such that $2l_p \leq L$, $2h_p \leq H$. This is obvious for the situation illustrated in Fig. 5b by (4), whereas for the situation in Fig. 5c we have $\frac{1}{4}L < l_w$ and thus, $l_p \leq L - 2l_w \leq \frac{1}{2}L$. By considering the strip of height h_p in Lemma 1 we have $k = 2$. By using the same strip in Theorem 1, relation (1) gives $V \geq \frac{4}{9}V^*$.

Hence, for $k < 4$ the algorithm always gives $V \geq \frac{4}{9}V^*$.

For $k \geq 4$ similarly we conclude that there exists a piece which fits in the plate horizontally and vertically twice or trivially the optimal solution is equivalent to a horizontal (or vertical) strip.

The proof is concluded by applying relation (1) of Theorem 2 for the first case (putting $k = 2$ and $\lfloor H/h_w \rfloor = 2$ or Lemma 2 for the second case. □

Corollary 1. *The approximation ratio for the algorithm when the initial plate and the pieces to cut are squares is* $\frac{9}{16}$.

Proof. Consider k as in Theorem 1. Then, the approximation ratio derived as in Theorem 1 for the algorithm is $k^2/(k+1)^2$. For $k = 1$, 2 the algorithm provides a solution within $1 - \varepsilon$, for every $\varepsilon > 0$. Thus, when we consider the worst-case, $k = 3$, the result is immediately obtained. □

Remark. If we introduce some restrictions on the pieces size then stronger results can be derived. For example, it is easy to prove (as in Theorem 1), that *if there exists a piece* (l_j, h_j) with $H = \rho h_j$ for some integer ρ and $l_j \leq \lfloor L/k \rfloor$ for some integer k then, we obtain the approximation ratio $k/(k+1)$. Thereby, in a practical application where k equals to 100 is not rare, we obtain almost optimal solutions (i.e. $V > \frac{100}{101}V^*$).

7. Computational results

We have shown that the *0-cut phase* from which the approximation algorithm is derived gives an interesting theoretical approximation ratio. In this section, we shall see that gives also an excellent experimental approximation ratio.

The BSC algorithm as well as the *0-cut phase* of the algorithm were tested on up to 100 instances randomly generated. We have considered two sets of instances. The first set included 50 moderate size instances, with sizes L and H ranging between 50 and 100 and 10 to 30 pieces to cut. The second set included also 50 instances but of large size: for example plates with sizes ranged between 100 and 200 and number of pieces to cut ranged between 30 and 60. The dimensions of the initial plates and the number of pieces to cut were chosen uniformly on the fixed interval, whereas the dimensions of the

Table 1
The performance of the BSC algorithm for small and large instance sizes

Optimal sol. (%)	94	88	91
Exp. appr. ratio	0.98	0.99	0.98
Exp. av. appr. ratio	0.999	0.999	0.999
Instance sizes	small	large	total

Table 2
The performance of the *0-cut phase* for small and large instance sizes

Optimal sol. (%)	76	74	75
Exp. appr. ratio	0.96	0.98	0.96
Exp. av. appr. ratio	0.996	0.998	0.997
Instance sizes	small	large	total

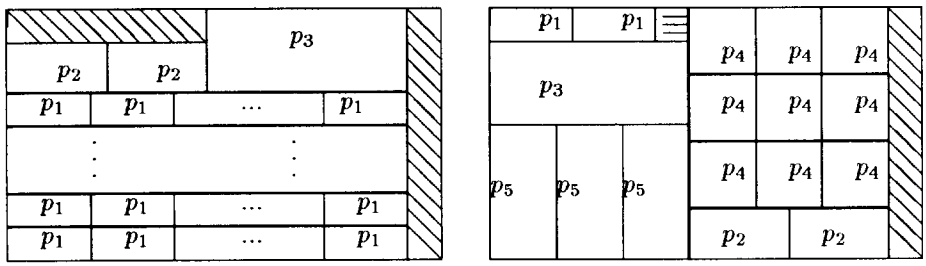
pieces were picked up uniformly on the intervals $(0, L)$ and $(0, H)$. Next, the optimal solutions were found by the first Gilmore and Gomory's algorithm.

In Table 1 we can see the performance of the BSC algorithm by referring to the percentage of optimal solutions, the average approximation ratio and the minimum ratio. The knapsack problems were solved exactly by a dynamic programming algorithm. We see that 91% of the instances are optimally solved, whereas the totality of the instances were solved very close to the optimum. The experimental approximation ratio was found to be 0.98 and the average one 0.999. Some instances taken from the literature [4,23] (we have relaxed the upper bounds on the variables), were solved exactly by our algorithm.

As concerns the *0-cut phase*, it provided also very good solutions. In Table 2, it can be seen that the worst-case experimental approximation ratio is 0.96, whereas the percentage of optimal solutions remains still very interesting (75%). Consequently, in practical applications the BSC algorithm can be stopped at this phase giving very good solutions with relatively small computational effort.

In Fig. 6 we give the solutions found by the BSC algorithm and the *0-cut phase* for an example taken from [15].

The dimensions of the initial plate are $(L, H) = (127, 98)$ and the set of pieces $p_i = (l_i, h_i)$ is $S = \{(21, 13), (36, 17), (54, 20), (24, 27), (18, 65)\}$ with $\pi_i = l_i h_i$. The solution obtained by filling the initial plate by horizontal strips (0-cut phase) is given in Fig. 6a with a value $F(L, H) = 12\,132$. This solution presents an approximation ratio equal to 0.982 and was found in 344 ms on a Univac 1100, whereas Gilmore and Gomory's algorithm provided the optimal solution in 34 823 ms. In Fig. 6b we can see an improved solution obtained by the BSC algorithm. A vertical cut has been made at the point $x_j = 54$ and the two sub-plates are both composed by horizontal strips. The value for this solution is $F(L, H) = 12\,192$ and presents an approximation of 0.987, requiring only 2297 ms. For the second problem with dimensions $(L, H) = (40, 70)$ and 10 pieces to cut, given in [4], the *0-cut phase* needs only 314 ms for finding the optimal solution. The exact algorithm gives this solution in 3962 ms.



(a) (b)

Fig. 6. (a) A cutting pattern of the initial plate $(L, H) = (127, 98)$ composed by horizontal strips. Two kinds of strips are involved: One homogeneous strip repeated 6 times and one general type strip. (b) A better cutting pattern composed also by horizontal strips, but after a vertical cut has been made at a point $x_j = 54$. The strips involved are all uniform strips.

The third problem in [4] having dimensions $(L, H) = (40, 70)$ and 20 pieces to cut was also solved to the optimality within 506 ms by the *0-cut phase*, whereas the exact algorithm needs 4512 ms. However, for small problem sizes the algorithm has not the same time performance. For example the first problem given by Christofides and Whitlock in [4] having sizes $(L, H) = (15, 10)$ and 7 pieces to cut, needs 90 ms by the 0-cut phase for an approximation solution equal to 0.996. The optimal solution was found by the BSC algorithm in 383 ms, but the exact algorithm needs only 100 ms.

It is obvious that the presented algorithms require far less computational efforts than other known algorithms, particularly for large size instances and also considerably reduced memory. Thus, they can be used to solve efficiently real-world problems of very large sizes.

8. Conclusions

In this paper we have described an efficient algorithm for solving two-space knapsack problems. From this algorithm we have derived an approximation algorithm and we have shown that this algorithm has approximation ratio equal to $\frac{4}{9}$. The algorithms deal efficiently with large size instances and they are easily generalized to deal with other versions of the TDK, for example, profits no proportional to areas, or number of appearances of each piece in the final solution bounded [25]. Finally, the algorithm can easily be used for solving general Cutting Stock problems [24].

Acknowledgement

We would like to express our thanks to W. Fernandez de la Vega for helpful comments and suggestions.

References

[1] Adamowicz, M., and Albano, A., "A solution of the rectangular cutting-stock problem", *IEEE Transactions on Systems, Man and Cybernetics* 6/4 (1976) 302–310.
[2] Balas, E., Zemel, E., "An algorithm for large knapsack problems", *Operations Research* 28 (1980) 1130–1154.
[3] Chambers, M., and Dyson, R., "The Cutting Stock Problem in the flat glass industry – Selection of stock sizes", *Operational Research Quarterly* 27 (1976) 949–957.
[4] Christofides, N., and Whitlock, C., "An algorithm for two-dimensional cutting problems", *Operations Research* 25/1 (1977) 30–44.
[5] Codd, E.G., "Multiprogram scheduling", *Communications of the ACM* 3/6 (1960) 347–350; 7 (1960) 413–418.
[6] Dyson, R., and Gregory, A., "The Cutting Stock Problem in the flat glass industry", *Operational Research Quarterly* 25 (1974) 41–54.
[7] Fayard, D., and Plateau, G., "An algorithm for the solution of the 0–1 Knapsack Problem", *Computing* 28 (1982) 269–287.
[8] Fayard, D., and Plateau, G., "Resolution of the 0–1 Knapsack Problem: Comparison of methods", *Mathematical Programming* 8/3 (1975) 272–307.
[9] Garey, M.R., and Graham, R.L., "Bounds on multiprocessing scheduling with resource constraints", *Siam Journal on Computing* 4/2 (1975) 187–200.
[10] Gilmore, P., "Cutting stock, linear programming, knapsacking, dynamic programming and integer programming; some interconnections", *Annals of Discrete Mathematics* 4 (1979) 217–235.
[11] Gilmore, P., and Gomory, R., "Multistage cutting problems of two and more dimensions", *Operations Research* 13 (1965) 94–119.
[12] Gilmore, P., and Gomory, R., "The theory and computation of knapsack functions", *Operations Research* 14 (1966) 1045–1074.
[13] Haessler, R., "A note on computation modifications to the Gilmore–Gomory cutting stock algorithm", *Operations Research* 28/4 (1980) 1001–1005.

[14] Haessler, R., and Sweeney, P.E., "Cutting stock problems and solution procedures", *European Journal of Operational Research* 54 (1991) 141–150.
[15] Herz, J., "A recursive computing procedure for two-dimensional stock cutting", *IBM Journal of Research and Development* 16 (1972) 462–469.
[16] Hinxman, A.I., "The trim-loss and assortment problems: A survey", *European Journal of Operational Research* 5 (1980) 8–18.
[17] Kantorovich, L.V., "Mathematical methods of organizing and planning production", *Management Science* 6 (1960) 363–422.
[18] Lawler, E.L., "Fast approximation algorithms for Knapsack Problems", *Mathematics of Operations Research* 4 (1979) 339–356.
[19] Martello, S., and Toth, P., "Branch and bound algorithms for the solution of the general uni-dimensional Knapsack Problem", in: *Proceedings of the Second European Congress on Operations Research*, Stockholm, 1976.
[20] Tilanus, C., and Gerhardt, C., "An application of cutting stock in the steel industry", in: K.B. Haley (ed.), *Operational Research*, North-Holland, Amsterdam, 1976, 669–675.
[21] Tokuyama, H., and Ueno, N., "The cutting stock problems in the iron and steel industries", in J.P. Brans (ed.), *Operational Research*, North-Holland, 1981, 809–823.
[22] Toth, P., "Dynamic programming algorithms for the Zero–One Knapsack Problem", *Computing* 25 (1980) 29–45.
[23] Wang, P.Y., "Two algorithms for constrained two-dimensional cutting stock problems", *Operations Research* 31/3 (1983) 573–586.
[24] Zissimopoulos, V., "Problèmes de découpe: Algorithmes ε – approchants", Thesis, L.R.I., Orsay, 1984.
[25] Zissimopoulos, V., "Heuristic methods for solving (un)constrained two dimensional cutting stock problems", *Methods of Operations Research* 49 (1985) 345–357.