UNIVERSIDADE DE SÃO PAULO

A THEORY OF THE GENERATION OF CUTTING AND PACKING PATTERNS PART I: FOUNDATIONS

MARCOS NEREU ARENALES
REINALDO MORABITO

Nº 31

NOTAS

Instituto de Ciências Matemáticas de São Carlos

Instituto de Ciências Matemáticas de São Carlos

ISSN - 0103-2577

A THEORY OF THE GENERATION OF CUTTING AND PACKING PATTERNS PART I: FOUNDATIONS

MARCOS NEREU ARENALES
REINALDO MORABITO

Nº 31

NOTAS DO ICMSC
Série Computação

São Carlos
Nov./1996

Uma Teoria da Geração de Padrões de Corte e Empacotamento Part I: Fundamentos

Marcos N. Arenales
Instituto de Ciências Matemáticas de São Carlos
Universidade de São Paulo
13.560-970 - São Carlos - SP, Brasil, c.p. 668
e-mail:arenales@icmsc.sc.usp.br

Reinaldo Morabito
Depto. de Engenharia da Produção
Universidade Federal de São Carlos
13.565-905 - São Carlos - SP, Brasil
e-mail:morabito@power.ufscar.br.

Resumo

O problema de corte e empacotamento em discussão neste artigo consiste em cortar um objeto grande em itens menores demandados de modo que uma função objetivo seja otimizada, p.ex., minimizar a perda total de material. Em outras palavras, estamos preocupados com a geração de padrões de cortes eficientes que surgem em vários algoritmos para problemas mais gerais de corte/empacotamento (p.ex., o problema de corte de peças em estoque). As formas geométricas e tamanhos do objeto e dos itens são considerados somente em termos das dimensões relevantes para o processo de cortagem, as quais definem as dimensões do problema (p.ex., em um problema unidimensional de corte somente o comprimento é considerado). A teoria apresentada neste artigo é baseada em simples hipóteses que permitem a representação de um padrão de corte/empacotamento como um caminho completo num grafo-E/OU finito e é aplicada a problemas de corte/empacotamento de diferentes dimensões e formas geométricas. Limitantes são desenvolvidos de modo a descrever um método de enumeração implícita e o conceito de regularidade é formalizado sob a luz da teoria. Problemas de corte e empacotamento são sabidos pertencerem à classe dos NP-difíceis e algoritmos exatos não são, em geral, viáveis para resolver problemas grandes que surgem nas aplicações reais. Os desenvolvimentos neste artigo levam a várias heurísticas que serão apresentadas na parte II deste artigo, onde vários estudos de casos serão também apresentados.

Palavras-chaves: Problemas de corte e empacotamento, Busca em grafo-E/OU, Método de enumeração implícita.

A Theory of the Generation of Cutting and Packing Patterns Part I: Foundations

Marcos N. Arenales
Instituto de Ciências Matemáticas de São Carlos
Universidade de São Paulo
13.560-970 - São Carlos - SP, Brazil, c.p. 668
e-mail:arenales@icmsc.sc.usp.br

Reinaldo Morabito
Depto. de Engenharia da Produção
Universidade Federal de São Carlos
13.565-905 - São Carlos - SP, Brazil
e-mail:morabito@power.ufscar.br.

Abstract

The cutting/packing problem addressed in this paper consists of cutting a large object into smaller ordered items in such a way as to optimize an objective function, e.g., to minimize the total waste material. In other words, we are concerned with generating efficient cutting/packing patterns that arise in a number of algorithms for more general cutting/packing problems (e.g. the cutting stock problem). The shapes and sizes of the object and the items are considered only in terms of their relevant dimensions for the cutting/packing process, which define the dimensions of the problem (e.g. in a 1-dimensional cutting problem only the length is taken into account). The theory presented in this paper is based on simple hypotheses that allow the representation of a cutting/packing pattern as a complete path in a finite *AND/OR*-graph and is applicable to cutting/packing problems of different dimensions and geometric shapes. Bounds are provided making it possible to describe a branch and bound method, and the concept of regularity is formalized in light of the theory. Cutting/packing problems are known to be NP-hard and exact algorithms are usually not able to solve large problems that arise in real-world applications. The developments in this paper lead to a number of heuristics to be presented in part II of this paper, where several case studies will be provided.

Keywords: cutting/packing problems, AND/OR-graph search, branch and bound method.

1 Introduction

In this paper we are concerned with the problem of cutting a large object into smaller ordered items in such a way as to optimize an objective function, e.g., to minimize the total waste, or if utility values are assigned to the items, the objective could be to maximize the total value given by the sum of the utility values of the produced items. Note that these objective functions depend only on the number of items produced, which is a fundamental characteristic for the theory proposed in this paper. The reverse problem of packing items into a larger object (e.g., boxes into a container) has the same structure as the cutting problem since the empty space in the object can be thought of as being cut in order to produce the corresponding spaces that items occupy (e.g. see Dyckhoff, 1990 and Dowsland and Dowsland, 1992, for surveys of cutting and packing problems).

If a specified demand for items has to be satisfied by cutting a certain number of objects, the problem is referred to in the literature as a *cutting stock problem.* In this case the cutting problem above (where just one object is cut) works as a fundamental sub-problem of generating efficient cutting patterns for most solution methods (see Golden, 1976, and Hinxman, 1980, for method-oriented surveys).

Cutting problems arise in a number of industries and are essential for production planning for industries such as paper, wood, metal, glass, plastic, furniture, textile, mattresses, etc. (see Dyckhoff et al., 1985, for a problem-oriented survey).

It is worth noting that when a cutting/packing optimizer is implemented in an industry people perceive other related problems. The authors had an interesting experience in a furniture factory where the staff, at first, wanted to eliminate the difficult and unpleasant task of, each morning, choosing rectangular plates and deciding how they should be cut in order to produce a number of smaller rectangles. After having defined a cutting pattern by hand for a type of chosen plate, trying to minimize the waste (the cutting problem), this pattern was repeated until the demand for one or more smaller rectangles was reached (this is known in the literature as *repeated exhaustion reduction heuristic*). Note that they had the two basic problems related to this area (Hinxman, 1980): the assortment problem (i.e., what should the stock sizes be) and the cutting stock problem (i.e., given the stock sizes, how the stock should be cut). The assortment problem was until then unthinkable. The supplier of the larger plates offered five common types (i.e., plates of five different sizes) of which the furniture factory usually bought three types without any selection criteria. The supplier also offered five other uncommon types at a 10% discount but with limited stock. These uncommon types were smaller than the ordinary ones (and for this reason were never chosen) but were large enough to be used in the furniture industry. With the optimizer, they pretended to have all sizes in stock, and using a forecast demand, they were amazed to learn that the computer program chose to cut all of the available uncommon types. (The program is a two-staged guillotine cutting problem for the column generation procedure of Gilmore and Gomory, 1965, for the LP model with stock limitation). Since then, the optimizer has been used not only to solve the cutting stock problem, but also to make the purchases (the assortment problem).

The seminal works of Gilmore and Gomory (1961,1963,1965,1966), who proposed meth-

ods that enable the resolution of large real-world problems, as well as the evolution of computer technology, have increasingly attracted the attention of academic researchers and industrial practitioners in the cutting/packing area, especially in the last few years, as can be seen from special issues in OR journals (Dyckhoff and Wäscher, 1990; Martelo, 1994 and Bischoff and Wäscher, 1995), specialized books (e.g. Martelo and Toth, 1990; Dyckhoff and Finke,1992), chapters in optimization books (e.g. Lasdon, 1970, Chvatal,1983), and several surveys (see References section). However, the diversity of real-world problems and the complexity of cutting/packing problems have stimulated people to develop heuristic methods focusing on case studies. In spite of this, it is possible to theorize on this subject by generalizing properties and methods used in particular instances to more general cases.

In the pioneering and singular work of Gilmore and Gomory (1966), "The Theory and Computation of Knapsack Functions" (the cutting problem defined above was called a knapsack function), the authors made an effort to characterize properties of functions that supported their previous work, as well as to review and improve the methods based on dynamic programming. However, that theory preserved in essence a limitation of their previous work which handled regular pieces (i.e., bar, rectangle or parallelepiped type) and the cutting process was restricted to the guillotine type.

Our previous experience solving some particular cutting/packing problems (Morabito et al, 1992; Morabito and Arenales, 1994, 1995, 1996; Arenales and Morabito, 1995) led us to use a problem reduction method (Nilsson,1971), where the initial problem is broken into smaller sub-problems, which in turn may be separated. Each resulting sub-problem does not itself provide a solution to the problem before the separation is made, but the union of their solutions does. A kind of data structure called an AND/OR-graph is used to represent this method. Since the initial problem is of optimizing, an implicit enumeration (branch and bound method) can be devised by smaller modifications of the ordinary principles of branching and pruning.

Initially we were concerned with some specific cutting/packing problems, such as guillotine or nonguillotine, staged or multi-staged, restricted or unrestricted, and two-dimensional or three-dimensional cutting/packing problems. Fig. 1 depicts some examples of cutting patterns able to be generated using the AND/OR-graph approach from our previous work.

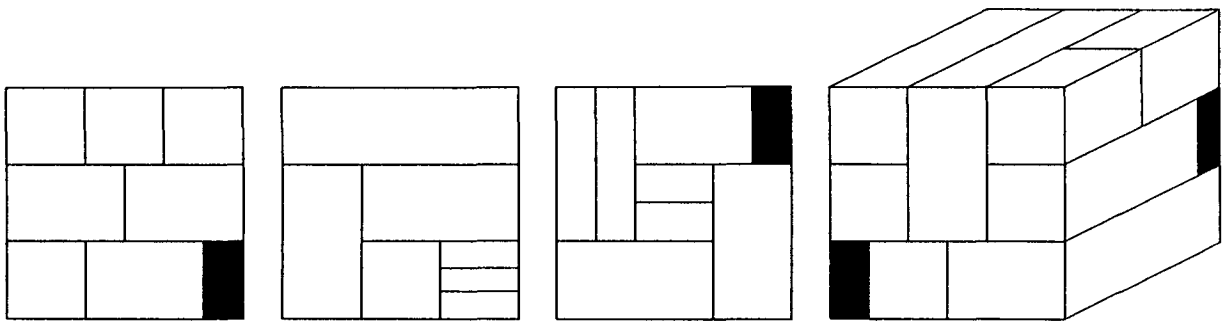


Figure 1. Types of Cutting/Packing Patterns already studied using AND/OR-graph Approach

a) Staged, Constrained or not, Two-Dimensional Guillotine Cutting
b) Multistaged, Constrained or not, Two-Dimensional Guillotine Cutting
c) Two-dimensional Nonguillotine Cutting
d) Three-Dimensional Guillotine Cutting

The theory to be presented in this paper generalizes our previous work, showing that the approaches utilized to solve the above-mentioned cutting/packing problems can be applied to handle many other cutting/packing problems in such a way that the dimensions of the problem and the geometry of the object and the items are irrelevant from theoretical point of view (obviously they are highly relevant from the computational point of view).

So, different from the previous work where specific problems were under consideration, we now focus on the broadness of the approach to representing and searching for the solution set for more general cutting/packing problems. The issue of computational efficacy in this approach to more complex problems is opened, but the successful results already obtained by dealing with some specific problems have encouraged us in this research since they suggest that fruitful results can also be found by either improving problem-based heuristic methods or coping with new cutting/packing problems not yet explored. Fig.2 illustrates some kinds of cutting/packing patterns involving different dimensions and geometric shapes of items and objects for which the theory is applicable.

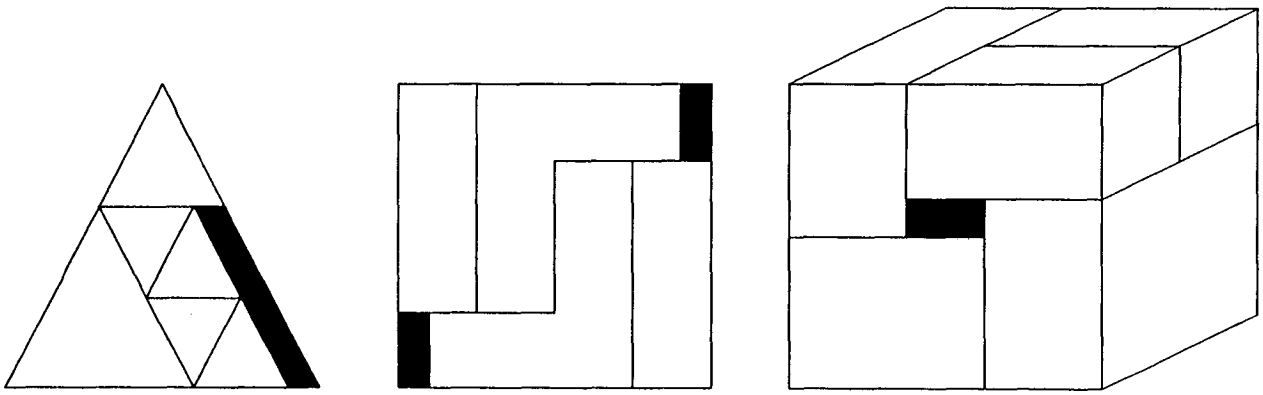


Figure 1 - Types of cutting/packing patterns treatable by the theory

This study is divided into two parts: Part I (Foundations) presents assumptions, concepts, theorical results and an exact branch and bound method, and Part II (Computations) will describe heuristics, case studies and computational results.

Part I has the following organization. Section 2 presents the basic assumptions, concepts and results. The representation of the solution space as paths in an AND/OR-graph is in Section 3 which also contains a number of feasibility and optimality properties which are the basis for the branch and bound method presented in Section 4. In Section 5 we define what a regular problem is, in light of the theory presented here.

2 The Definition of the Problem

Cutting/packing patterns consist of the way items are arranged in a particular object. This arrangement depends, of course, on the shapes and sizes of the items and the object, and rules imposed by the cutting or packing process. The shapes and sizes are considered only in terms of relevant dimensions for the cutting or packing process which define the dimensions of the problem.

Example 1. The cutting of a large bar (object) into smaller pieces (items) which considers only the lengths of the pieces and the bar is a one-dimensional cutting problem. Similarly, the cutting of a large plate into smaller rectangles considers only the length and width of the items and the object, so it is a two-dimensional cutting problem. The problem of packing boxes of the same height on a pallet is also a two-dimensional problem since the loading must be made by layers where only length and width are considered. Other examples can be found in the literature, including n-dimensional problems arising in production planning, where the relevant dimensions are time and other resources (see Dyckhoff and Finke, 1992). ■

How to efficiently generate cutting/packing patterns is the core of most methods to solve cutting/packing problems, and it is the problem we are focusing on in this paper which is better defined below. Because of the importance of this problem several authors have studied it, but only for particular instances, such as, two-dimensional guillotine cutting and container loading. Surveys of applications and methods are in Hinxman, 1980; Dyckhoff et al., 1985; Dyckhoff, 1990; Sweeney and Paternoster, 1992; and Dowsland and Dowsland, 1992.

Suppose that m types of items are to be produced, and v_i is a utility value for item i, $i = 1, \ldots, m$. These utility values can be defined in terms of the simplex multiplier vector in the column generation procedure of Gilmore and Gomory (1961,1963,1965) to solve the cutting stock problem where the quantity of each item to be produced is also specified. Alternatively, they can be heuristically defined in a repeated exhaustion reduction procedure (Hinxman, 1980). An efficient cutting/packing pattern on a particular object is one that maximizes the total utility value. This problem can be stated as determining $(y_1, y_2, \ldots, y_m)$, such that:

$$maximize\ v_1 y_1 + v_2 y_2 + \ldots + v_m y_m \tag{1}$$

$$\text{subject to}: \quad (y_1, y_2, \ldots, y_m) \in \mathbb{P}\ , \tag{2}$$

where $\mathbb{P}$ is the solution space defined by:

$\mathbb{P} = \{(y_1, y_2, \ldots, y_m)$ of non-negative integer number, such that there exists at least one cutting/packing pattern that produces y_i items of type $i, i = 1, \ldots, m\}$.

Example 2. The solution space $\mathbb{P}$ can be sometimes modeled as a system of inequalities. For example, the set $\mathbb{P}$ for the simplest one-dimensional cutting problem (also known as a knapsack problem) is characterized by:

$$\sum_{i=1}^{m} l_i y_i \leq L, \tag{3}$$

$$y_i \geq 0 \text{ and integer, } i = 1, 2, \ldots, m.$$

where L is the length of the object and l_i the length of item $i, i = 1, 2, \ldots, m$.

The inequality (3) is concerned only with physical aspects but other constraints can arise from the cutting process. For example, if the total number of items in the pattern is limited by F, then the following inequality also has to be considered:

$$\sum_{i=1}^{m} y_i \leq F.$$

(see Gilmore and Gomory(1963) for an application in a paper factory where $F+1$ is the number of knives in the cutting process)

The problem (1)-(2) is therefore written by:

$$\begin{aligned} & maximize\ v_1 y_1 + v_2 y_2 + \ldots + v_m y_m \\ \text{subject to}: \quad & l_1 y_1 + l_2 y_2 + \ldots + l_m y_m \leq L \\ & y_1 + y_2 + \ldots + y_m \leq F \\ & y_i \geq 0 \text{ and integer, } i = 1, 2, \ldots m. \end{aligned}$$

This model is known in the literature as a two-dimensional knapsack problem (Martelo and Toth, 1990), but it should be clear that it does not model a two-dimensional cutting problem. It can be thought of as handling rectangles where the object has length L and width F, and the items have lengths $l_i i = 1, 2, \ldots m$ and the same width equal to one. But a feasible cutting pattern is illustrated in Fig. 2a, while the cutting pattern in Fig. 2b is infeasible since $\Sigma_i l_i y_i > L$.

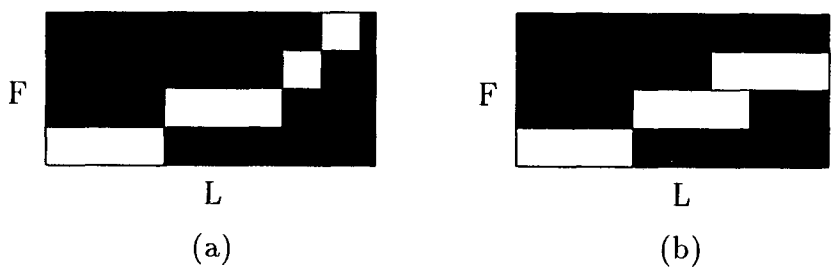


Figure 2. a) Feasible cutting pattern for two-dimensional knapsack problem

b) Infeasible Solution: $\Sigma_i l_i y_i > L$. ■

It should be clear in defining problem (1)-(2) that the emphasis is put on the number of items in the cutting pattern and not on their particular positions in the object. Of course, this problem does not model a container loading problem, for example, where the unloading of items has an order given by a previously defined route, such that well-allocated items, considering only the minimization of empty space in the container, could cause a problem if they were under a number of others to be unloaded later. Then the utility values should vary with the position of the items in the object (however, heuristics based on problem (1) - (2) can be devised for such problems). Then, the following basic hypothesis was implicitly assumed when problem (1)-(2) was stated.

Hypothesis 1. Patterns corresponding to the same element of $\mathbb{P}$ (i.e., having the same number of each items) are equivalent.

The Fig. 3 illustrates equivalent 2-dimensional cutting patterns. Of course, an additional constraint on the cutting process could discard the third pattern. This is taken into account when feasible cuts are being defined (see Definition 1).

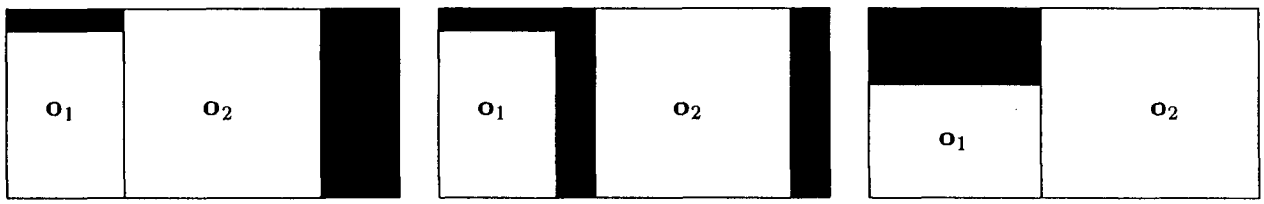


Figure 3. - Equivalent Cutting Patterns

The way we first defined cutting/packing patterns, the items could be arranged in the object in any non-overlaping way. Fortunately, for a number of significant problems this is not the case, and a cutting/packing pattern must be formed based on a number of rules, e.g., orthogonal guillotine cutting. In this paper we consider the cutting/packing formation as recursively built, that is, the original object is separated into parts (new smaller objects) using a specific rule, then each part can be separated again by means of other rules. This process continues until the decision of allocating items in the generated parts finally producing a cutting/packing pattern. The above mentioned rules are called elementary cuts, since they give the idea of cutting the object although the problem could also be one of packing as well.

An alternative way of building a cutting/packing pattern can be found in Wang(1983) and Daza et al. (1995), where items are combined by using specific rules to form smaller packings, then these smaller packings and items can be combined again until the dimensions of the object are reached. These rules for combining items could be thought of as *elementary packing*, and extended beyond the way they were considered in the above papers. These

approaches can seen as dual of each other.

Definition 1. *(Elementary Cut)* An elementary cut is a previously-defined rule that when applied to a particular object produces other disconnected objects in such a way that if they were conveniently glued together again the previous object would be successfully obtained (i.e., it is assumed that no part of the object is lost nor is there any deformation).

In practice, the elementary cuts are defined beforehand based on simple rules, but of course they depend on the case study (see the guillotine elementary cut in example 4).

Definition 2. *(AND-successors, partial cutting pattern)* Let $\mathbf{o}$ be an object, and an elementary cut defined on $\mathbf{o}$ be denoted by $C(\mathbf{o}) = (\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_s)$, where $\mathbf{o}_i, i = 1, \ldots, s$, are new objects produced by the elementary cut C, which may or may not be items. The objects $\mathbf{o}_i, i = 1, \ldots, s$, are called *AND-successors* of $\mathbf{o}$. The elementary cut C assigns a partial cutting pattern to the object $\mathbf{o}$ formed by the objects $\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_s$ that we denote as $C^{-1}(\mathbf{o}_1, \mathbf{o}_2, \ldots, \mathbf{o}_s)$ (see Fig. 4b).

Three basic hypotheses are assumed now to allow the set of all possible cutting/packing patterns to be represented as a finite AND/OR-graph (Section 3).

Hypothesis 2. Every object (the original and the generated ones) is assumed to be limited and having a nonempty interior.

This simple and realistic hypothesis assures the following theorem.

Theorem 1. The solution space $\mathbb{P}$ is finite.

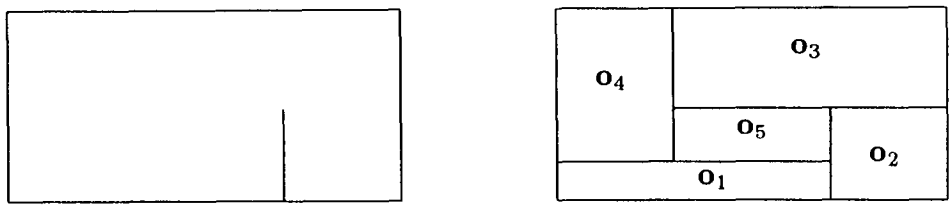
Hypothesis 3. Every cutting/packing pattern is obtained by a sequence of elementary cuts, in the following way: an elementary cut is made on the original object, producing new objects. Each new object can be independently cut again by an elementary cut, and so on, until a stopping criterion is verified.

Hypothesis 4. Every item can be obtained by a finite sequence of elementary cuts.

The stopping criterion mentioned above can be stated using a special elementary cut, called *O-cut (zero-cut)*, that maintains the object intact. That is, if C_0 denotes the *O-cut*, then $C_0(\mathbf{o}) = \mathbf{o}$, for every object $\mathbf{o}$. After a *O-cut* is made on an object, no other cut is allowed. In this case the object can be an item or waste. Other criteria can be devised using lower and upper bounds, since an optimization problem is being solved and an implicit enumeration of the cutting/packing patterns is possible (see Section 4).

Example 3. It is interesting to note that we do not recognize the action of an "interrupted cut" as an elementary cut, as illustrated in Fig. 4a, since it does not produce disconnected

objects. Therefore, the cutting in Fig. 4b is not seen as a sequence of interrupted cuts, but simply as an elementary cut producing 5 new objects. It was called a 1^{st} order non-guillotine cut in Arenales and Morabito (1995). We also exclude the possibility of cutting circles using elementary cuts based only on straight lines (hypothesis 4).



(a) (b)

Figure 4. a) interrupted cut (not an elementary cut).
b) the partial cutting pattern $C^{-1}(o_1, o_2, o_3, o_4, o_5)$ by a 1^{st} order non-guillotine elementary cut. ■

Problem (1) - (2) indicates that no distinction is made among different cutting patterns which correspond to the same element in IP (Hypothesis 1). Although there are infinite cutting patterns, we can only pay attention to a finite number of them, since IP is finite.

These hypotheses yield the following theorem.

Theorem 2. Without loss of generality, the elementary cuts can be taken from a finite set.

Proof: By contradiction, suppose that infinite elementary cuts are needed in order to obtain all non-equivalent cutting patterns. This would occur if:

i) infinite alternatives of cutting an object produced non-equivalent cutting patterns. This contradicts the fact that IP is finite; or

ii) an infinite sequence of elementary cuts was needed in order to produce a particular cutting pattern. This contradicts our assumption (hypothesis 4) that every item can be obtained by a finite sequence of elementary cuts. ■

As a consequence of this theorem we can state the following corollary.

Corollary 1. The set of objects produced by elementary cuts can be considered a finite set.

Note that the geometric shapes of the object and the items were transferred to the definition of elementary cuts. We still emphasize that these results were possible only because problem (1) - (2) makes no distinction among equivalent cutting patterns. Then, the cutting/packing problem can be seen as a discrete optimization problem.

In the next section we shall show how to organize the cutting/packing patterns as complete paths in a finite graph.

3 AND/OR-Graph Representation

3.1 The Graph Definition

A graph $G = (V, E)$ consists of a finite, non-empty set $V = \{1, 2, \ldots, r\}$ and a set $E = \{e_1, e_2, \ldots, e_s\}$ whose elements are subsets of V of size 2, that is, $e_u = (i, j)$, where $i, j \in V$. The elements of V are called vertices (or nodes) and the elements of E are called edges (or arcs).

One way of generalizing a graph is to allow arcs in E of different sizes, for example, an $e_u = (i, j, k)$ where $i, j, k \in V$. For this generalization $G = (V, E)$ is called a hypergraph.

Another way of generalizing a graph is to define the arcs as pairs $e_u = (i, V_u)$ where $i \in V$ and $V_u \subset V$, for example, an arc $e_u = (i, \{j, k\})$ where $i \in V$ and $\{j, k\} \subset V$. The arc e_u is said to be emerging from i and pointing to j and k. Different arcs emerging from i are called OR-arcs. If V_u has a cardinality greater than 1, then e_u is called an AND-arc and G an AND/OR-graph.

Note that an arc of a graph defines a relationship between two nodes, an arc of a hypergraph defines a relationship among a subset of nodes, and an arc of an AND/OR-graph defines a relationship between a node and a subset of nodes (see Berge, 1973 and Pearl, 1984).

In order to produce items (demanded smaller pieces), we first cut the original object (large piece in stock) using a defined rule (elementary cut) obtaining new intermediary objects, which are successively and independently cut until the items are obtained. This procedure determines an arrangement of the items in the original object. This is the way we assume a cutting pattern is built (Hypothesis 3).

For example, suppose that in the first stage of the sequence of cuts the original object, $\mathbf{o}$, is cut by an elementary cut, C_1, producing intermediary objects $\mathbf{o}_1$, $\mathbf{o}_2$ and $\mathbf{o}_3$. Then, these *AND*-successors of $\mathbf{o}$ are independently cut. The object $\mathbf{o}_1$ is cut by C_2 producing $\mathbf{o}_4$ and $\mathbf{o}_5$ which are kept intact, i.e., they are cut by a *0*-cut which is denoted by C_0; the object $\mathbf{o}_2$ is then cut by C_3, producing $\mathbf{o}_6$, $\mathbf{o}_7$ and $\mathbf{o}_8$ while $\mathbf{o}_3$ is cut by a *0*-cut. Then, $\mathbf{o}_6$ is cut by C_4, producing $\mathbf{o}_9$ and $\mathbf{o}_{10}$, which are in the end kept intact by *0*-cuts, as are $\mathbf{o}_7$ and $\mathbf{o}_8$. Therefore, the cutting process ends.

Note that just one elementary cut can be chosen to be made on each object. But different choices of elementary cuts would lead to different cutting sequences, and new cutting patterns would be determined. The cutting sequence described above is schematized in Fig. 5.

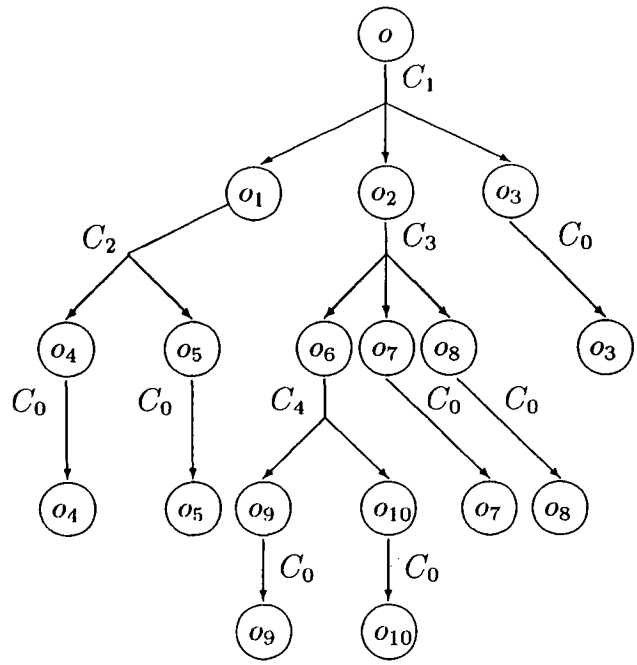


Figure 5 - A Cutting Sequence (sequence of *AND*-arcs)

Example 4. As an illustration, we consider the problem of cutting a rectangular plate into rectangular pieces by 1^{st}-order non-guillotine cuts (Arenales and Morabito, 1995), using 3 types of elementary cuts on an object (i.e. a rectangle $\mathbf{o} \equiv (a,b)$), plus the *0*-cut. In this example any object generated can be well-represented by 2 parameters: length and width (this is a regular problem which we will define better in Section 5). The feasible elementary cuts are:

a) *vertical guillotine cut:* $V_x(a,b) = ((x,b),(a-x,b))$, $o < x < a$.

b) *horizontal guillotine cut:* $H_y(a,b) = ((a,y),(a,b-y))$, $o < y < b$.

c) 1^{st}*-order non-guillotine cut:*

$$N_{r,s,u,v}(a,b) = ((r,u),(a-r,v),(a-s,b-v),(s,b-u),(r-s,u-v)),$$

$$o < s < r < a,\ o < u < v < b.$$

Figs. 6a and 6b depict vertical and horizontal guillotine cuts, and in Fig. 6c we can see the outcome of applying a 1^{st}-order cut on the rectangle (a,b).

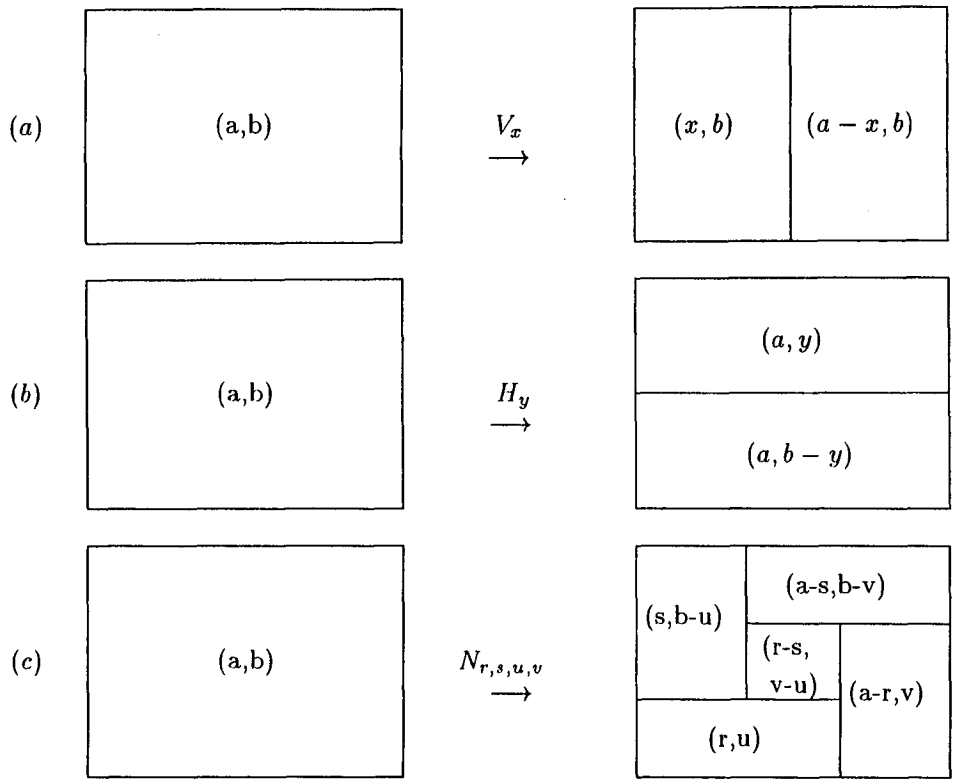


Figure 6 - Vertical, horizontal and 1^{st} order nonguillotine elementary cuts.

Note that although we have only 4 types of elementary cuts (including the *0*-cut), the number of elementary cuts for each type is very large and is dependent on the parameters x, y, r, s, u and v, which assume values in a finite *discretization* set (Arenales and Morabito, 1995). The finite discretization sets are specific versions of Theorem 2 (see Theorem 7). This kind of set, also called a breakpoint set, has been used for a longtime, see Salkin and Kluyven, 1975; Herz, 1972; Christofides and Whitloch, 1979; and Beasley, 1985, among others. Fig. 7a shows a 1^{st}-order non-guillotine cutting pattern obtained by a sequence of elementary cuts defined above, and schematized in Fig. 7b. (Each cut C_k belongs to the class of vertical, horizontal or 1^{st}-order non-guillotine cuts defined above and the corresponding class is assigned as V, H, or N, respectively).

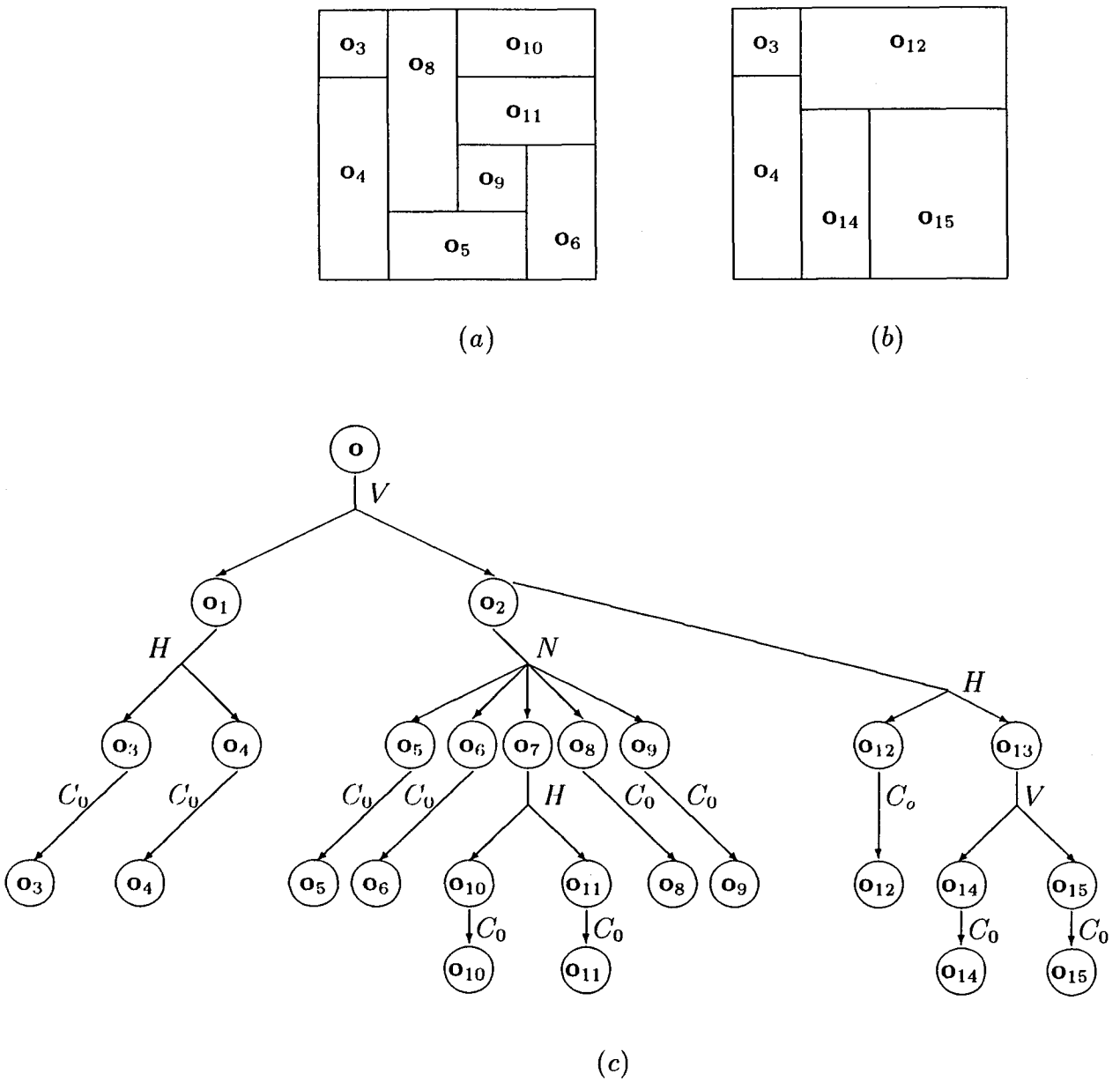


Figure 7 - Cutting patterns and the corresponding sequences of elementary cuts. ■

The Definition of Nodes

For each newly generated object (intermediary object) we have a subproblem which is similar to the original problem of how to determine the best cutting pattern for that object. Each intermediary object (associated with its problem) is represented by a node in a graph, where the original object is represented by the initial node. Nodes representing *AND*-successors are also called *AND*-nodes.

The Definition of Arcs

The decision to cut an object by an elementary cut is represented as an *AND*-arc that emerges from that object and points to the *AND*-nodes which are the outcome of the cut. *AND*-arcs define the links among intermediary objects. The *0*-cut is represented as an ordinary arc pointing to a facsimile of the node from which it emerges (see Fig. 5 or 7).

We use the same notation of an elementary cut to the corresponding *AND*-arc: if the elementary cut made on the object **o** is $C(\mathbf{o})=(\mathbf{o}_1,\mathbf{o}_2,\ldots,\mathbf{o}_s)$, and the node N represents the object **o**, and N_1, N_2 ,...,N_s represent the outcome objects $o_1,o_2,\ldots,o_s$, then we denote the *AND*-arc as $C(N)=(N_1,N_2,\ldots,N_s)$, and the partial cutting pattern as $C^{-1}(N_1,N_2,\ldots,N_s) = C^{-1}(o_1,o_2,\ldots,o_s)$.

In the example above of non-guillotine cutting, the guillotine elementary cuts point to two new rectangles (*AND-nodes*), whereas non- guillotine elementary cuts point to five *AND*-nodes. Fig. 7b illustrates these *AND*-arcs.

The definition of elementary cuts depends on the case study. For example, in the wooden rectangular plate cutting problem the elementary cut must necessarily be of the guillotine type, whereas for the pallet loading problem one can have non-guillotine cutting patterns. Other cutting problems could use more complex elementary cuts allowing the generation of L-shaped items, for example.

Different options of branching a node are represented by *OR*-arcs. The set of all nodes and arcs is called an *AND/OR*-graph. This *AND/OR*-graph can be considered finite since the number of elementary cuts and intermediary objects can be taken from finite sets.

3.2 Feasibility Properties

Definition 3. (*Final Node*) A node pointed to by a *0*-cut has no successors and is called a final node. The object represented by a final node may or may not be an item. If not, it is considered waste.

Definition 4. (*Complete Path*) Consider the following sequence of arcs in the AND/OR graph: from a node N choose one and only one arc (*AND*-arc or *0*-cut arc), and from each AND-successor node pointed to by this arc, choose again one and only one arc, and so on, until all successor nodes obtained are final nodes. This sub-graph is called a complete path from N; and if N is the initial node, it is simply called a complete path.

Definition 5. *(Homogeneous Cutting Pattern)* If the final nodes in a complete path represent only one type of item or waste, the corresponding cutting pattern is called homogeneous, that is, a cutting pattern formed by just one type of item.

For each node N in the graph there is a cutting problem defined for the object **o** represented by that node. The cutting patterns for that object produce a subset of $\mathbb{P}$, which is denoted by $\mathbb{P}(N)$. A homogeneous cutting pattern for node N using item k produces $(y_1,\ldots,y_m) \in \mathbb{P}(\mathrm{N})$ such that $y_i = 0, i = 1,2,\ldots,m, i \neq k$.

Example 5. If node N represents a rectangular object $\mathbf{o} \equiv (a, b)$ and item k is also rectangular shaped defined by (a_k, b_k), then $y_k = [a/a_k] \cdot [b/b_k]$ provides an easy homogeneous solution (see Fig. 8), where $[x]$ is the largest integer $\leq x$. It should be clear that the homogeneous cutting pattern must agree with the elementary cuts. In Example 2 (see Fig. 2), the value of y_k should be: $y_k = minimum\{a/l_k, b\}$, since the item k is defined as $(l_k, 1)$. If item k is not rectangular shaped, a more complicated approach is needed in order to produce an easy homogeneous solution. For example, it is possible to pack one or more items into an artificial rectangle (a_k, b_k) and then calculate y_k equals $[a/a_k][b/b_k]$ times the number of items inside the artificial rectangle.

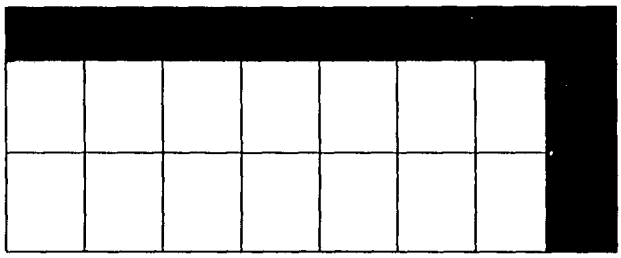


Figure 8 - A homogeneous solution for rectangles ■

By constructing the *AND/OR*-graph we can state the following Theorem.

Theorem 3. Every cutting pattern for the original object (defined by a given elementary cut set) is represented as at least a complete path in the *AND/OR*-graph described above. And vice versa, a complete path in the *AND/OR*-graph corresponds to a cutting pattern for the original object.

Note that the correspondence in Theorem 3 is not a one-to-one correspondence, since different paths may correspond to the same cutting pattern. Fig. 9 depicts such a duplication using a simple one-dimensional problem. Numbers outside the circles are the length of intermediary objects.

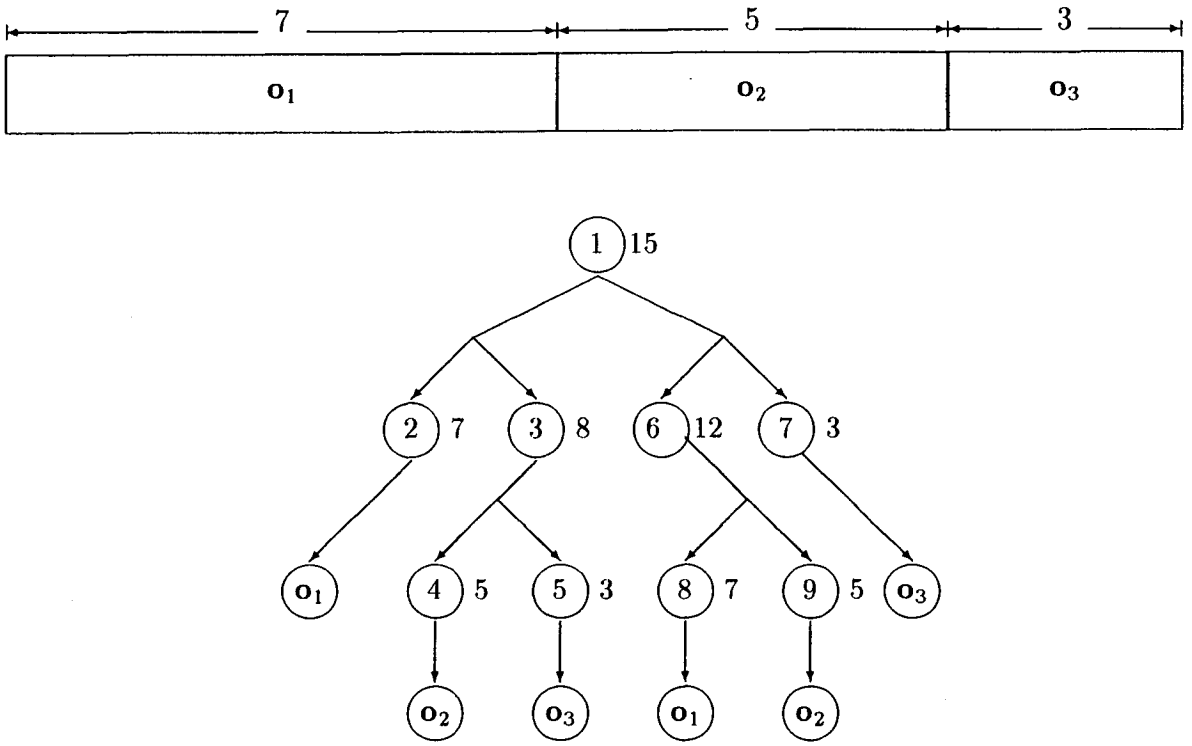


Figure 9 - Different paths leading to the same pattern

In our definition of the cutting pattern, the sequence of the elementary cuts (or *AND*-arcs in the *AND/OR*-graph) should be explicit and different sequences imply different paths, although they can produce the same cutting pattern. On the other hand, different cutting patterns can produce the same element in $\mathbb{P}$, since in $\mathbb{P}$ only the number of items is taken into account.

These distinctions are interesting because the sequence of cuts may be relevant for a particular problem, because of technical or operational restrictions, where the sequence of cuts may be important due to manipulations of intermediary objects. That is, together with the physical constraints which only take into account the number of generated items, other constraints may arise that consider the sequence of the cuts. The solution space for this problem should be the complete paths in the *AND/OR*-graph rather than set $\mathbb{P}$.

As we have already stressed, the finiteness of the sets of cutting patterns and complete paths are due to the problem being defined using set $\mathbb{P}$. Therefore, many or even infinite cutting patterns and complete paths are omitted, since the approach is based on a finite set of elementary cuts, but they could be relevant to the optimal solution if the solution space is changed. This has not yet been investigated and may be a theme for future research.

3.3 Optimality Properties

In Section 3.2 we were concerned with representing feasible solutions. Now we characterize optimal solutions.

The problem for node N is given by:

$$F^*(N) = maximum\ v_1y_1 + v_2y_2 + \ldots + v_my_m$$

$$\text{subject to : } (y_1, y_2, \ldots, y_m) \in \mathbb{P}(N)$$

The above problem is (1)-(2), if N is the initial node.

Definition 6. (*Solved Node*) A node is called *solved* if its optimal cutting pattern is known. A final node N, representing the object **o**, is considered solved and its value is given by:

$$F^*(N) = \begin{cases} v_i \ , & \text{if } \mathbf{o} \equiv \text{ item } i \\ 0 \ , & \text{otherwise.} \end{cases}$$

Theorem 4. If all successors of a node N are solved, then this node is also solved and the optimal cutting pattern is given by $C_i^{-1}(N_{i1}, N_{i2} \ldots)$, where $N_{i1}, N_{i2}, \ldots$ are AND-sucessors of N and i such that:

$$F^*(N_{i1}) + F^*(N_{i2}) + \ldots$$

$$= max\ \{F^*(N_{k1}) + F^*(N_{k2}) + \ldots, \text{ for all } k, \text{ such that } C_k(N) = (N_{k1}, N_{k2}, \ldots)\} .$$

To prove Theorem 4 it is sufficient to bear in mind that we assumed that a cutting pattern (in particular the optimal one) is obtained from a sequence of elementary cuts. That is, there is no chance of obtaining a cutting pattern for an object without dealing with its successors.

Definition 7. (*Complete Path Value*) The value of a complete path from a node N is the sum of the values of its final nodes.

If we restrict the solution space to homogeneous solutions, what follows is the homogeneous problem:

$$\begin{aligned} &\underset{y_k}{maximize}\ \ v_k y_k \quad (\equiv v_k\ maximize\ y_k) \\ &\text{subject to : } \ (0, \ldots, y_k, \ldots 0) \in \mathbb{P}(\mathrm{N}) \ . \end{aligned} \tag{4}$$

The solution of the homogeneous problem is not itself a trivial task, even for a simple cutting problem such as when the object and items are rectangles (the pallet loading problem: rectangular boxes have to be arranged on a rectangular pallet). It depends on how the

elementary cuts are defined (e.g. in the pallet loading problem, non-guillotine cutting are acceptable). However, if we are interested in obtaining easy solutions for the original problem, then a good solution would be enough (not necessarily optimal for the homogeneous problem), if it was easy to be determined (see Section 3.2).

Let $H_k(N)$ be the adopted solution using item k for the homogeneous problem (4) (not necessarily optimal), and let y_k^H be the number of items k in $H_k(N)$. This enables us to assign a first value to node N:

$$F^H(N) = max\ \left\{v_k y_k^H, k = 1, \ldots, m\right\}$$

which provides a lower bound to $F^*(N) : F^H(N) \leq F^*(N)$.

If on the one hand restricted problems (homogeneous problems) give lower bounds, on the other relaxed problems provide upper bounds, i.e., a solution of a solution of a relaxed problem provides a value $\overline{F}(N)$ such that:

$$\overline{F}(N) \geq F^*(N).$$

Although the constraints in the cutting problem are not available in general, it is possible to write a constraint that has to be verified for every feasible solution.

For the problem represented at node N, let σ_i be the hypervolume of item i and σ the hypervolume of the object **o**. For example, if item i is the rectangle (a_i, b_i) and object **o** is the rectangle (a, b), then the hypervolumes are their areas: $\sigma_i = a_i b_i$ and $\sigma = ab$. Their volumes would be considered if they were in 3-dimensional space.

Let IM(N) be the set of items that can be obtained from the object **o** represented in N. If y_i is the number of items type i in any pattern, then: $\sum_{i \in \mathbb{M}(N)} \sigma_i y_i \leq \sigma$

So, we can write a volume-based *relaxed problem* for node N as:

$$\begin{aligned} & maximize\ \textstyle\sum_{i \in \mathbb{M}(N)} v_i y_i \\ \text{subject to :}\ & \textstyle\sum_{i \in \mathbb{M}(N)} \sigma_i y_i \leq \sigma \qquad (5) \\ & y_i \geq 0 \text{ and integer, } i \in \mathbb{M}(N). \end{aligned}$$

The relaxed problem above is the *knapsack problem.* However, it is important to have easily computed upper bounds and we have preferred to relax problem (5), discarding the integral condition on the variables leading to a very easy linear program, whose solution is given by:

$$\overline{F}(N) = \sigma v_k / \sigma_k = max_i \{\sigma v_i / \sigma_i,\ i \in \mathbb{M}(N)\}$$

Example 6. Suppose that the object **o** is a rectangle **o** $\equiv (a, b)$, and that the items are rectangles also defined by $(a_i, b_i), i = 1, \ldots, m$. In addition, suppose that the items should be cut with their sides parallel to sides of the object and that they cannot be rotated, then

$$\mathbb{M}(N) = \{i \text{ such that } a_i \leq a \text{ and } b_i \leq b\}$$

and

$$\overline{F}(N) = v_k(ab)/(a_k b_k) \;=\; max \sum_{i \in \mathbb{M}(N)} v_i y_i$$

$$\text{Subject to :} \qquad \sum_{i \in \mathbb{M}(N)} (a_i b_i) y_i \;\leq\; (ab)$$

$$y_i \;\geq\; 0,\; i \in \mathbb{M}(N).$$

since the solution of the problem above is given by $y_k = (ab)/(a_k b_k), y_i = 0, i \neq k$, where k is such that: $v_k/(a_k b_k) = max\ \{v_i/(a_i b_i), i \in \mathbb{M}(N)\}$. ■

If the items and object are not rectangles, the approximated hypervolume can be used in problem (5) by choosing $\overline{\sigma}_i \leq \sigma_i$ and $\sigma \leq \overline{\sigma}$, where $\overline{\sigma}_i$ and $\overline{\sigma}$ are approximations of the hypervolumes of item i and object **o**, respectively.

Of course, $F^*(N) \;\leq\; \overline{F}(N)$. Let $F(N)$ be best value of all known complete paths from node N (a homogeneous solution can provide a value in the absence of others, i.e., $F(N) = F^H(N)$). Then:

$$F(N) \;\leq\; F^*(N) \;\leq\; \overline{F}(N). \tag{6}$$

From these observations follows:

Theorem 5. If $F(N) = \overline{F}(N)$, then the feasible solution that provides $F(N)$ is optimal for node N.

The problem of determining the optimal cutting pattern consists of determining the most valuable complete path in the described *AND/OR*-graph. This path is determined as soon as the initial node is solved. A graph search strategy is a particular way of traversing a graph, or a way of enumerating its nodes that defines a method for solving the problem. The complete enumeration (or generation) of the nodes is, in general, computationally infeasible. However in practice it is enough to generate a few of them. Several rules can be devised without loss of generality in order to avoid unnecessary branchings (e.g. see the rules devised for guillotine cuttings in Herz, 1972, or Christofides and Whitlock, 1977, and rules devised for non- guillotine cutting problems in Arenales and Morabito, 1995). These rules are problem-dependent.

Theorem 4 suggests a recursive formula for solving cutting problems (*dynamic programming*):

$$F^*(N) = max\ \{F^*(N_{k1}) + F^*(N_{k2}) + \ldots,\ \text{for all } k \text{ such that } C_k(N) = (N_{k1}, N_{k2}, \ldots)\}\ .$$

This formula consists of solving a node after all its successors have been solved, and the successors are obtained after making all possible elementary cuts on the object represented by that node. The initial conditions are $F(N)$=0, for every N that represents an object that

cannot produce any items. In the above formula it should be clear that the successor *AND*-nodes $N_{k1}, N_{k2}, \ldots$ must be independent of each other. This is not the case for constrained problems and therefore that formula is not valid. In part II of this study we will discuss how the *AND/OR*-graph can be used to yield heuristic solutions for constrained cutting problems.

Dynamic programming has been frequently employed in the cutting literature since the pioneer work of Gilmore and Gomory. However, it is severely limited because of memory availability as the number of elementary cuts can be enormous leading to an untractable number of successor nodes whose values need to be kept in the computer memory. Of course, heuristics can be devised by selecting a subset of elementary cuts. This is what Beasley(1985) did by considering only larger items in order to define elementary cuts (the discretization sets), and slightly changing the formula in theorem 4 by including homogeneous solutions. In homogeneous solutions, the smaller items are then taken into account.

However, different kinds of graph searches can be employed for which the memory limitation is not so relevant, since only the best path must be kept in the memory. The search can also be reduced by using upper and lower bounds in order to prune paths that do not lead to optimal solutions, implicitly enumerating the nodes (a branch and bound method) or discarding non-promising paths producing heuristic solutions. The branch and bound method described in Morabito and Arenales (1992, 1994a) and Arenales and Morabito (1995) can be straightforwardly extended to treat general problems, in addition to the heuristics they provided.

It is still worth noting that the dimension of the problem is subtle in the *AND/OR*-graph approach, arising, of course, from the implementation when a number of parameters have to be defined in order to represent objects and cuts. This point will be introduced in Section 5, and better detailed in Part II of this study.

4 A Branch and Bound Method

In the earlier section we presented a reduction problem method to solve the cutting/packing problem, where the solution of the original problem (i.e., of determining the best cutting pattern for a given object) led to solving a number of smaller problems which could again be divided and so on. Such an approach has an AND/OR-graph representation so that a method for solving the cutting/packing problem can be designed by a particular way of transversing the graph or ,equivalently, a way of enumerating its nodes. The efficiency of this approach depends on our ability to solve the smaller problems which arise at in each node N of the graph as follows:

$$F^*(N) = \textit{maximum } v_1 y_1 + v_2 y_2 + \ldots + v_m y_m$$

$$\text{subject to } (y_1, y_2, \ldots, y_m) \in \mathbb{P}(\mathrm{N}).$$

This section presents a procedure for implicitly enumerating part of the nodes by using the bounds defined in Section 3. This is known in the literature as the Branch and Bound or Implicit Enumeration Method.

4.1 The Implicit Enumeration Method

The ordinary implicit enumeration method is based on simple principles to prune a node (Nemhauser and Wolsey, 1988):

1. Infeasibility;
2. Optimality;
3. Value dominance.

That is, a node N which represents an optimization problem can be pruned if there is no feasible solution, if its optimal solution is known, or if its maximum value is not superior to an objective function value of a known feasible solution.

Slight modifications should be done in order to apply these principles to our problem. For example, $0 \in \mathbb{P}(N)$ regardless of the object that N represents. The first principle should be changed to state that the null solution is the only one (or equivalently, $\mathbb{M}(N) = \phi$).

The bounds defined in Sections 3.2 and 3.3 can be used to prune nodes from the graph by identifying optimality or value dominance without having to solve the problems at nodes.

Other smaller modifications should be done in order to fit the implicit enumeration method to the cutting problem, since a single branching leads to different nodes which alone do not produce a solution to the preceding node. Such modifications are explained in the following.

Updating

Let $F(N)$ be the best value of known solutions at node N, which can be started with $F^H(N)$ from homogeneous solutions. The value $F(N)$ is updated as soon as a better solution is obtained. For example, if an elementary cut $C(N) = (N_1, N_2, \ldots, N_s)$ is made, then $F^H(N_1) + F^H(N_2) + \ldots + F^H(N_s)$ provides the objective function value of a feasible solution for node N, composed by various homogeneous solutions. So, if

$$F^H(N_1) + F^H(N_2) + \ldots + F^H(N_s) > F(N)$$

then $F(N)$ is redefined by:

$$F(N) \leftarrow F^H(N_1) + F^H(N_2) + \ldots + F^H(N_s)$$

After the node N is updated, all its preceding nodes can also be updated. Of course, the value of a node, e.g. N', that precedes N could have been calculated using another path, and even with an improvement of $F(N)$, its value $F(N')$ could remain the same.

Example 7. Considering Fig. 10 where circles represent nodes, the numbers inside are the node enumeration and the numbers outside their known best values. After cuts C_1 and C_2 are made the best available solution equals 12. If node 4 is branched into nodes 6 and 7 (i.e., $C_3(4) = (6, 7)$), the value of node 4 should be updated if $x > 4$. But, if $4 < x \leq 6$, the value of node 1 is the same.

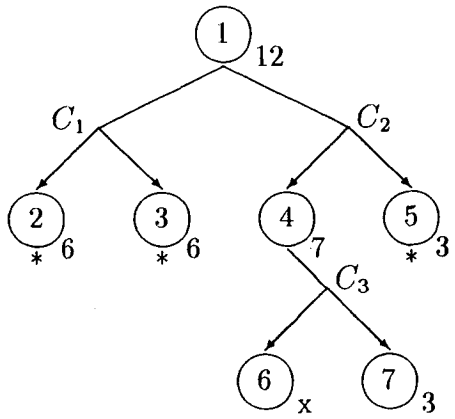


Figure 10 - Updating procedure
(*) Solved Nodes ■

The updating of preceding nodes assures that the value of the initial node $F(1)$ is always the value of the best known solution at any stage of the enumeration process.

Pruning by Optimality

If $F(N) = \overline{F}(N)$ then the feasible solution that provides $F(N)$ is optimal for node N (Theorem 5). Node N is then considered solved and no more branching of it is necessary. Note that without using bounds in order to determine a node to be solved, one should search all its successors and then choose the best solution (the optimality principle in dynamic programming, see the formula in theorem 3). If the bounds are efficient, the criterion used to solve a node may prevent an enormous number of branchings.

Pruning by Dominance

Let $C(N) = (N_1, N_2, \ldots, N_s)$ be an AND-arc pointing from N to $N_1, N_2, \ldots, N_s$. Now suppose that:

$$F(N) \geq \overline{F}(N_1) + \overline{F}(N_2) + \ldots + \overline{F}(N_s).$$

Therefore, this branching can be discarded without loss of optimality, i.e., the nodes N_1, $N_2, \ldots, N_s$ and their successors are implicitly enumerated. A particular instance of this occurs when the cut is innocuous, that is, all the generated objects are waste, and $\overline{F}(N_i) = 0, i = 1, \ldots, s$.

These remarks can be summarized in the following theorem.

Theorem 6. Without losing optimality, nodes of the graph can be pruned if any one of the following conditions holds.

1. Null Solution: $\Pi^{\prime}(N) = \{0\}$ (node N is pruned).

2. Optimality: $F(N) = \overline{F}(N)$ (node N is pruned).

3. Value dominance: $F(N) \geq \overline{F}(N_1)+\overline{F}(N_2)+\ldots+\overline{F}(N_s)$, with $C(N) = (N_1, N_2, \ldots, N_2)$ (nodes $N_1, N_2, \ldots, N_s$ are pruned).

When the initial node (that represents the object for which we are looking for the best cutting pattern) is solved, the original cutting problem is solved.

4.2 Algorithm

Before describing the algorithm, we would like to make some additional remarks.

A list L is used to indicate nodes that were generated but not yet expanded or pruned. At first it contains only the initial node representing the original object. We omitted the representations of the objects associated with the nodes, but they have to be considered in a computational implementation in order to determine F^H and F^U, as well as to identify the set of feasible elementary cuts. In the next section we shall discuss this point for some applications.

Step 1 (Initialization): $\{L\} = \{1\}$. *Node 1 represents the original object. Determine* $F(1) = F^H(1)$ *and* $\overline{F}$.

Step 2 (Termination Test): If $\{L\} = \emptyset$, *the complete path that yielded F(1) determines the optimal cutting pattern.*

Step 3 (Node Selection): Select a node N from L.

Step 4 (Node Pruning): If $F(N) = \overline{F}N$ *or there is no branches from N, then delete N from L. Go to Step 2.*

Step 5 (Branch Selection): Select any arc emerging from N, not previously select. Let $C(N) = (N_1, N_2, \ldots, N_s)$ *be the select arc.*

Step 6 (Branch Pruning): If $F(N) \geq \overline{F}(N_1) + \overline{F}(N_2) + \ldots + \overline{F}(N_s)$, *skip arc* $C(N)$ *and go to step 4.*

Step 7 (Updating): Put nodes $N_1, N_2, \ldots, N_s$ *into* $\{L\}$.

$$\textit{If: } F^H(N_1) + F^H(N_2) + \ldots + F^H(N_s) > F(N),$$
$$\textit{Then: } F(N) = F^H(N_1) + F^H(N_2) + \ldots + F^H(N_s).$$

Update the preceding nodes until the initial node 1 is reached, deleting from L all nodes that satisfy the optimality condition in Theorem 5. Go to Step 2.

Although the implicit enumeration approach can produce a remarkable reduction in the graph search, a very large number of paths to be searched may remain, which causes this approach to fail from a computational point of view.

Therefore it is necessary to design heuristics that lead to the search of parts of the graph that seem to contain the optimal solution. This is done by abandoning a large number of branchings which are considered nonpromising. Although the optimal solutions could be among the discarded solutions, the heuristic searches should be devised in such a way that good or near-optimal solutions are found and for a number of simple instances it should be able to obtain an optimal solution which can be observed from computational experiments. Moreover, heuristics should be performed in a reasonable computational time. In part II of this study we will present some heuristics that in some case studies have confirmed the above expectation.

5 Regular Problems

Unfortunately, in practice, it is not possible to use the approach presented in the previous section to treat general problems, where objects and items of any dimensions and shapes should be handled. In this section we shall see how to explore particularities in a special and practical class of cutting problems with certain regularities in the shape of objects and items, as well as in the cutting process. The simplifications due to problem regularity enable the computational implementation of AND/OR-graph approaches which have presented good results for solving instances of cutting and packing problems (see Arenales and Morabito, 1995; Morabito et al., 1992; Morabito and Arenales, 1992, 1994).

Definition 8. *(regular problem)* A cutting problem is called *regular* if the original object, the items and intermediary objects generated by the elementary cuts have limited geometric shapes characterized by a small number of parameters.

In the definition above we use "limited geometric shapes" and "small number of parameters" in order to prevent generic geometric shapes, polygons for instance, and then avoiding intermediary objects with an increasing number of parameters as the cutting process procedes. In fact, a set of geometric shapes should be defined *a priori* in agreement with elementary cuts.

Example 8. In order to better understand the meaning of Definition 8, consider the cutting problem where an equilateral triangle (object) is to cut into equilateral triangles (items). Note that just one parameter is necessary to represent each of them. Assume the elementary cuts are defined under the rule: "each single cut is parallel to one edge of the original object". Then, the objects generated are equilateral triangles, equilateral trapezoids and parallelograms (see Fig. 11). Therefore, the problem is regular and three geometric shapes are handled, where the first one, the triangular shape, needs only one parameter to be represented, and the other two geometric shapes need two parameters (of course, a third logical

parameter should be used in order to distinguish trapezoid and parallegram).

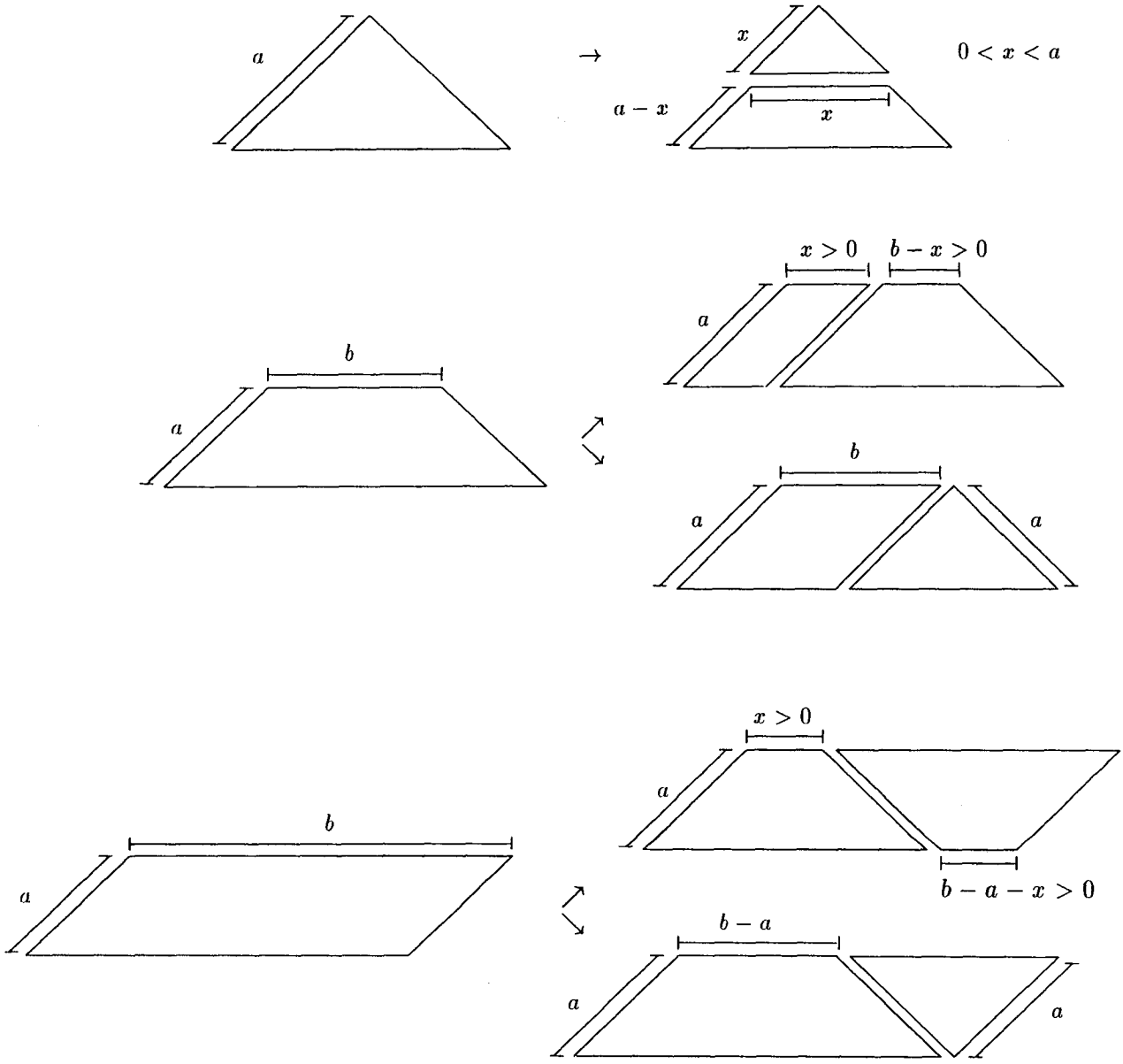


Figure 11 - Possible elementary cuts and resulting objects ■

The regularity of the problem has been implicit in the literature. For example, a two-dimensional cutting problem is concerned with rectangular- shaped items, objects and intermediary objects. Otherwise, the adjctive *irregular* is used. In general, regularity in the

literature is associated with rectangles and boxes for two and three-dimensional cutting problems, respectively. The above definition extends the regularity concept to other geometric shapes. However, note that the cutting of a rectangle into smaller rectangles would not be a regular problem if the elementary cuts were such that complicated polygons should be handled.

A particular elementary cut that often arises from practical applications, called the guillotine cut, is now defined together with the regularity concept.

Definition 9. *(Guillotine Cut)* A cut is of the *guillotine* type if it divides an object into two new intermediary objects. Specifically, if the generated objects have the same geometric shapes as the original object and items which characterize the problem regularity, the cut is then called a *regular guillotine cut.*

Example 9. In Example 4, the elementary cuts V_x and H_y are regular guillotine cuts. The guillotine cuts in Example 8 are not regular, since the object and items are triangles, and other geometric shapes arise in the cutting process. ■

Example 10. If rectangular-shaped items are to be packed on a rectangular pallet, the packing pattern in Fig. 12a can be obtained by a sequence of irregular guillotine cuts illustrated in Fig. 12b, since L-shaped geometric shapes should be handled.

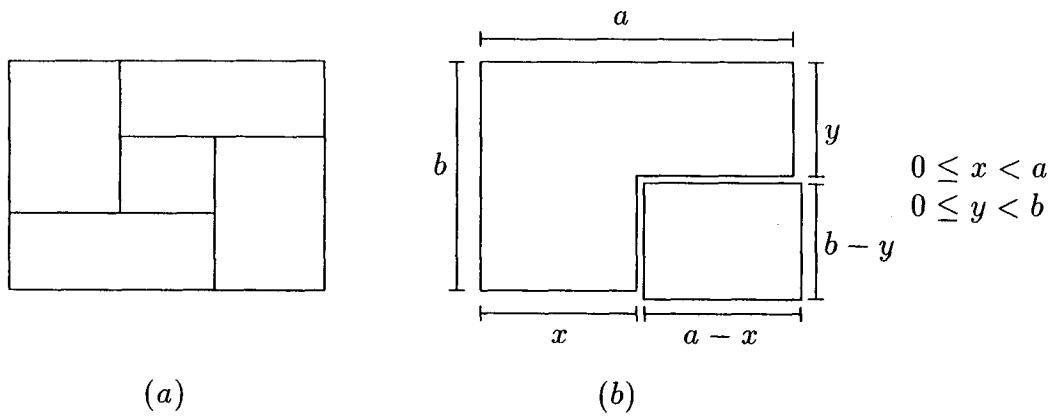


Figure 12 - (a) A cutting pattern involving only rectangles
(b) An irregular guillotine elementary cut ■

Example 11. Assume a rectangular-shaped plate is to be cut into rectangular and L-shaped items. Then, regular guillotine elementary cuts can be defined as V_x, H_y (see Example 4), or the elementary cut illustrated in Fig. 12b (note that in Example 10, it was an irregular guillotine cut since object and the items were rectangles), or the one illustrated in Fig. 13.

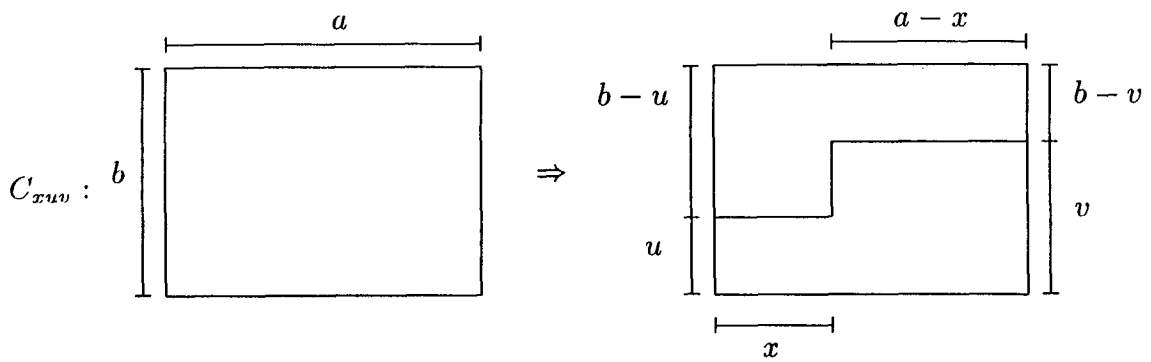


Figure 13 - (a) A kind of regular guillotine elementary cut for L-shaped items

It should be clear that a kind of C_{xuv} cut can also be devised for L-shaped objects, but restricted to avoid other geometric shapes. ■

Most of problems studied in the literature are of the regular guillotine type, and treat rectangles and boxes, in two and three dimensions, respectively. Now we show how these problems, involving rectangles and boxes, can be straightforwardly generalized for any dimension.

Consider a hyper-box as the original object: $(A_1, A_2, \ldots, A_d)$. Hyper-boxes consist of rectangles when $d = 2$, where A_1 and A_2 are the length and width, respectively. Similarly, a hyper-box is a parallelepiped when $d = 3$ or a bar in the case of $d = 1$. Let the items be the hyper-boxes:$(a_{1i}, a_{2i}, \ldots, a_{di}), i = 1, \ldots, m$. Assuming that only the geometric shape is the hyper-box shape, we need only d parameters to represent each object, which are the measure of each dimension. Let $\mathbf{o} \equiv (a_1, a_2, \ldots, a_d)$ be an intermediary object. We can define a type of regular guillotine elementary cut for each dimension:

$$C_x^k(\mathbf{o}) = (\mathbf{o}_1, \mathbf{o}_2)$$

where

$$\mathbf{o}_1 = (a_1, \ldots, x, \ldots, a_d) \quad \text{and} \quad \mathbf{o}_2 = (a_1, \ldots, a_k - x, \ldots, a_d)$$

for $k = 1, \ldots, d$ and $0 < x < a_k$.

Example 12. When $d = 2$, we have the two-dimensional guillotine cutting problem and $C_x^1 \equiv V_x$ and $C_x^2 \equiv H_y$, as defined in Example 4. ■

Theorem 7. Consider the node N representing the object $\mathbf{o} \equiv (a_1, a_2, \ldots, a_d)$. Without a loss of generality, the parameter x in C_x^k can be defined in the finite set:

$$X_k(N) = \left\{ x \text{ such that } x = \sum_{i=1}^{m} \alpha_i a_{ki} < a_i, \alpha_i, \geq 0 \text{ and integer} \right\}$$

called *discretization set* in dimension k.

In this example of a regular guillotine problem, the number of parameters necessary to represent an object coincides with the space dimension d. We conjecture that even if different geometric shapes are possible, requiring different parameters in order to represent them, the elementary cuts can still be a combination of the values of the parameters which define the items. For example, the cuts on equilateral triangles, equilateral trapezoids and parallelograms illustrated in Example 8 can be combinations of a_{1i}, where a_{1i} is the only parameter necessary to define the item i.

The discretization sets defined in Theorem 7 allow the exhibition of the finite set of elementary cuts (see Theorem 2) and therefore, they clarify the proposed approach for an important class of cutting problems. However, such sets are usually very large what makes them practically infeasible. The sizes of these sets depend on the ratios a_{ki}/A_i such that when small, they produce large scale cutting problems. In this case, heuristic solutions can be obtained by selecting subsets of the these discretizions sets (see Beasley, 1985) or imposing artificial constraints on the original cutting/packing process (see Morabito and Arenales, 1994,1995). This topic will be considered in Part II of this study.

A number of rules can be defined in order to avoid equivalent cutting patterns, such as the symmetry rule:

$$x \in X_k(N) \text{ such that } x \leq a_k/2.$$

That is, the cuts can be made on the first half of each dimension of the object. This is valid since if $C_x^k(\mathbf{o}) = (\mathbf{o}_1, \mathbf{o}_2)$ with $x \in X_k(N)$ but $x > a_k/2$, then we can push the items in the cutting pattern in the object $\mathbf{o}_2$ until they are in contact with each other, without overlapping. Let x' be the position of the left side of the leftmost item in $\mathbf{o}_2$. Therefore, $z = a_k - x' \in X_k(N)$ with $z < a_k/2$, and if we cut the object at z we are able to produce a cutting pattern equivalent to that obtained by the cut at x. This is illustrated in Fig. 14.

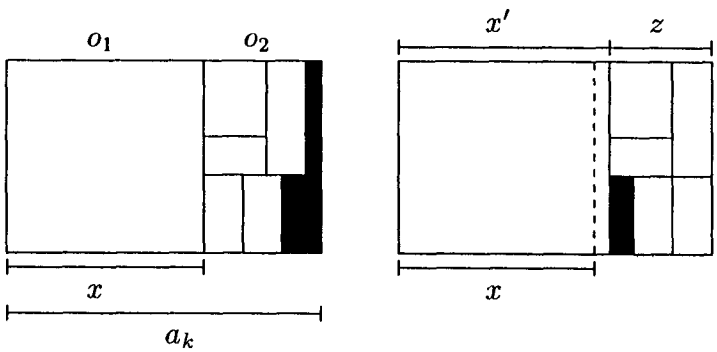


Figure 14 - Equivalent cutting patterns with $z \in X_k(N)$ and $z < a_k/2$.

6 Final Remarks

The problem of generating cutting/packing patterns was focused on in this paper, consisting of the main problem in solving cutting/ packing stock problems. A number of simple and realistic hypotheses were assumed which allowed the representation of the cutting patterns as finite complete paths in a kind of data structure called an AND/OR-graph. A number of properties were provided in order to characterize feasible and optimal solutions in preparation for devising solution methods. A straightforward extension of the ordinary branch and bound method was also given. Finally, we focused on regular problems in light of the theory presented. Throughout the paper, we gave a number of cutting/packing pattern examples treatable by the theory in order to clarify its breadth. Of course, they were not exhaustive, and several other problems of different kinds and dimensions can be handled by using this theory. Part II of this study will generalize methods and heuristics designed for specific cutting/packing problems, and compile computational experiences by using problems of different dimensions.

Acknowledgments

This research was partially supported by FAPESP and CNPq.

References

Arenales, M. and Morabito, R. (1995), *An AND/OR-Graph Approach to the Solution of Two-Dimensional Non-Guillotine Cutting Problems*, European J. Opl. Res., 84, 599-617.

Beasley, J. (1985), *Algorithms for Unconstrained Two-Dimensional Guillotine Cutting*, J. Opl. Res. Soc., 4, 297-306.

Berge, C. (1973), *Graphs and Hypergraphs*, North-Holland.

Christofides, N. and Whitlock, C. (1977), *An Algorithm for Two- Dimensional Cutting Problems*, Opn. Res., 25, 30-44.

Chvatal, V. (1983), *Linear Programming*, W.H.Freeman.

Daza, V.P.; Alvarenga, A.G. and Diego, J. (1995), *Exact Solutions for Constrained Two-Dimensional Cutting Problems*, European J. Opl. Res., 84, 633-644.

Dowsland, K. and Dowsland, W. (1992), *Packing Problems*, European J. Opl. Res., 56, 2-14.

Dyckhoff, H. (1990), *A typology of Cutting and Packing Problems*, European J. Opl. Res., 44, 145-159.

Dyckhoff, H., Kruse, H.-J., Abel, D. and Gal, T. (1985), *Trim Loss and Related Problems*, OMEGA, 13, 59-72.

Dyckhoff, H. and Finke, U. (1992), *Cutting and Packing in Production and Distribution: A Typology and Bibliography*, Physica-Verlag.

Dyckhoff, H. and Wäscher, G. (1990), *Special Issue: Cutting and Packing Problems*, European J. Opl. Res., 44(2).

Gilmore, P.C. and Gomory, R.E. (1961), *A Linear Programming Approach to the Cutting Stock Problem*, Opns. Res., 9, 849-859.

Gilmore, P.C. and Gomory, R.E. (1963), *A Linear Programming Approach to the Cutting Stock Problem, Part II*, Opns. Res., 11, 863-888.

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

Gilmore, P.C. and Gomory, R.E. (1966), *The Theory and Computations of Knapsack Functions*, Opns. Res., 14, 1045-1074.

Golden, B. (1976), *Approaches to the Cutting Stock Problem.* AIIE Transactions, 8, 265-274.

Hinxman, A.I. (1980), *The Trim-Loss and Assortment Problems: A Survey*, European J. Opl. Res., 5, 8-18.

Lasdon, L.S. (1970), *Optimization Theory for Large Systems*, Macmillan.

Martelo, S. (1994), *Special Issue: Knapsack, Packing and Cutting, Part I: One Dimensional Knapsack Problems; Part II: Multidimensional Knapsack and Cutting Stock Problems*, 32(3,4).

Martelo, S. and Toth, P. (1990), *Knapsack Problems: Algorithms and Computer Implementations*, J. Wiley.

Nemhauser, G.L. and Wolsey, L.A. (1988), *Integer and Combinatorial Optimization*, J. Wiley.

Morabito, R. and Arenales, M. (1994), *An AND/OR-Graph Approach to the Container Loading Problem*, Int. Trans. Opl. Res., 1, 59-73.

Morabito, R. and Arenales, M. (1995), *Performance of Two Heuristics to Solve Large Two-Dimensional Guillotine Cutting Problems*, INFOR, 32(2), 145-155.

Morabito, R. and Arenales, M. (1996), *Staged and Constrained Two- Dimensional Guillotine Cutting Problems: An AND/OR-Graph Approach*, European J. Opl. Res., 94(2), 548-560.

Morabito, R., Arenales, M. and Arcaro, V.F. (1992), *An AND/OR-Graph Approach for Two-Dimensional Cutting Problems*, European J. Opl. Res., 58(2), 263-271.

Nilsson, N.J. (1971), *Problem Solving Methods in Artificial Intelligence*, McGraw-Hill.

Pearl, J. (1984), *Heuristics: Intelligent Search Strategies for Computer Problem Solving*, Addison-Wesley.

Salkin, H.M. and Kluyver, C.A. (1975), *The knapsack Problem: A Survey*, Naval Research Logistics Quartely 22, 127-144.

Sweeney, P.E. and Paternoster, E.R. (1992), *Cutting and Packing Problems: A Catehorized Application-Oriented Research Bibliography*, J.Opl.Res.Soc.,43, 7, 691-706.

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