# An Efficient-FFD Method to Solve Very Large Guillotinable Cutting Stock Problem*

**David Mauricio**
Universidad Nacional Mayor de San Marcos, FISI
Av. Germán Amézaga s/n, Ciudad Universitaria, Lima01, Lima, Perú
dms_research@yahoo.com

## Abstract

In this work we introduce an efficient version of classical First Fit Decreasing (FFD for short) algorithm that is denoted by efficient-FFD and that is oriented to solve very large guillotine- cutting-stock-problem (GCSP). We show that FFD algorithm for GCSP presents $O(N^2 \log_2(N) + N^2)$ asymptotic complexity, where $N$ is the total number of items to attend. The Efficient-FFD algorithm has complexity $O(n \log_2(n) + n^2 \log_2(N/n) + N)$, where $n$ is the number of different items. Also, we show that both algorithms obtain the same solution, i.e. both algorithms have the same efficiency according to the quality of solution. The numerical experiments on instances with thousands of items confirm the theoretical results.

**KEYWORDS. Guillotine cutting stock problem, Bin packing problem, FFD algorithm.**

**Main area (MH – Metaheuristics, OC – Combinatorial Optimization)**

(*)Work partially supported by CSI-UNMSM, Perú.

## 1. Introduction

(GCSP):

> Given an unlimited rectangular boards with *L*x*A* dimension and rectangular items with dimensions $l_1 x w_1, l_2 x w_2, ..., l_n x w_n$, where $l_i \le L$, $w_i \le W \ \forall i$ and demands $d_1, d_2, ..., d_n$ respectively. The *Guillotine-Cutting-Stock Problem* consists to compute all items from the rectangular boards through straight line cut and parallel of from beside to beside and with the minimum number of boards.

GCSP is known in the literature with several names as cutting stock (Gilmore (1965), Israni (1982), Dyckhoff (1990)), bin packing (Chung (1982), Dowsland (1992)) and trim loss (Hinxman (1980), Dyckhoff (1985)). By Dyckhoff (1990) this problem can be classified as 2/V/I/M.

The main application of the GCSP is to minimize the loss obtained in cutting processes, for example wood (Morabito (1997), Garcia (1996), Venkateswarlu (1992)), paper (Harjunkoski (1996), Westernlund (1995)), glasses (Dyson (1976), Farley (1983), Madsen (1979), Canto (2010)), textile manufacturing (Farley (1988)), mats (Liton (1977), canvas Farley (1990)), etc.

Since GCSP is NP-hard problem, it is justified the development of heuristics and metaheuristics as simulated annealing (Parada (1998), Faina (1999)), AND/OR graph approach (Morabito (1992), Parada (1995)), tabu search (Lodi (1999)), genetics algorithm (Kröger (1995)), greedy algorithm FFD/ BFD (Mauricio (2002)), GRASP (Mauricio (2003), Alvarez (2008)), for a review see for example Whitwell (2004). All these approaches consider unitary demand and are not good for very large problem for example with thousands of items.

Mauricio et alt (2010) developed a fast version of First Fit Decreasing (FFD for short) algorithm (Johnson et al, 1974) to solve one-dimension cutting stock problem. Their algorithm builds a pattern by placing all possible requirements on a board, and then replicates this one as often as possible. The proposed algorithm guarantees the same quality of solution that FFD but with complexity $O(n^2 \log_2(N/n))$ rather than $O(N \log_2 N)$, where $N$ is total number of items and $n$ is the total number of different types of items. In this work we present an extension of this idea to develop a fast version of FFD algorithm to solve big instance of GCSP.

This paper is organized as follows. In the second section, it is reviewed the cutting process of Wang (1983) and Mauricio (2002). A brief review of FFD algorithm for GCSP and its complexity are presented in section 3. The proposed algorithm, its complexity and quality of solution are presented in section 4. The numerical results and conclusions are presented in section 5 and 6 respectively.

## 2. The guillotine cutting process

Several efforts have been developed to solve GCSP of which most are heuristic and meta-heuristic because this problem is NP-Hard. Wang (1983) introduced a schema to build a pattern cut as follow, select an unattended item with largest area then attached horizontally or vertically an item unattended (see Figure 1) such that the building block has an acceptable loss. In the example of Figure 1, the block on the right shows less loss and will be considered in next process.

The cutting-guillotine process of Wang is one of the most widely used to solve GCSP. We can ask some questions on Wang's process. What is the acceptable loss? Does the process ensure less loss on the board?

Another alternative to construct a pattern is described in Mauricio (2002). The alternative process considers the remaining parts generated before cutting. Thus, to meet an item from an uncut board, this item fitted into the bottom left (see Figure 2), and then you attach a piece horizontal and vertical. To meet an item from a piece horizontal (or vertical) we proceed as follows. First, it fits the item into the bottom left of the horizontal (or vertical). Second, we generate a horizontal piece and vertical piece. Third, we update the dimensions of the vertical

piece (horizontal) associated with the horizontal (vertical) piece to be cut, called "fixed piece" (see figure 3). The decision to use a horizontal or vertical piece is done so as to obtain less waste.

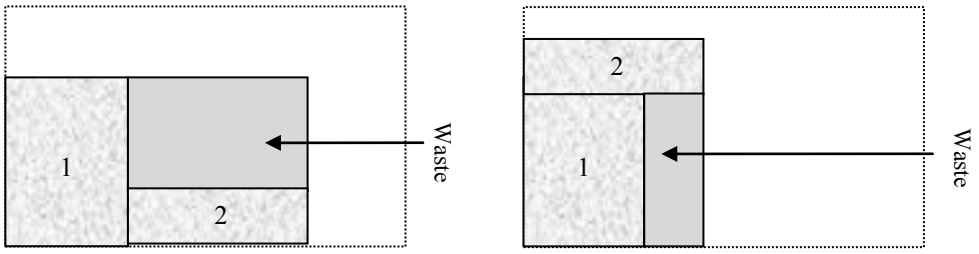
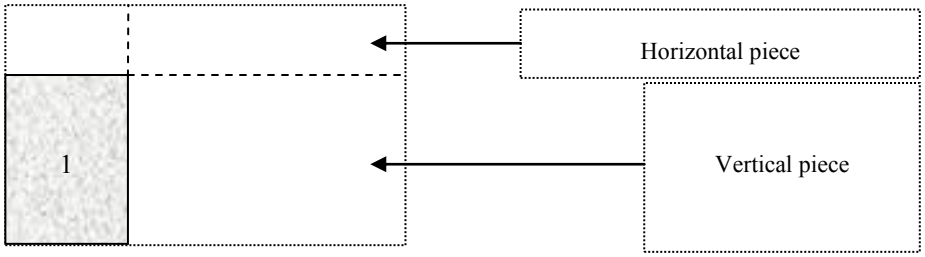



Figure 1. Wang process for constructing a pattern




(a) (b)

Figure 2. (a) Cutting a product on a new board. (b) Horizontal and vertical piece associated.

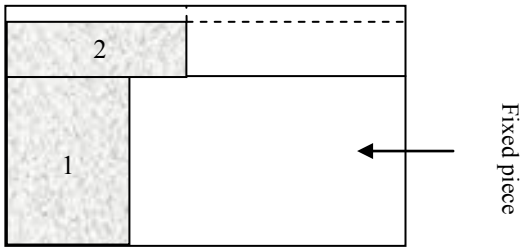



Figure 3. Cut on a horizontal piece to meet an item.

When the items have dimensions close or equal, the alternative process can generate many pieces horizontal or vertical with undesirable dimensions as seen in Figure 4.A. Mauricio (2002) proposed a way to overcome this difficulty that involves bringing these products to treat them as a unique product, see Figure4.B.

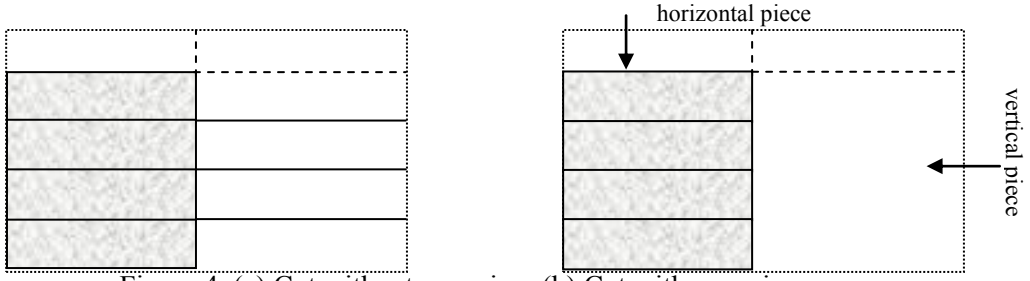



Figure 4. (a) Cut without grouping. (b) Cut with grouping.

The alternative cutting process has shown promising for solving GCSP with unit demands; see for example Cevallos (2010), Guzman (2010). In the remainder of this paper we will consider the alternative cutting process.

## 3. FFD Algorithm

All algorithms developed to solve GCSP only considered unit demand, i.e. $d_j = 1 \ \forall j$. An alternative to overcome this difficulty is to transform the GCSP with not unit demand to a GCSP problem with unit demand. This transformation is easy; it is enough to replicate the items as often as directed by your demand.

Denote by GCSP-U the transformed problem of GCSP with unitary demands. Let $N := \sum_{j=1}^{n} d_j$ and consider $\tilde{l}_i = l_1,\ \tilde{w}_i = w_1 \ \forall i = 1,...,d_1,\ \tilde{l}_i = l_k,\ \tilde{w}_i = w_k \ \forall i = \sum_{j=1}^{k-1} d_j + 1, \sum_{j=1}^{k-1} d_j + 2,...,$ $\sum_{j=1}^{k} d_j \ \forall k = 2,...,n$

The FFD algorithm is defined as follows. First, we order the items so that $\tilde{l}_1\tilde{w}_1 \geq \tilde{l}_2\tilde{w}_2 \geq ... \geq \tilde{l}_N\tilde{w}_N$. We then proceed to pack the items in the order, starting with first item $\tilde{I}_1 = (\tilde{l}_1, \tilde{w}_1)$, which we place in the first board $B^1$ through the guillotine cutting process (see section 2). In general the item $\tilde{I}_k = (\tilde{l}_k, \tilde{w}_k)$ is placed into first board that has room for it, i.e. we find the smallest *i* such that $\tilde{I}_k$ can be obtained by guillotine cutting process from $B^i$ board.

We denote respectively by $H^i, V^i, F^i$, the set of horizontal vertical and fixed pieces associated to board $B^i$, then we have FFD algorithm as follow.

**FFD- Algorithm**

1. InputInstance $\left(N, \tilde{l}_1, \tilde{w}_1, \tilde{l}_2, \tilde{w}_2, ..., \tilde{l}_N, \tilde{w}_N, L, W\right)$
2. Sort $\tilde{I}_k = (\tilde{l}_k, \tilde{a}_k)$ such that: $\tilde{l}_1\tilde{w}_1 \geq \tilde{l}_2\tilde{w}_2 \geq ... \geq \tilde{l}_N\tilde{w}_N$
3. $H^1 := V^1 := \varnothing; \quad F^1 := B^1 = (L,W); \quad m := 0;$
4. For $k:=1,..,N$
4.1 $\tilde{I}_k := (\tilde{l}_k, \tilde{w}_k);$
4.2 $i := Min\left\{j \in \{1,2,...,m+1\} : H^j \cup V^j \cup F^j \ include \ \tilde{I}_k\right\};$
4.3 If $i = m+1$
4.4 Then Cut-on-a-new-board($\tilde{I}_k, B^{m+1}$), $m:=m+1$, $F^{m+1} := B^{m+1} = (L,W)$
4.5 Else Cut-on-an-used-board($\tilde{I}_k, B^i$);
4.6 end-if;
4.7 end-for;
5. Return($m, B^i \ \forall i$)

Figure 5. FFD algorithm to solve GCSP-U

Note that *m* is the number of boards required by the algorithm FFD to address all items. Note also that when *i=m+1* (step 4.3) so we find on new board and therefore the Cut-on-a-new-board procedure (step 4.4) is activated else the Cut-on-a-used-board (step 4.5) is activated. Also, theses procedures are defined in guillotine-cutting-process. The Cutting-on-a-new-board procedure refers to cut on a non-used board to address an item. The Cutting-on-a-used-board procedure refers to cut a piece (horizontal or vertical) associated to used-board to address an item.

***Theorem 1.*** The complexity of FFD algorithm to solve GCSP-U is $O(N\log_2(N) + N^2)$

***Proof:*** Note the complexity of FFD algorithm is given by the complexity of step2 and steps 4-4.5. Let $T$ the complexity of steps 4-4.7 then the complexity of FFD algorithm is

$$O(N\log_2(N)+T) \quad (1)$$

Now let us calculate the complexity of $T$. Note that complexity of procedures Cut-on-a-new-board and Cut-on-an-used-board is $O(1)$ for all iteration. Therefore $T$ depend on the effort to verify if a item is included in a used board (step 4.2), more precisely whether an item can be included in any pieces (horizontal, vertical or fixed), i.e. the total number of pieces horizontal vertical and fixed generated, because to know if an item fits into a piece has complexity $O(1)$.

Note that when is realized a cut on horizontal or vertical piece this piece is removed and is generated in worse case a horizontal vertical and fixed piece, but when is realized a cut on fixed piece this piece is removed and is generated in worse case a horizontal and vertical piece. The next table illustrates this observation

| Cut on | Number and type of pieces generated | | | Number of pieces removed |
|---|---|---|---|---|
| | Horizontal | Vertical | fixed | |
| Horizontal | 1 | 1 | 1 | 1 |
| Vertical | 1 | 1 | 1 | 1 |
| Fixed | 1 | 1 | 0 | 1 |

Table 1. Number of pieces generated when a cut is realized

Let $h_k, v_k, f_k$ the number of cuts horizontal vertical and fixed respectively realized until the iteration $k$ then the total number of pieces horizontal vertical and fixed generated until $k$ is given by:

$$2h_k + 2v_k + f_k \quad (2)$$

By the other hand, the number of cuts is equal to number of items, i.e.

$$k = h_k + v_k + f_k \quad (3)$$

From (2) and (3), the cost to compute step 4.2 for each iteration $k$ is given by:

$$O(2h_k + 2v_k + f_k) = O(h_k + v_k + f_k) = O(k) \quad (4)$$

Since $k = 1,...,N$ then from (5) $T$ is given by:

$$T = O(N^2) \quad (5)$$

Finally, from (1) and (5) we have shown the theorem □

The complexity of FFD algorithm can be improved. Note that each item can be represented by a pair of points. Thus, a way to improved the complexity is to record a piece with two distant points for each board, then verify if an item is included in some pieces of board (step 4.2) is reduced to verify if an item is include in its piece with two distant points, and this can be compute in $O(1)$. Therefore, it is possible to develop a version of FFD of complexity $O(N\log_2(N)+mN)$

## 4. An Efficient-FFD Algorithm

All algorithms developed to solve GCSP only considered unit demand, i.e. $d_j = 1\ \forall j$. We consider GCSP in its original form, i.e. we develop an algorithm to solve GCSP with no-unitary demand. Our proposed is an extension of Efficient-FFD algorithm for cutting stock problem (one-dimension) to solve GCSP (two-dimension) and is called by the same name. The approach proposed is defined as follows. First, we order the items so that $l_1a_1 \geq l_2a_2 \geq ... \geq l_na_n$. Second, an iterative process is executed until to meet all demand for all items. The iterative process consist tree phases. In the first phase, we construct a cutting pattern as follows, pack the item that has non zero demand and in order, and we place this item many times as possible

without exceeding the demand and in the same board. We repeat this process with the rest of the items that have non-zero demand in the order of arrangement and on the same board. In the second phase, we replicate the pattern built many times permitted by demand for products that have non-zero demand. In the third phase, we proceed to update the demand of all packaged items.

As the algorithm FFD, we denote respectively by $H^i, V^i, F^i$, the set of horizontal vertical and fixed pieces associated to board $B^i$. Follows the Efficient-FFD algorithm.

**Efficient-FFD Algorithm**

1. InputInstance $(n, l_1, w_1, l_2, w_2, ..., l_n, w_n, L, W)$
2. Sort $I_k = (l_k, w_k)$ such that: $l_1 w_1 \geq l_2 w_2 \geq ... \geq l_n w_n$
3. $H^1 := V^1 := F^1 = \varnothing$; $m := 0$;
4. While $\sum_{i=1}^{n} d_i > 0$

/* construction of a pattern

4.1 $m:=m+1$; $B^m = (L, W)$; $x_i := 0 \quad \forall i = 1,...,n$;

4.2 For $k:=1,..,n$

4.3 $I_k := (l_k, w_k)$;

4.4 While $x_k < d_k$ and $H^m \cup V^m \cup F^m$ *include* $I_k$

4.5 Cut-on-a-board($I_k, B^m$)

4.6 $x_k := x_k + 1$

4.7 end-while

4.8 end-for

/* replication of the pattern

4.9 $t_m := Min_{1 \leq k \leq n} \{ \lfloor d_k / x_k \rfloor \text{ such that } x_k > 0 \}$;

/* updating of demands

4.10 $d_i := d_i - x_i t_m \quad \forall i = 1,...,n$

4.11 end-while

5. Return($m, B^j, t_j \quad \forall j = 1,..., m$)

Figure 6. Efficient-FFD algorithm to solve GCSP

Note that the process of building a pattern is given in steps 4.1-4.9. The cut-on-board procedure is defined by the procedures cut-on-a-new-board and cut-on-an-used-board both defined in previous section. The replication of the pattern and updating of the demands are given by the steps 4.10 and 4.11 respectively.

The following theorem states that the complexity of Efficient-FFD algorithm (EFFD for short) is less than complexity of FFD algorithm.

***Theorem 2.*** The complexity of EFFD algorithm to solve GCSP is $O(n \log_2(n) + n^2 \log_2(N/n) + N)$

***Proof:*** Note the complexity of Efficient-FFD algorithm is given by the complexity of step 2 and steps 4-4.11. Let $T$ the complexity of steps 4-4.11 then the complexity of Efficient-FFD algorithm is

$$O(n\log_2(n)+T) \quad (6)$$

Now let us calculate the complexity of *T*. Since the complexity of steps 4.9 (replicate the pattern phase) and 4.10 (update the demand phase) is $O(n)$ then the complexity of steps 4.1-4.11 is dominated by complexity of the construction phase of a pattern. Next, we compute the complexity of the construction phase of a pattern, that is equivalent to the complexity of steps 4.2-4.8 plus $O(n)$.

Let $x_k^j$ the number of piece $I_k$ in the board $B^j$ (7)

As seen in the previous proof, the complexity of Cut-on-a-board procedure (step 4.5) and the complexity of check if $H^j \cup V^j \cup F^j$ includes $I_k$ (step 4.5) is $O(1)$, then by (7) the complexity of steps 4.45-4.7 is $O(x_k^j)$. Thus, the complexity of steps 4.2-4.8 is $O(\sum_{k=1}^{n} x_k^j)$ and therefore The complexity of steps 4.1-4.11 is $O(n+\sum_{k=1}^{n} x_k^j)$ (8)

By the other hand, the iteration number of step 4 depends of the convergence to zero of the expression $\sum_{i=1}^{n} d_i$, where each remaining of demand $d_i$ is decreased in some iteration after forming a pattern $B^j$ and their replication. So, we concluded that exist an items $I_k$ with $d_k > 0$ and a pattern $B^j$ such that all or part of this demand is covered by this pattern and their replications $t_j$, i.e.

$$0 < x_k^j \le d_k \quad (9)$$

$$t_j := \lfloor d_k / x_k^j \rfloor = Min_{1\le r\le n}\{\lfloor d_r / x_r^i \rfloor \text{ such that } x_r > 0\}; \quad (10)$$

This imply $t_j \ge 1$, because from (9) we have $d_k / x_k^j \ge 1$. This means that the number of repetitions of the $B^j$ pattern is always greater than zero. With this, the expression 4.10 $d_k := d_k - x_k^j t_j$, will be expressed as

$$d_k := d_k - \lfloor d_k / x_k^j \rfloor x_k^j \quad (11)$$

From (9) we have two cases. First case: if $x_k^j = d_k$, by using (11), we have $d_k := d_k - \lfloor d_k / d_k \rfloor d_k = 0$. This means that all items of type $I_k$ are attended. The same occurs when $x_k^j = 1$. Second case: if $x_k^j < d_k$, then from (9) $d_k$ is update by the remaining of the division of $d_k$ by a positive and not unitary number $x_k^j$. This means that in the worst case, in each iteration (steps 4-4.11), as minimum one demand of all requirements is updated by a positive and non-unitary number. In this last case we have two situations: If $x_k^j < d_k / 2$ then $d_k$ will be updated by a value less than $d_k / 2$; and If $x_k^j \ge d_k / 2$ then $d_k$ will be updated by a value less than $d_k / 2$. This means that in the worst case the series of the values of $d_k$ will be dominated by:

$$d_k/2^1, d_k/2^2, ..., d_k/2^{u_k} \quad (12)$$

Where $u_k$ is the number of times is updated $d_k$. Therefore, from the previous expression, in worst case, we have:

$$d_k/2^{u_k} = 1 \quad (13)$$

That is, after iterations $u_k$, the demand of the item $I_k$ is 1. We can compute $u_k$ from (13), by applying logarithms, as

$$u_k = \log_2 d_k \tag{14}$$

On the other hand, the stopping condition in step 4 controls the iteration to attend all demand. Therefore, in the worse case the number of total iterations (step 4.-4.11) is given by:

$$\sum_{k=1}^{n} u_k = \sum_{k=1}^{n} \log_2 d_k \tag{15}$$

Using logarithmic properties to the expression above, we have:

$$\sum_{k=1}^{n} \log_2 d_k = \log_2 \prod_{k=1}^{n} d_k \leq \log_2 \left( \frac{\sum_{k=1}^{n} d_k}{n} \right)^n \tag{16}$$

As $N = \sum_{k=1}^{n} d_n$ then from (15) and (16), the total of iterations of step 4 is dominated as:

$$\sum_{k=1}^{n} u_k \leq n \log_2 (N/n) \tag{17}$$

Note that the total of iterations of step 4 is $m$, because in each iteration a pattern is built. Therefore, from (8) and (17) we have:

$$T = O(n^2 \log_2 (N/n) + \sum_{i=1}^{m} \sum_{k=1}^{n} x_k^i) \tag{18}$$

Note that $\sum_{i=1}^{m} \sum_{k=1}^{n} x_k^i \leq \sum_{i=1}^{m} t_i \sum_{k=1}^{n} x_k^i = N$, because $N$ is the total items and the number of replicates is always at least one. Therefore from (18) we have:

$$T = O(n^2 \log_2 (N/n) + N) \tag{19}$$

Finally, from (6) and (19) we have shown the theorem □

Note that the EFFD algorithm is more efficient than original FFD algorithm when the number of different items is small compared to the total items. For example, for $N = 1'048{,}576$ total items and $n = 16$ different items in the worst case, the EFFD algorithm requires 1'052736 operations but the original FFD algorithm will require a number of iterations of the order of 120 digits, i.e. it is impossible to solve this instance with FFD algorithm.

Now, we analyze the quality of calculated solution by proposed algorithm. The following theorem says that the quality of the solution obtained by EFFD and FFD are the same.

***Theorem 3.*** Given $M^{FFD}, M^{EFFD}$ the number of board required by FFD and EFFD algorithms to solve GCSP, respectively. Then,

$$M^{FFD} = M^{EFFD}$$

***Proof:*** Without loss of generality, we consider that all demands of CGSP are unitary, for this, it is sufficient replicate the items as times as their demand.

Denote by $B_i^{FFD}$, $B_i^{EFFD}$ the patterns generated in the iteration $i$ $(\forall i \leq M^{FFD}, M^{EFFD})$ by FFD and EFFD algorithms respectively

Suppose that $M^{FFD} \neq M^{EFFD}$ (20)

Then there is $j \leq Min\{M^{FFD}, M^{EFFD}\}$ such that

$$B_j^{FFD} \neq B_j^{EFFD} \tag{21}$$

$$B_i^{FFD} = B_i^{EFFD} \quad \forall i < j \tag{22}$$

Let $B_{jh}^{FFD}$, $B_{jh}^{EFFD}$ the $h$ item placed in $B_j^{FFD}$, $B_j^{EFFD}$ respectively, then from (21) there is $v$ such that $v \leq Min\{|B_j^{FFD}|, |B_j^{EFFD}|\}$ and there is an item $I_k$ for some $k$, $1 \leq k \leq N$ such that

$$B_{jv}^{FFD} = I_k \neq B_{jv}^{EFFD} \quad (23)$$

$$B_{jh}^{FFD} = B_{jh}^{EFFD} \qquad \forall h \text{ such that } 1 \leq h < v \quad (24)$$

This means that first $v-1$ items are the same for both patterns. In this situation these horizontal, vertical and fixed set are the same for both patterns, i.e.

$$H_{jv-1}^{FFD} = H_{jv-1}^{EFFD}, \quad V_{jv-1}^{FFD} = V_{jv-1}^{EFFD}, \quad F_{jv-1}^{FFD} = F_{jv-1}^{EFFD} \quad (25)$$

Where $H_{jv-1}^{FFD}, V_{jv-1}^{FFD}, F_{jv-1}^{FFD}$ and $H_{jv-1}^{EFFD}, V_{jv-1}^{EFFD}, F_{jv-1}^{EFFD}$ are the horizontal vertical and fixed set for $B_{jv-1}^{FFD}$ and $B_{jv-1}^{EFFD}$ respectively.

Then, from (23) and step 4.2 of FFD algorithm we have

$$I_k \subseteq (H_{jv-1}^{FFD} \cup V_{jv-1}^{FFD} \cup F_{jv-1}^{FFD}) \quad (26)$$

From (25) and (26):

$$I_k \subseteq (H_{jv-1}^{EFFD} \cup V_{jv-1}^{EFFD} \cup F_{jv-1}^{EFFD}) \quad (27)$$

From (27) and steps 4.4-4.7 of EFFD algorithm we have:

$$B_{jv}^{EFFD} = I_k \quad (28)$$

This last result contradicts the affirmation given in (23) and therefore the supposed given in (20) is false. □

The theorems 1, 2 and 3 show that for guillotine-cutting-stock very large problem is possible to develop a fast version of FFD algorithm without sacrificing the quality of solution. To continue we show some numerical results that confirm this assertion.

## 5. Test Results

We observe the results of some set of input data defined by Items $I_k$ and their dimension $(l_k, w_k)$ and demand $d_k$, number of items type *n* and the dimension of board $(L,W)$. We use the FFD and EFFD algorithm implemented on Java IDE NetBeans 6.9 to compare the results, running in PC core duo 2.1Ghz, 3GB RAM Windows 7.

The results of same efficiency of FFD and EFFD algorithm, as formalized in theorem 3, is presented in Table 2, where the Factor is multiplied to each demand that define the total number of demand *N*. So, the first line is referred to instance with original demands, the second line is referred to instance with original demand multiplied by 6 for each items, etc. The both method obtain the same result in number of board used and percent lost.

Items: (1,1), (2,1), (2,2),(3,1),(3,2),(3,3)(4,1),(4,2),(5,1),(5,2),(6,2),(7,1),(7,2),(10,1)
Demands: 6 21 6 15 22 13 3 8 5 2 1 7 1 1
*N*=111, *n*=14, *L*=10, *W*=5

| Factor | *N* | FFD | | | EFFD | | |
|---|---|---|---|---|---|---|---|
| | | boards used | milliseconds | % lost | boards used | milliseconds | % lost |
| 1 | 111 | 12 | 350 | 5 | 12 | 352 | 5 |
| 6 | 666 | 69 | 2217 | 1 | 69 | 1120 | 1 |
| 11 | 1221 | 126 | 6782 | 0 | 126 | 1034 | 0 |
| 16 | 1776 | 184 | 12921 | 1 | 184 | 1684 | 1 |
| 21 | 2331 | 241 | 20447 | 0 | 241 | 1666 | 0 |
| 26 | 2886 | 298 | 30551 | 0 | 298 | 1784 | 0 |
| 31 | 3441 | 355 | 40961 | 0 | 355 | 1232 | 0 |
| 36 | 3996 | 412 | 54655 | 0 | 412 | 1700 | 0 |
| 41 | 4551 | 470 | 71621 | 0 | 470 | 1093 | 0 |
| 46 | 5106 | 527 | 89517 | 0 | 527 | 1016 | 0 |

Table 2. Similar results of boards used and percent lost obtained by FFD and EFFD algorithms

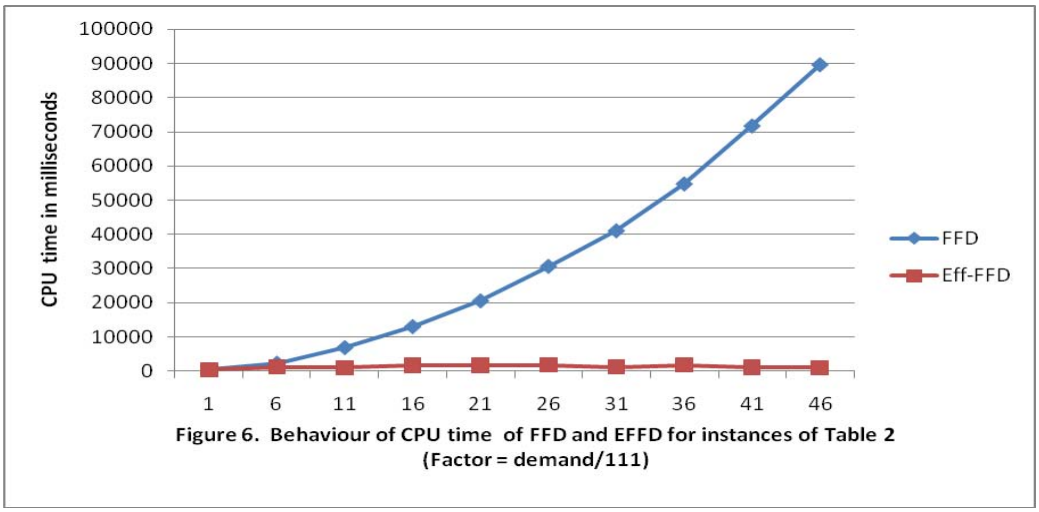



Figure 6. Behaviour of CPU time of FFD and EFFD for instances of Table 2 (Factor = demand/111)

In Table 3, another instance is used to determine the time consumed by the EFFD and FFD algorithms implemented. The Factor is increased by 1 and multiplied to initial demanded number for each executions, generating the total demanded number *N*. We observe that time consumed, getting in milliseconds, for EFFD methods is very low than the FFD method. The Figure 7 illustrates, by using Table 3, the behavior of the time consumed by both methods.

| Items: (50,30), (22,60), (13,9), (5,4), (10,9), (8,6), (5,6), (5.55), (6.5,4.5), (2.5,2) | | | |
|---|---|---|---|
| Demands: 91 88 109 82 105 93 109 116 94 113 | | | |
| Total demanded | | Time consumed in milliseconds | |
| Factor | *N* | FFD | EFFD |
| 1 | 1000 | 57 | 4 |
| 2 | 2000 | 79 | 21 |
| 3 | 3000 | 129 | 8 |
| 4 | 4000 | 163 | 4 |
| 5 | 5000 | 182 | 21 |
| 6 | 6000 | 219 | 7 |
| 7 | 7000 | 276 | 5 |
| 8 | 8000 | 364 | 18 |
| 9 | 9000 | 381 | 7 |
| 10 | 10,000 | 460 | 7 |

Table 3. Time consumed by FFD and EFFD methods for demands increasing in geometric proportion defined by Factor

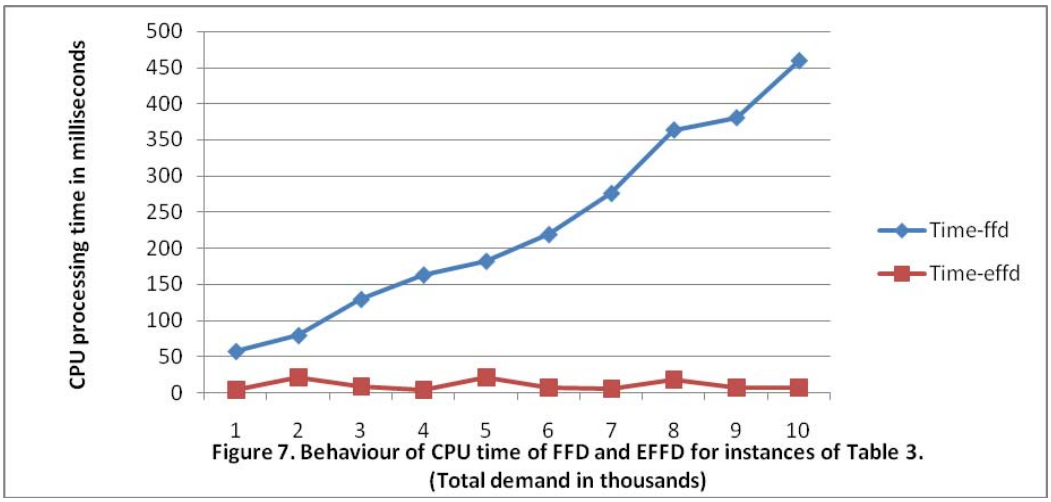



Figure 7. Behaviour of CPU time of FFD and EFFD for instances of Table 3. (Total demand in thousands)

## 6. Conclusion

We introduce a fast version of $O(n\log_2(n)+n^2\log_2(N/n)+N)$ complexity of the successful FFD algorithm to solve Guillotine Cutting Stock Problem (GCSP) with non-unitary demands, where $N$ is the total requirements demanded and $n$ the total of different size of pieces. The quality efficient analysis of our proposed method preserves the same quality efficient of original FFD algorithm. The proposed method is almost constant in time processing when the demand size varies.

## Acknowledgements

The author is grateful to Kenny Cáceres by implementing the algorithm to generate the numerical tests.

## References

**Alvarez-Valdes R.**, Parreño F., and Tamarit J.M., (2008), Reactive GRASP for the strippacking problem. Computers and Operations Research, 35(4):1065–1083.

**Cevallos J**., Algoritmo GRASP de corte de guillotina 2D con agrupamiento y rotación. 2010. Master Thesis, Universidad Inca Garcilaso de la Vega. (2010).

**Canto, N.C.F**., Costa F., Sassi R.J., (2010), A genetic algorithm for solving the two-dimensional cutting in glass sheets problem. Proceeding of 5th Iberian Conference on Information Systems and Technologies (CISTI).

**Chung F.K.R**., Garey M.R., Johnson D.S., (1982), On packing two-dimensional bins, SIAM Journal on Algebraic and Discrete Methods, 3 (1) 66-76.

**Dowsland K.A**., Dowsland W.B., (1992), Packing problems, European Journal of Operation Research. 56 (1) 2-14.

**Dyckhoff H**., Kruse H.J., Abel D., Gal T., (1985), Trim loss and related problems. Omega 13 59-72.

**Dyckhoff H.,** (1990), A typology of cutting and packing problems. European Journal of Operation Research 44 145-159.

**Dyson R.G**., Gregory A.S., (1976), The cutting stock problem in the flat glass industry. Operational Research Quartely 25 41-53.

**Farley A.A**., (1983), Trim loss pattern rearrangement and its relevance to the flat-glass industry. European Journal of Operation Research 14, 386-392.

**Farley A.A**., (1988), Mathematical programming model for cutting stock problems in the clothing industry. Journal of the Operation Operation Research Society 39 41-53.

**Farley A.A.,** (1990), The cutting stock problem in the canvas industry. European Journal of Operation Research 44 247-255.

**Faina L.,** (1999), An application of simulated anneling to the cutting stock problem, European jounal of Operation Research. 114, 542-556.

**Garcia V.,** (1996), Otimizacao de padroes de corte de chapas de fibra de madeiran reconstituida. Dissertation, Deapartamento de Engenharia de Producao, Universidade Federal de Sao Carlos, Brazil.

**Gilmore P**., and Gomory R., (1965), Multistage cutting problems of two and more dimensions. Operation Research 13 94-119.

**Guzman J.,** Sistemas de optimización de cortes de guillotina en 2D basado en el algoritmo GRASP BFD reactivo con 2 parámetros de relajación. Enegineering Thesis, Universidad Nacional Mayor de San Marcos, 2010.

**Harjunkoski I.,** (1996), Ewsternlund T., Isaksoon J., Skrifvars H., Different formulations for solving trim-loss problems in a paper converting mill with ILP. ESCAPE 6.

**Hinxman A.I**., (1980), The trim-loss and assortment problems: a survey. European journal of Operation Research, 5, 8-18.

**Johnson D.S.,** Garey M. R., Graham R. L., A. Demers and J. D. Ullman., (1974), Worst Case Performance Bounds for Simple One-Dimensional Packing Algorithms, SIAM J. Computing, 3, 299-325.

**Israni S.,** Sanders J.L., Two dimensional cutting stock problem research: A review and new rectangular layout algorithm. Journal of Manufacturing Systems 1 (1982) 169-182.

**Kröger B.,** Guillotinable bin packing: a genetic approach. European Journal of Operation Research 84 (1995) 645- 661.

**Liton C.D**., A frecuency approach to the one dimensional cutting problem for carpet rolls. Operation Researchn Quartelly 28 (1977) 927-938.

**Lodi A.,** Martello S., Vigo D., Approximation algorithms for the oriented two-dimensional bin packing problem, European Journal of Operation Research. 112 (1999) 158-166.

**Madsen O.G.B.,** Glass cutting in small firm. Mathematical Programming 17 (1979) 85-90.

**Mauricio D.,** Delgadillo R., Algoritmos FFD y BFD para resolver el problema de cortes de Guillotina. Technical Report UPG-FISI/2002-01, Universidad Nacional Mayor de San Marcos. (2002), Lima, Perú.

**Mauricio D.,** Algoritmos GRASP para el Problema de Cortes. D. Mauricio. Relatorio Técnico UPG-FISI, Universidad Nacional Mayor de San Marcos, (2003). Lima, Perú.

**Mauricio D.** Rivera L. Maculan N., (2010), An efficient FFD method to solve the one dimensional stock cutting problem. *GEST Int'l Trans.Computer Science and Engr., Vol. 61, (1) 25-36.*

**Morabito R.N.,** Arenales M.N., Arcaro V.F., (1992), An And-Or-Graph approach for two dimensional cutting problems. European Journal of Operation Research, 58, 263-271.

**Morabito R.,** Garcia V., (1997), The cutting stock problem in hardboard industry: a case study. Computer Operation Research, 25 6, 469-485.

**Parada V.,** Gómes A., De Diego J., (1995), Exact solutions for constrained two-dimensional cutticng stock problems. European Journal of Operation Research, 84, 633- 644.

**Parada V.,** Sepúlveda M., Solar M., Gómes A., (1998), Solution for the constrained guillotine cutting problem by simulated annealing. Computer Operation Research, 25, 1 37 – 47.

**Venkateswarlu P.,** Martyn C.W., (1992),The trim loss problem in a wooden drum industry, OR-92 Proceeding of the Convention of Operation Research Society of India.

**Wang, P.Y.** (1983), Two algorithms for constrained two-dimensional cutting stock problems, Operations Research 3l, 573-58

**Westernlund T.,** Isaksoon J., Harjunkoski I., Solving a production optimization problem in the paper industry. Report 95-146A, Process Desing Laboratory, Abo Akademi University (1995).

**Whitwell G.,** Novel Heuristic and metaheuristic approaches to guillotine packing, PhD Thesis, University of Nottingham, 2004.