Two Dimensional Strip Packing with Unloading Constraints [1]

Jefferson L. M. da Silveira [2], Eduardo C. Xavier [3] and Flávio K. Miyazawa [4]

Institute of Computing
University of Campinas - UNICAMP
Campinas, Brazil

Abstract

In this paper we present approximation algorithms for the two dimensional strip packing problem with unloading constraints. In this problem, we are given a strip S of width 1 and unbounded height, and n items of C different classes, each item a_i with height $h(a_i)$, width $w(a_i)$ and class $c(a_i)$. As in the strip packing problem, we have to pack all items minimizing the used height, but now we have the additional constraint that items of higher classes cannot block the way out of lower classes items. In all problems but one we assume that orthogonal rotation of the items is allowed. For the case in which horizontal and vertical movements to remove the items are allowed, we design an algorithm whose asymptotic performance bound is 3. For the case in which only vertical movements are allowed, we design a bin packing based algorithm with asymptotic approximation ratio of 5.745. Moreover, we also design approximation algorithms for restricted cases of both versions of the problem. These problems have practical applications on routing problems with loading/unloading constraints.

Keywords: Strip Packing Problem, Approximation Algorithms, Unloading/loading Constraints

1 Introduction

In this paper we study two variants of the strip packing problem with unloading constraint, that are generalizations of the well known NP-Hard strip packing problem. These problems arise in operations research transportation problems, where items are delivered along a route. In these problems, it is necessary to consider the order in which items are packed in a vehicle in order to minimize the effort while unloading it [**?**,**?**,**?**,**?**].

In the strip packing problem with unloading constraint (SPU), we are given a strip S of width 1 and unbounded height, and n items of C different classes, each item a_i with height $h(a_i)$, width $w(a_i)$ and class $c(a_i)$. A packing is a feasible solution, if items do not intersect, all of them are packed inside the strip, and there is an order to unload the items, such that no item of larger class blocks the way out of other items. The class values c represent the order in which items must be removed. While removing one item, only this item can be moved and only in the available free space of the strip. Since items are removed in increasing order of values c, we can assume that, in a feasible packing, when removing item a_i, only items a_j with class $c(a_j) \geq c(a_i)$ are still packed. The other items a_k with $c(a_k) < c(a_i)$ should be removed previously. In one version of the problem, denoted by SPU_v, only vertical movements are allowed while removing one item. In this case, if an item a_i is packed in (x_i, y_i) (i.e. its bottom left corner is placed at this position) and a_j is packed in (x_j, y_j) and $c(a_j) > c(a_i)$ then either $x_j + w(a_j) \leq x_i$ or $x_i + w(a_i) \leq x_j$ or $y_j + h(a_j) \leq y_i$. These constraints ensure that the item a_j is not blocking a_i during its removal while using only vertical movements (see Fig. 1, parts c and d). The cost of a feasible solution is the height of the used area of the strip.

We also consider the variant (denoted by SPU_{vh}) in which one horizontal movement and then one vertical movement are allowed while removing an item (see Fig 1, parts a and b). In our algorithms we assume that orthogonal rotation of the items is allowed unless it is said otherwise.

Given an algorithm $\mathcal{A}$ to SPU_v, and an instance I, we denote by $\mathcal{A}(I)$ the cost of the solution computed by $\mathcal{A}$ over the instance I. We denote by $OPT(I)$ the height used by an optimum packing of I. The proposed algorithm to the SPU_v problem is asymptotically bounded, thus it satisfy $\mathcal{A}(I) \leq \alpha OPT(I) + \beta$,

[1] This research was supported by CNPQ and FAPESP.
[2] Email: jmoises@ic.unicamp.br
[3] Email: ecx@ic.unicamp.br
[4] Email: fkm@ic.unicamp.br

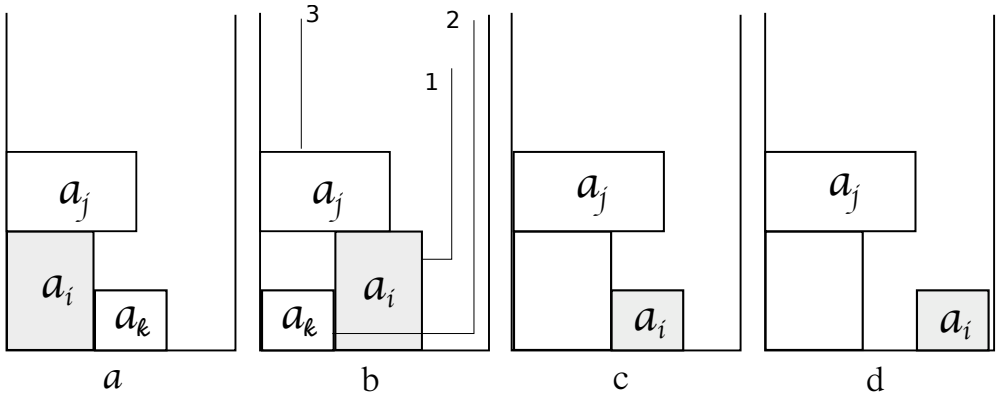

Fig. 1. Suppose that $c(a_j) > c(a_k) > c(a_i)$. (a): An infeasible solution to SPU_{vh}, because a_k and a_j are blocking a_i. (b): A feasible placement to SPU_{vh}, where the items can be removed in order 1, 2 and 3. (c): This placement is not a feasible solution to SPU_v, because a_j is blocking a_i. (d): A feasible placement to SPU_v, where the items can be removed in order.

where β is a constant and α is the asymptotic approximation ratio.

In [**?**], Azar and Epstein proposed an online 4-competitive algorithm to a version of the strip packing problem, where while packing one item there must be a free way from the top of the bin until the position where the item is packed. In this model, a rectangle arrives from the top of S as in the well known *TETRIS game*, and it should be moved continuously using only the free space until it reaches its place. In this case both horizontal and vertical movements are allowed. If we consider that items can perform both horizontal and vertical movements, their online algorithm can be easily modified to an *offline* algorithm to the problem SPU_{vh}. If we sort the list L of items by non-increasing order of class values we get L', and then if we use their algorithm in L' we find a feasible solution to SPU_{vh}, since each item a_i had reached its place when there were packed only items of class greater than or equal to $c(a_i)$. Since it could be packed in order, it can also be removed in order. This way, we easily devise an *offline* 4-approximation algorithm for the SPU_{vh} problem.

In [**?**], Fekete, Kamphans and Schweer, proposed a 2.6154-competitive online algorithm for the strip packing problem, where items must be squares. In this algorithm, the items are packed from the top of S and are moved only with vertical movements to reach its final position. In addition, an item is not allowed to move upwards and has to be supported from below when reaching its final position. These conditions are called *gravity constraints*. Their slot based algorithm can be easily used to the SPU_v problem, achieving a 2.6154-approximation, in the special case where items are squares. We just need to

sort the items in non-increasing order of class values. To the authors knowledge the best online algorithm for the general case of rectangles for the strip packing problem, is still the result of Azar and Epstein [**?**].

In this paper we present an algorithm with asymptotic approximation ratio 5.745 to the SPU_v problem and another one with approximation ratio 3 to the SPU_{vh} problem when rotations are allowed. We show an algorithm for the parametric cases of both versions problems. For the SPU_{vh} problem we design an algorithm for the oriented case in which the rectangles have width bounded by $1/m$, where $m \geq 2$. This algorithm has asymptotic ratio $(\frac{m}{m-1} + \varepsilon)$ plus an additive constant of $\frac{2+\varepsilon}{\varepsilon}$. For the parametric case of the SPU_v problem, in which the rectangles have width bounded by $1/m$, $m \geq 3$, we design an asymptotic $\left(\frac{m}{m-2}\right)$-approximation algorithm.

2 An algorithm for the SPU_{vh} problem

In this section we describe a 3-approximation algorithm, denoted by $\mathcal{A}_{vh}$, for the SPU_{vh} problem. We assume the constraint that items must be removed in order from the final packing using vertical and horizontal movements in the available free space of the strip. We assume that both width and height of each rectangle is bounded by 1. First, we present two algorithms that will be used as routines in the $\mathcal{A}_{vh}$ algorithm.

Consider the following four types of items:

- $\mathcal{L}$: items a_i with $h(a_i)$ and $w(a_i) > 2/3$.
- $\mathcal{T}$: items a_i with $h(a_i) > 2/3$ and $w(a_i) < 1/3$.
- $\mathcal{M}$: items a_i with any $h(a_i)$ and $2/3 \geq w(a_i) \geq 1/3$.
- $\mathcal{S}$: items a_i with $h(a_i)$ and $w(a_i) < 1/3$.

It is easy to see that one can properly rotate an item such that it fits into one of these four item types.

We present now a modified version of the classical NFDH (Next Fit Decreasing Height) algorithm, called MNFDH (*Modified* NFDH) (Alg. 1) . The NFDH algorithm generates a packing divided into horizontal levels, each level has height equal to the maximum rectangle height among the rectangles in the level. Rectangles packed in a same level are packed side by side. As in the classical NFDH algorithm, the MNFDH pack the items into horizontal sub-strips (levels), but the rule to close a sub-strip is modified. The MNFDH algorithm deals with items of width strictly smaller than w and generate sub-strips of height h only (w and h are parameters of this algorithm). In the

MNFDH algorithm, a sub-strip F is closed if $w(F) \geq 1 - w$.

Algorithm 1. *Modified* NFDH

1: **Input:** List L of items with width smaller than w and the parameters w and h.
2: **Begin**
3: $\mathcal{F} \leftarrow \varnothing$.
4: Let F be a strip at the bottom of S (level 0) of height h and width $w(F) = 0$.
5: **for** each item $a \in L$ in the input order **do**
6: pack a into F at the left-most position.
7: $w(F) = w(F) + w(a)$
8: **if** $w(F) \geq 1 - w$ **then**
9: $\mathcal{F} \leftarrow \mathcal{F} \cup F$.
10: Close the actual strip F and create a new sub-strip F, with $w(F) = 0$ and height h, above the last one.
11: Let $\mathcal{F}^{nf}$ be the set containing the last sub-strip created if its width is smaller than $1 - w$.
12: **Return**: $(\mathcal{F}, \mathcal{F}^{nf})$.
13: **end**.

Denote each closed sub-strip as a *full* sub-strip (Alg. 1, line 10) and *non-full* otherwise. Notice that all the generated sub-strips are *full*, except possible the last one, since each item a has $w(a) < w$. Furthermore, if we remove the last item (the right-most one) of each *full* sub-strip F, then $w(F) < 1 - w$, since once $w(F) \geq 1 - w$ the sub-strip is closed.

Now we present an algorithm called *Base* (Alg. 2) which deals with items of types $\mathcal{M}$, $\mathcal{T}$ and $\mathcal{S}$. The algorithm MNFDH is used by the *Base* algorithm as a routine to pack items of types $\mathcal{T}$ and $\mathcal{S}$. The algorithm *Base* starts rotating the items a of types $\mathcal{S}$ and $\mathcal{T}$, such that $h(a) \geq w(a)$ and items of type $\mathcal{M}$ such that, $2/3 \geq w(a) \geq 1/3$ (Line 3). After that, it generates five sets of sub-strips, based on the types of each item (Lines 8 - 10). The types $\mathcal{S}$ and $\mathcal{T}$ are packed with the MNFDH algorithm and the items of type $\mathcal{M}$ are just piled left justified on the strip S (packed in individual sub-strips). The sub-strips containing one item of type $\mathcal{M}$, are considered *full*. Finally, it generates the final packing by properly sorting and packing the sets of sub-strips created, aiming to attend the unloading constraint (Lines 11 - 14 and Fig. 2).

We prove two lemmas concerning the *Base* algorithm. The first one (Lemma

Algorithm 2. *Base*

1: **Input:** List L of items of types $\mathcal{T}$, $\mathcal{M}$ and $\mathcal{S}$, partitioned into C classes.
2: **Begin**
3: Rotate the items a of type $\mathcal{S}$ and $\mathcal{T}$ such that $h(a) \geq w(a)$ and the items of type $\mathcal{M}$ such that, $2/3 \geq w(a) \geq 1/3$.
4: Let $L_P = \{a \in L$: a has type $\mathcal{S}\}$
5: Let $L_T = \{a \in L$: a has type $\mathcal{T}\}$
6: Let $L_M = \{a \in L$: a has type $\mathcal{M}\}$
7: Partition the set L_P into $L_{P_0}, L_{P_1}, \ldots, L_{P_k}$, such that, $a \in L_{P_i}$ iff $\frac{1}{2^i \cdot 3} \geq h(a) > \frac{1}{2^{i+1} \cdot 3}$.
8: $(\mathbf{P}_i^F, \mathbf{P}_i^{NF}) \leftarrow$ MNFDH(L_{P_i}, 1/3, $\frac{1}{2^i \cdot 3}$), for $0 \leq i \leq k$
9: $(\mathbf{T}^F, \mathbf{T}^{NF}) \leftarrow$ MNFDH(L_R, 1/3, 1)
10: Sort items of L_M in non-increasing order of class and pack each item $a \in L_M$ in an individual sub-strip left justified. Denote this packing by $\mathbf{M}$.
11: For each sub-strip in $\mathbf{T}^F$,$\mathbf{T}^{NF}$, $\mathbf{P}_i^F$ and $\mathbf{P}_i^{NF}$ $(0 \leq i \leq k)$ sort its items by non-increasing order of class (Starting from left).
12: Let $\mathbf{F} \leftarrow \mathbf{T}^F \cup (\mathbf{P}_i^F,\ 0 \leq i \leq k)$ be the final packing.
13: Sort the sub-strips in $\mathbf{F}$ in non-increasing order of class value of the right-most item. Pack in S the sub-strips of $\mathbf{F}$ in this order.
14: Pack in S, $\mathbf{T}^{NF}$, $\mathbf{P}_i^{NF}, 0 \leq i \leq k$ and $\mathbf{M}$, in order.
15: **Return**: The packing in S.
16: **end**.

2.1), deals with the unloading constraint and Lemma 2.2 deals with the fraction of occupied area of the strip S.

Lemma 2.1 *The packing produced by Base satisfies the constraints of* SPU_{vh}

Proof. We are going to show that any item can be removed from the strip S using one horizontal movement and the one vertical movement.

If an item $a \in \mathbf{M}$ then it can be removed from S since the items that are packed above a have lower class values (Line 10). Now consider an item $a \in \mathbf{T}^{NF} \cup (\mathbf{P}_i^{NF},\ 1 \leq i \leq k)$. In this case, the items in the same sub-strip of a that are to its right will be removed before a (Line 11). Moreover, since a can reach the right-boudary of S, it can be removed from S since all strips F above it have $w(F) \leq 2/3$ (Line 14). Finally, consider an item $a \in \mathbf{T}^F \cup (\mathbf{P}_i^F,\ 1 \leq i \leq k)$, packed in a sub-strip F. Since the items in F are sorted, from left to right, by non-increasing class, a can reach the right-most side of S (Line

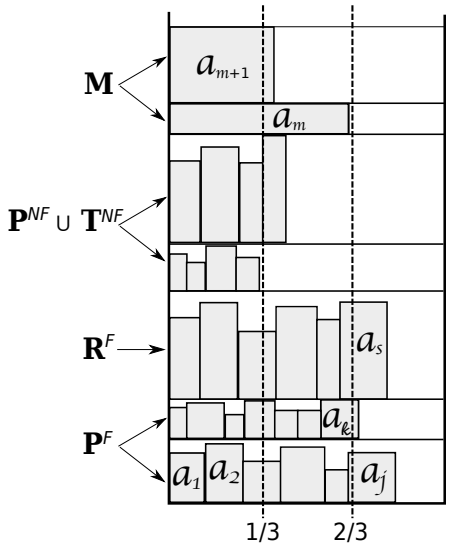

Fig. 2. The packing generated by the *Base* algorithm. Items in each sub-strip are sorted by class in non-increasing order (i.e. $c(a_1) \geq c(a_2) \geq \ldots \geq c(a_j)$. Furthermore, the sub-strips in $\mathbf{P}^F \cup \mathbf{T}^F$ are sorted by the class of the right most item (i.e. $c(a_j) \geq c(a_k) \geq c(a_l)$). Finally, the items in $\mathbf{M}$ are sorted by class in non-increasing order (i.e. $c(a_m) \geq c(a_{m+1})$).

10). Furthermore, since the strips are sorted, starting from the bottom of S, in non-increasing order of the class of the right-most item in the sub-strip, we can conclude that at least one item (the right-most one) of each sub-strip above F, in $\mathbf{T}^F \cup (\mathbf{P}_i^F,\ 1 \leq i \leq k)$, will be removed before a. Thus, a can be removed using the right-most side of S, since at this moment each sub-strip above F has width strictly smaller than $1 - w = 2/3$. □

Lemma 2.2 *The sub-strips in the sets* $\mathbf{F}$ *and* $\mathbf{M}$ *generated by the Base algorithm have at least* $1/3$ *of occupied area.*

Proof. We are going to show that the strips in $\mathbf{T}^F$, $(\mathbf{P}_i^F,\ 0 \leq i \leq k)$ and $\mathbf{M}$, have at least $1/3$ of occupied area.

The sub-strips F with one item a of type $\mathcal{M}$ have $w(a) \geq 1/3$ and $h(F) = h(a)$, we can conclude that this sub-strips has at least, $w(a) \cdot h(a)/h(F) \geq 1/3$ of occupied area.

The sub-strips $F \in \mathbf{T}^F$ have $h(F) = 1$ and $w(F) \geq 1 - w = 2/3$. They have, at least $(2/3)^2 > \frac{1}{3}$ of occupied area, since items of type $\mathcal{T}$ items have height strictly larger than $2/3$. Finally, the sub-strips in $\mathbf{P}_i^F$ have height $\frac{1}{2^i \cdot 3}$

and are used to pack items of height larger than $\frac{1}{2^{i+1}\cdot 3}$, so, they have, at least

$$\frac{(1-w)\cdot\frac{1}{2^{i+1}\cdot 3}}{\frac{1}{2^i\cdot 3}} = \frac{1}{3}$$

of occupied area. □

Now we present the algorithm $\mathcal{A}_{vh}$ (Alg. 3) that computes the packing of all items. The algorithm starts sorting the list L in non-increasing order of class value with ties broken by widest first (Line 4). Then the algorithm proceeds selecting a maximal prefix of items of types $\mathcal{S}$, $\mathcal{T}$ and $\mathcal{M}$, which are going to be packed with the algorithm *Base* (Lines 10 to 12). After that (Lines 13 to 15), the algorithm selects and removes a maximal prefix of items of type $\mathcal{L}$ items, which are going to be packed in individual strips (like the items of type $\mathcal{M}$ in *Base* algorithm). The next step of the algorithm is to push in front of L the *non-full* sub-strips which do not brake the unloading constraint (Lines 16 to 20). Then the algorithm proceeds packing the selected items and sub-strips (Lines 20 and 21). The algorithm ends when there are no more items to be packed.

Lemma 2.3 *The packing produced by $\mathcal{A}_{vh}$ satisfies the constraints of the SPU_{vh} problem.*

Proof. We are going to prove that the unloading constraint is satisfied as a loop invariant for line 5 of the $\mathcal{A}_{vh}$ algorithm.

Notice that before the first iteration we have an empty solution or a sequence of type $\mathcal{L}$ items piled left justified on S, which is trivially feasible. Now suppose that the algorithm is starting its ith iteration over the while loop considered, and assume that the current solution is feasible. Let $\mathbf{F}$ be the set of sub-strips generated in this iteration and $\{b_1, \ldots, b_j\}$ be the set of items of type $\mathcal{L}$ that will be packed in this iteration. By Lemma 2.1 the packing is feasible considering items in $\mathbf{F}$. The only problem are those items of types $\mathcal{T}$ and $\mathcal{S}$ in $\mathbf{F}$ with classes higher than the classes of the items already packed. These items came from *non-full* sub-strips F that were not packed in previous iterations because $w(F) + w(b) \leq 1$ (for some item b of type $\mathcal{L}$) Notice that $w(F) \leq 1 - w(b) \leq 1/3$ (Line 12). Then, each new sub-strip F' of type $\mathcal{T}$ or $\mathcal{S}$ contains at most $w' = 1/3$ of width occupied with higher class items. Since items in each one of these sub-strips are sorted by non-increasing order of class, these higher classes items can not block any other item a already packed because it can block only items of type $\mathcal{L}$. But this does not occur

Algorithm 3. $\mathcal{A}_{vh}$

1: **Input:** List L of n items partitioned into C classes.
2: **Begin**
3: Assign each item of L into one of the types $\mathcal{L}$, $\mathcal{T}$, $\mathcal{M}$ and $\mathcal{S}$ doing rotations if necessary.
4: For each item $a_i \in \mathcal{L} \cup \mathcal{S}$ rotate it such that $h(a_i) \geq w(a_i)$, $1 \leq i \leq n$.
5: Sort items of L by class value in non-increasing order. Ties are broken by widest first.
6: Let $B = \{b_1, \ldots, b_j\}$ be a maximal prefix containing only items of type $\mathcal{L}$. in L.
7: Pack B left justified in individual sub-strips at the bottom of S in order.
8: $L \leftarrow L \backslash B$
9: **while** $L \neq \varnothing$ **do**
10: Let $\{a_1, \ldots, a_i\}$ be a maximal prefix of L containing only items of types $\mathcal{S}$, $\mathcal{T}$ or $\mathcal{M}$.
11: Let $\mathcal{F} \leftarrow Base(\{a_1, \ldots, a_i\})$
12: $L \leftarrow L \backslash \{a_1, \ldots, a_i\}$
13: Let $B = \{b_1, \ldots, b_j\}$ be a maximal prefix of items of type $\mathcal{L}$ in L.
14: $L \leftarrow L \backslash B$
15: Let b be the widest item in B
16: **for** each *non-full* sub-strip $F \in \mathcal{F}$ **do**
17: **if** $w(F) + w(b) \leq 1$ **then**
18: $\mathcal{F} \leftarrow \mathcal{F} \backslash \{F\}$.
19: **for** each item $a' \in F$ **do**
20: Insert a' in L, keeping it sorted.
21: Pack $\mathcal{F}$ in S in order.
22: Pack items in B left justified in individual sub-strips above all previous items.
23: **Return**: The generated packing.
24: **end**.

since $w' + w(b) \leq 1$ for all items b of type $\mathcal{L}$ of lower classes.

□

2.1 $\mathcal{A}_{vh}$ *Analysis*

Lemma 2.4 *The set of sub-strips generated by the* $\mathcal{A}_{vh}$ *algorithm has at least* 1/3 *of occupied area, except for 5/3 of height.*

Proof. At first, notice that the *full* sub-strips have, at least, 1/3 of occupied

area (Lemma 2.2).

So we just need to prove that the *non-full* sub-strips and the strips with items of type $\mathcal{L}$ also have, at least 1/3 of occupied area on average. We are going to associate each *non-full* sub-strip packed in line 16 of the $\mathcal{A}_{vh}$ algorithm with the largest item of type $\mathcal{L}$ above it packed in the same iteration. Notice that at any given iteration, the *Base* algorithms generates at most one *non-full* sub-strip in $\mathbf{T}^{NF}$ of type $\mathcal{T}$ and at most one *non-full* sub-strip $\mathbf{P}_i^{NF}$ for each $i \geq 0$, that packs items of type $\mathcal{S}$.

The total height of $\mathbf{P}_i^{NF}$ sub-strips is

$$sh = \sum_{i=0}^{\infty} \frac{1}{2^i \cdot 3} = 2/3$$

The total height of an item of type $\mathcal{T}$ is 1.

If at some iteration no item of type $\mathcal{L}$ is packed, then this is the last iteration. So at most 5/3 of height of *non-full* sub-strips will not be associated with any items of type $\mathcal{L}$.

Consider some iterations where *non-full* sub-strips aare packed, and let b be the widest item of type $\mathcal{L}$ packed in this same iteration above this *non-full* sub-strips. We associate them uniquely with b.

The sub-strip B that contains b has $h(B) = h(b)$ and $w(B) = w(b) \geq 2/3$. Besides that, the possible sub-strip $F \in \mathbf{T}^{NF}$ has $h(F) = 1$ and $w(F) \geq 1 - w(b)$ otherwise it would not be packed in this iteration. Furthermore each item a in F has $h(a) \geq 2/3$. Finally, each sub-strip $F_i \in \mathbf{T}^{NF}$ has $h(F_i) = \frac{1}{2^i \cdot 3}$ and also $w(F_i) \geq 1 - w(b)$. Moreover each item a in F_i has $h(a) > \frac{1}{2^{i+1} \cdot 3}$.

We can bound the occupied area in the strips B, F, and all F_i by

$$(1) \quad \frac{h(b) \cdot w(b) + [1 - w(b)] \cdot 2/3 + [1 - w(b)] \cdot \frac{sh}{2}}{h(b) + 1 + sh}$$

where $2/3 < w(b) \leq h(b) \leq 1$ and $0 \leq sh \leq 2/3$. The minimum of this function occurs when $w(b) = h(b) = 2/3 + \epsilon$ and $sh = 2/3$ and has value $> 1/3$ (see Appendix 7.1). □

Theorem 2.5 *Let L be a list of rectangles, then, $\mathcal{A}_{vh}(L) \leq 3 \cdot OPT(L) + 5/3$.*

Proof. Let S be the solution returned by the algorithm $\mathcal{A}_{vh}$. It is filled by *full* sub-strips, and type $\mathcal{L}$ items with its associated *non-full* sub-strips. Due to Lemma 2.4, the strip S have at least 1/3 of occupied area on average, except perhaps the 5/3 of height. So we can conclude that $(\mathcal{A}_{vh}(L) - 5/3) \cdot 1/3 \leq$

$\sum_{a_i \in L} h(a_i) \cdot w(a_i)$ and then

$$\mathcal{A}_{vh}(L) \leq 3 \cdot OPT(L) + 5/3$$

□

3 An Algorithm to a parametric oriented case of the SPU$_{vh}$ Problem

In this version of the problem, we are going to assume that the items can not be rotated and the width of the items is bounded by $1/m$, $m \geq 2$. Since the approximation analysis is based on area arguments the result remain valid for the case in which rotations are allowed.

The algorithm is called $\mathcal{A}^p_{vh}$ (Alg. 4) and takes two parameters: the factor m and a constant ϵ. The algorithm starts by partitioning the input list L by the height of the items. Then it generates a set of strips for each partition using the algorithm MNFDH (Alg. 1). Finally it proceeds by packing and sorting the *full* sub-strips and then packing the *non-full* ones.

Algorithm 4. $\mathcal{A}^p_{vh}$

1: **Input:** List L of items partitioned into C classes, m and ϵ.
2: **Begin**
3: $\mathcal{P} \leftarrow \varnothing$ and $\mathcal{N} \leftarrow \varnothing$
4: Let $r = \frac{1}{1+\epsilon/2}$
5: Let $L_i = \{a : a \in L \text{ and } r^{i+1} < h(a) \leq r^i\}$, for $i \geq 0$.
6: **for** each i **do**
7: $\quad (\mathcal{F}, \mathcal{F}') \leftarrow \text{MNFDH}(L_i, 1/m, r^i)$
8: $\quad \mathcal{P} \leftarrow \mathcal{P} \cup \mathcal{F}$
9: $\quad \mathcal{N} \leftarrow \mathcal{N} \cup \mathcal{F}'$
10: Sort items in each sub-strip in $\mathcal{P} \cup \mathcal{N}$ by non-increasing order of class (Starting from left).
11: Sort the sub-strips in $\mathcal{P}$ by non-increasing order of class of the right-most item.
12: Pack $\mathcal{P}$ in S in order and then pack , $\mathcal{N}$ in S above $\mathcal{P}$.
13: **Return**: The generated packing.
14: **end**.

Lemma 3.1 *The packing produced by $\mathcal{A}^p_{vh}$ satisfies the constraints of SPU$_{vh}$*

Proof. This proof is similar to the proof of the Lemma 2.1. □

3.1 $\mathcal{A}^p_{vh}$ Analysis

Theorem 3.2 *Let L be a list of rectangles, then $\mathcal{A}^p_{vh}(L) \leq \left(\frac{m}{m-1} + \epsilon\right) \cdot OPT(L) + \frac{2+\epsilon}{\epsilon}$.*

Proof. First, consider some sub-strip $F \in \mathcal{P}$. Since this sub-strip is *full*, we have that $w(F) \geq 1 - 1/m$ and, moreover, suppose that F was created to pack items from the list L_i, which means that this sub-strip has, at least $\frac{r^{i+1}}{r^i} = r$ of occupied height. Then we can bound the occupied area in the strip F by $r(1 - 1/m) = \frac{r(m-1)}{m}$. Also notice that there is, at most one sub-strip $F \in \mathcal{N}$ with items from L_i. So we can bound the total height of strips in $\mathcal{N}$ by $\sum_{i=0}^{\infty} r^i = \frac{1}{1-r} = \frac{2+\epsilon}{\epsilon}$

Finally we have $(\mathcal{A}^p_{vh}(L) - \frac{2+\epsilon}{\epsilon}) \cdot \frac{r(m-1)}{m} \leq \sum_{a_i \in L} h(a_i) \cdot w(a_i)$ and then

$$\mathcal{A}^p_{vh}(L) \leq \left(\frac{(1+\epsilon/2) \cdot m}{m-1}\right) \cdot OPT(L) + \frac{2+\epsilon}{\epsilon} \leq \left(\frac{m}{m-1} + \epsilon\right) \cdot OPT(L) + \frac{2+\epsilon}{\epsilon}$$

□

4 An Algorithm to the SPU$_v$ Problem

In this Section we present an algorithm, denoted by $\mathcal{A}_v$ to solve the SPU$_v$ problem when only vertical movements of items are allowed and orthogonal rotation of items is allowed. We are going to use the NFDH algorithm. In [**?**], Meir and Moser proved the following result for the NFDH:

Theorem 4.1 *Any list of rectangles $L = \{a_1, \ldots, a_n\}$ with total area A can be packed by the NFDH algorithm into a unit square if $1 \geq w(a_i) \geq h(a_i)$, for $i = 1, \ldots, n$; $h(a_i) \geq h(a_{i+1})$, for $i = 1, \ldots, n-1$ and $A \leq \frac{7}{16}$.*

The algorithm $\mathcal{A}_v$ computes the solution in two stages. First it packs the items into bins of height 1 and width 1 (unit square bins), using the NFDH algorithm. The algorithm also packs some large items alone in one bin. The bins used to pack these large items have height 1 and width equal to the width of the item. Then it packs the bins rotated into S (the strip) such that each level (sub-strip) created by the NFDH becomes vertical, but with limited height 1, the maximum width of the bin. We first show the bin packing algorithm named *Bin Packing Decreasing Order* (BPDO) in Algorithm 5, and then we present the $\mathcal{A}_v$ in Algorithm 6.

Denote by B_k the kth bin created by the algorithm BPDO, $h(B_k)$ its occupied height and $w(B_k)$ the occupied width. Also denote by $A(L) = \sum_{a_i \in L} h(a_i)w(a_i)$, where $L = (a_1, \ldots, a_m)$ is a list of items.

The BPDO algorithm starts by sorting the items in non-increasing order of class values (line 4). Then it selects the largest prefix of items that can be packed in a bin B of height 1 and width 1 using the NFDH algorithm (line 6). By Theorem 4.1 these items can be packed in a single bin (line 9). After packing these items, the algorithm sorts each level created by the NFDH algorithm by class values (line 10). The algorithm then removes the packed items from the list of items (line 11). If the first item a_i of the remaining list has $w(a_i)h(a_i) \geq 0.263422$, the algorithm packs it alone in a new bin B', of width $w(a_i)$ (line 14). The algorithm repeats this process until $L = \varnothing$.

Algorithm 5. Bin Packing Decreasing Order (BPDO)

1: **Input:** List L of items partitioned into C different classes.
2: **Begin**
3: Let $LB = \varnothing$ be a list of bins.
4: Sort items in L in non-increasing order of class value.
5: **while** $(L \neq \varnothing)$ **do**
6: Let $L' = (a_i, \ldots, a_k)$ be the largest prefix of L such that $A(L') \leq 7/16 < A(L' \cup \{a_{k+1}\})$.
7: Rotate the items a_i in L' such that $w(a_i) \geq h(a_i)$.
8: Sort L' in non-increasing order of height.
9: Pack L' using the NFDH algorithm in a new bin B.
10: Sort itens in each generated level in non-increasing order of class value.
11: $L \leftarrow L \backslash L'$ and $LB \leftarrow LB + B$
12: **if** $(A(\{a_{k+1}\}) \geq 0.263422)$ **then**
13: Rotate a_{k+1} such that $h(a_{k+1}) \geq w(a_{k+1})$
14: Pack a_{k+1} in a new bin B' of height 1 and width $w(a_{k+1})$.
15: $L \leftarrow L \backslash \{a_{k+1}\}$ and $LB \leftarrow LB + B'$
16: Return LB.
17: **end**.

The algorithm $\mathcal{A}_v$ is presented in Algorithm 6. It just calls the algorithm BPDO, merge all bins returned side by side forming a strip of height 1 and width equal to the total width of the bins. The strip is rotated to provide a solution to the original problem.

Algorithm 6. $\mathcal{A}_v$

1: **Input:** $L = \{a_1, a_2, \ldots, a_n\}$
2: **Begin**
3: Let $B_1, B_2, \ldots, B_m$ be the bins computed by BPDO in the order they were created.
4: Concatenate $B_1, B_2, \ldots, B_m$ forming one strip S of height 1 and width $\sum_{k=1}^{m} w(B_i)$.
5: **Return** S rotated such that its width is 1 and its height is $\sum_{k=1}^{m} w(B_k)$.
6: **end**.

Theorem 4.2 *The packing produced by $\mathcal{A}_v$ satisfies the constraints of SPU_v*

Proof. Let $B_1, \ldots, B_m$ be the bins created by BPDO in the order they were created. These bins are packed in the strip S in the order they were created. For each successive pair of bins B_k and B_{k+1} we guarantee that all items in B_{k+1} have class smaller than or equal to the items in B_k, since the algorithm packs items in non-increasing order of classes. So items in one bin will not block items in previous bins. Inside each bin, the feasibility of the solution is guaranteed by the packing in levels. Items in each level are sorted by non-increasing order of class value. Each level generated by the NFDH algorithm is rotated in the final solution (line 5 of Alg. 6). Also notice that different levels does not interfere with each other, since all items are packed completely inside a sub-strip (level). □

4.1 $\mathcal{A}_v$ Analysis

In this Section we use arguments based on the occupied area of each bin to prove the approximation of the $\mathcal{A}_v$ algorithm.

Lemma 4.3 *Let $B_1, \ldots, B_m$ be the bins computed by BPDO in the order they were created. Then $B_1, \ldots, B_{m-1}$ have occupied area of at least* 0.174 *on average.*

Proof. We will prove that the bins created in each iteration of the main loop (line 5) have at least 0.174 of occupied area on average (except perhaps the last created bin). Consider some iteration of the main loop at line 5. Let B_j be the jth-created bin using NFDH(L') at line 9, where $L' = \{a_i, \ldots, a_k\}$.
Case 1: Consider that $A(\{a_{k+1}\}) < 0.263422$. Since $A(L') \leq 7/16 < A(L' \cup \{a_{k+1}\})$ we have

$$A(L') > 7/16 - A(\{a_{k+1}\}) > 7/16 - 0.263422 \approx 0.174078$$

and all items in L' are packed in B_j by Theorem 4.1. Since $A(\{a_{k+1}\}) < 0.263422$ only this bin is created on this iteration of the loop.

Case 2: Consider that $A(\{a_{k+1}\}) \geq 0.263422$. Then a_{k+1} is rotated, such that $h(a_{k+1}) \geq w(a_{k+1})$, and it is packed in a new bin B_{j+1}, where $w(B_{j+1}) = w(a_{k+1})$.

The total width occupied in the two bins B_j and B_{j+1} is at most $(1 + w(a_{k+1}))$ and the total area of the items packed in these two bins is $h(a_{k+1})w(a_{k+1}) + A(L')$. Then we can bound the occupied area in the occupied width of bins B_k and B_{k+1} by

$$\frac{w(a_{k+1})h(a_{k+1}) + A(L')}{1 + w(a_{k+1})} \tag{2}$$

where $1 \geq h(a_{k+1}) \geq w(a_{k+1}) > 0$, $A(\{a_{k+1}\}) > 0.263422$ and $0 < A(L') \leq 7/16$. The minimum of this function occurs when $w(a_{k+1}) = h(a_{k+1}) \approx 0.51325$, $A(L') = 0$ and has value $\approx 0.174077 > 0.174$ (see 7.2). □

Theorem 4.4 *Let L be a list of rectangles, then, $\mathcal{A}_v(L) \leq 5.745OPT(L) + 1$.*

Proof. Due to Lemma 4.3, each bin created by the algorithm has on average 0.174 of occupied area in the corresponding width, except perhaps the last generated bin. Since the total width of the bins corresponds to the total height of the used strip we have $(\mathcal{A}_v(L) - 1)0.174 \leq \sum_{a_i \in L} w(a_i) \cdot b(a_i)$ and then

$$\mathcal{A}_v(L) \leq 5.745 \sum_{a_i \in L} w(a_i) \cdot b(a_i) + 1 \leq 5.745OPT(L) + 1.$$

□

5 An algorithm to a parametric version of the SPU$_v$ problem

In this Section we present an algorithm for a special version of the SPU$_v$ problem where the width and the height of the items are bounded by $1/m$, $m \geq 3$ and the items can be rotated. The algorithm presented in this section is similar to the $\mathcal{A}_v$ algorithm, but even simpler. In this case we are going to use a result presented by Li and Cheng [**?**]

Theorem 5.1 *A list L of rectangles $L = \{a_i, \ldots a_n\}$, can be packed in a unit square bin by the NFDH algorithm if exists an integer $m \geq 3$ such that:*

(i) $h(a_i) \leq \frac{1}{m}$, $1 \leq i \leq n$

(ii) $w(a_i) \leq \frac{1}{m}$, $1 \leq i \leq n$

(iii) $\sum_{i=1}^{n} h(a_i)w(a_i) \leq \left(1 - \frac{1}{m}\right)^2$

The algorithm for this restricted case is called $\mathcal{A}_v^p$ (Alg. 7) and takes only one parameter, the value of m. The algorithm starts by sorting the input list L in class in non-increasing order of class (Line 4). Then, while exists items to be packed, it selects the largest prefix of items that can be packed in a unit square bin B using the NFDH algorithm (line 6). This prefix is selected using the Theorem 5.1 (line 9). After packing this prefix, the algorithm sorts items in each level created by the NFDH algorithm by class values (line 10). The algorithm repeats this process until $L = \varnothing$. Finally, it joins the created bins forming the final strip S (Line 11).

Algorithm 7. $\mathcal{A}_v^p$

1: **Input:** List L of items partitioned into C different classes.
2: **Begin**
3: Let $LB = \varnothing$ be a list of bins.
4: Sort items in L in non-increasing order of class.
5: **while** $(L \neq \varnothing)$ **do**
6: Let $L' = (a_i, \ldots, a_k)$ be the largest prefix of L such that $A(L') \leq \left(1 - \frac{1}{m}\right)^2 < A(L' \cup \{a_{k+1}\})$.
7: Rotate the items a_i in L' such that $w(a_i) \geq h(a_i)$.
8: Sort items in L' in non-increasing order of height.
9: Pack L' using the NFDH algorithm in a new bin B.
10: Sort items in each generated level in non-increasing order of class value.
11: Concatenate the list $LB = \{B_1, \ldots, B_m\}$ in order, forming one strip S of height 1 and width $\sum_{k=1}^{m} w(B_i)$.
12: **Return** S rotated such that its width is 1 and its height is $\sum_{k=1}^{m} w(B_k)$.
13: **end**.

Lemma 5.2 *The packing produced by $\mathcal{A}_v^p$ satisfies the constraints of SPU_v*

Proof. This proof is similar to the proof of the Theorem 4.2. □

5.1 $\mathcal{A}_v^p$ Analysis

Theorem 5.3 *Let L be a list of rectangles, then $\mathcal{A}_v^p(L) \leq \left(\frac{m}{m-2}\right) \cdot OPT(L) + 1$.*

Proof. Notice that each bin $B_i \in LB$ has at least $(1-\frac{1}{m})^2 - \frac{1}{m^2} = \frac{m-2}{m}$ of occupied area, since $A(B_i) + h(a)w(a) > \left(1-\frac{1}{m}\right)^2$ for some item a (Line 6).

Then we have $(\mathcal{A}_v^p(L)-1)\cdot\frac{m-2}{m} \leq \sum_{a_i\in L} h(a_i)\cdot w(a_i)$ and then

$$\mathcal{A}_v^p(L) \leq \left(\frac{m}{m-2}\right)\cdot OPT(L) + 1$$

□

6 Concluding Remarks

In this paper we consider a variant of the two dimensional strip packing problem where we have constraints on how items can be removed from the strip. These are called unloading constraints and appear in vehicle routing problems where items are delivered along a route. We presented an algorithm with asymptotic performance bound 5.745 for the SPU_v problem when rotations are allowed. We also design a 3-approximation algorithm for the SPU_{vh} problem when rotations are allowed. Finally, we also design two algorithms for parametric cases of both problems. For the parametric SPU_{vh} problem where the rectangles have width bounded by $1/m$, $m \geq 2$, we design an algorithm with asymptotic ratio $(\frac{m}{m-1}+\varepsilon)$ plus an additive constant of $\frac{2+\varepsilon}{\varepsilon}$. For the parametric SPU_v problem, in which the rectangles have width bounded by $1/m$, $m \geq 3$, we design an asymptotic $\left(\frac{m}{m-2}\right)$-approximation algorithm.

7 Appendix

7.1 Minimizing function (1)

Proof. We have to find the minimum of the function

$$f(x,y,z) = \frac{xy + (z/2+2/3)\cdot(1-y)}{x+z+1}$$

where $1 \geq y \geq x \geq 2/3$ and $2/3 \geq z \geq 0$.

This function has no critical points in the considered region. Moreover the function $\frac{\partial f}{\partial z} = \frac{x(3-9y)+y-1}{6(x+z+1)^2} < 0$, in the considered region. Thus the minimum of this function can be found on the limits of the region, in this case when $z = 2/3$.

So we just need to consider the function

$$g(x,y) = \frac{3y(x-1)+3}{3x+5}$$

over the regions $x = 1$, $x = y$ and $y = 2/3$.

When $x = 1$ the function g has value $\frac{3}{8}$. If $x = y$, then we must minimize the function $\frac{3(1-x+x^2)}{5+3x}$, $2/3 < x \leq 1$, which has a point of minimum when $x = 2/3$, which leads to a minimum of $1/3$. Finally, when $y = 2/3$, we must minimize the function $\frac{2x+1}{3x+5}$, $2/3 < x \leq 1$, which has minimum when $x = 2/3$, which leads , again, to $1/3$.

Thus, the minimum of f relies on $x = y = z = 2/3$ and has value $1/3$. □

7.2 Minimizing function (2)

Proof. Since $A(L')$ only adds area to the function we can easily see that the minimum of the function occurs when $A(L') = 0$ and so we have to find the minimum of the function

$$f(x,y) = \frac{xy}{x+1}$$

where $1 \geq y \geq x > 0$ and $xy > 0.263422$ (see the analyzed region in Fig. 3).

The critical points of this function can be found with $\frac{\partial f}{\partial x} = 0$ and $\frac{\partial f}{\partial y} = 0$ and we have the point $(x,y) = (0,0)$. Since this point is outside of the considered region (see Fig. 3) we analyze the boundaries of the region.

Case 1: Consider that $y = 1$. Then $f(x,1) = \frac{x}{x+1}$ which is strictly increasing at $(0.263422; 1]$ and so we find the minimum with $x = 0.263422$, which leads to a minimum of 0.208499.

Case 2: Consider that $x = y$. Then $f(x,x) = \frac{x^2}{x+1}$ which is strictly increasing at $(\sqrt{0.263422}; 1]$ and so we find the minimum with $x = \sqrt{0.263422}$, which leads to a minimum of $\frac{0.263422}{1+\sqrt{0.263422}}$.

Case 3: Consider that $xy = 0.263422$. Then $f(x,y) = \frac{0.263422}{x+1}$ which is strictly decreasing at $(0.263422, \sqrt{0.263422}]$ and so we find the minimum with $x = \sqrt{0.263422}$, which leads to a minimum $\frac{0.263422}{1+\sqrt{0.263422}}$.

So the minimum of the function

$$\frac{w(a_{k+1})h(a_{k+1}) + A(L')}{1 + w(a_{k+1})}$$

occurs when $w(a_{k+1}) = h(a_{k+1}) = \sqrt{0.263422}$, $A(L') = 0$ and has value $\frac{0.263422}{1+\sqrt{0.263422}} \approx 0.174077$. □

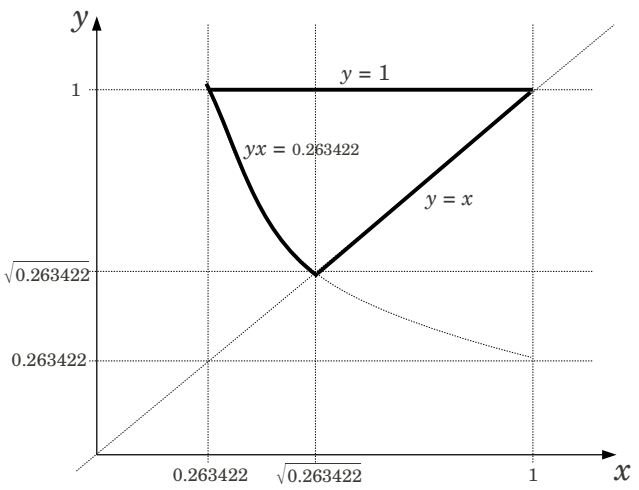

Fig. 3. The boundaries of the analyzed region: $y = x$, $y = 1$, $y = \frac{0.263422}{x}$.