High Multiplicity Strip Packing with Three Rectangle Types

Andrew Bloch-Hansen[1*†], Roberto Solis-Oba[1†] and Andy Yu[1†]

[1*]Department of Computer Science, Western Univerisity, 1251 Richmond Street, London, N6A 3K7, Ontario, Canada.

*Corresponding author(s). E-mail(s): ablochha@uwo.ca;
Contributing authors: solis@uwo.ca; ayu@uwo.ca;
†These authors contributed equally to this work.

Abstract

The two-dimensional strip packing problem consists of packing in a rectangular strip of width $\mathbf{1}$ and minimum height a set of $\boldsymbol{n}$ rectangles, where each rectangle has width $\mathbf{0} < \boldsymbol{w} \leq \mathbf{1}$ and height $\mathbf{0} < \boldsymbol{h} \leq \boldsymbol{h_{max}}$. We consider the high-multiplicity version of the problem in which there are only $\boldsymbol{K}$ different types of rectangles. For the case when $\boldsymbol{K} = \mathbf{3}$, we give an algorithm providing a solution requiring at most height $\frac{3}{2}\boldsymbol{h_{max}} + \boldsymbol{\epsilon}$ plus the height of an optimal solution, where $\boldsymbol{\epsilon}$ is any positive constant. For the case when $\boldsymbol{K} = \mathbf{4}$, we give an algorithm providing a solution requiring at most $\frac{7}{3}\boldsymbol{h_{max}} + \boldsymbol{\epsilon}$ plus the height of an optimal solution. For the case when $\boldsymbol{K} > \mathbf{3}$, we give an algorithm providing a solution requiring at most $\lfloor \frac{3}{4}\boldsymbol{K} \rfloor + \mathbf{1} + \boldsymbol{\epsilon}$ plus the height of an optimal solution.

Keywords: LP-relaxation, two-dimensional strip packing, high multiplicity, approximation algorithm

1 Introduction

The *two-dimensional strip packing problem* (2DSPP) is defined as follows.

Definition 1 Given n rectangles with widths w_1, w_2, ..., w_n and heights h_1, h_2, ..., h_n, where $0 < w_i \leq 1$ for $i = 1, 2, ..., n$, the goal is to pack all the rectangles without rotations or overlaps in a rectangular strip of width 1 and minimum height.

This is a well-studied problem with applications in areas as diverse as resource allocation, scheduling, manufacturing, and transportation, among others. 2DSPP is equivalent to the classical bin packing problem if all rectangles have the same height, and since the bin packing problem is NP-hard [7] then 2DSPP is also NP-hard; therefore, the best possible approximation ratio achievable in polynomial time for 2DSPP is $\frac{3}{2}$ unless P = NP.

Baker et al. [1] designed the first approximation algorithm for 2DSPP which has approximation ratio 3. Coffman et al. [4] presented an algorithm with approximation ratio 2.7, Sleator [15] improved the approximation ratio to 2.5, and Schiermeyer [14] and Steinberg [17] further reduced the approximation ratio to 2. Harren and Van Stee [8] later presented an algorithm with approximation ratio 1.9396. The best known approximation algorithm for 2DSPP is from Harren et al. [9] with approximation ratio $\frac{5}{3} + \epsilon$. Several Asymptotic Polynomial Time Approximation Schemes (APTAS) have been presented as well: Kenyon and Rémila [13] gave an APTAS with an additive constant of $O(\frac{1}{\epsilon^2})$, and Jansen and Solis-Oba [11] improved Kenyon and Rémila's additive constant to 1. Sviridenko [16] presented a polynomial time algorithm that computes a solution of value $OPT + O(\sqrt{OPT \log OPT})$, where OPT is the value of an optimal solution.

In this paper we study the *two-dimensional high multiplicity strip packing problem* (2DHMSPP), in which there is only a fixed number K of different rectangle types. First published in the 7th International Symposium, ISCO 2022, by Springer Nature [3], this paper extends the previous work by including additional proofs of correctness for our algorithm, an algorithm for the case when $K = 4$, a general algorithm for any fixed value for K, and experimental results for our algorithm for the case when $K = 3$.

Note that the input to 2DHMSPP can be described using a list of only $3K$ numbers: the width w_i, height h_i, and number n_i of rectangles of each type T_i. Therefore, a challenging issue faced when designing an approximation algorithm for the problem is to ensure that its running time is a polynomial function of the size of the input. Observe that even describing a feasible solution for the problem using a polylogarithmic number of bits is not trivial as this requires specifying the positions of n rectangles in the packing; therefore, it is unknown whether 2DHMSPP belongs to the class NP.

We present an algorithm for 2DHMSPP for the case when $K = 3$ that computes solutions of value at most $OPT + \frac{3}{2}h_{max} + \epsilon$, where OPT is the value of an optimum solution, h_{max} is the height of the tallest rectangle, and ϵ is a positive constant. This is an improvement over the works of Yu and Solis-Oba [18] and Bloch-Hansen and Solis-Oba [2] whose algorithms computed solutions of value at most $OPT + \frac{5}{3}h_{max} + \epsilon$. Our approach uses a formulation of 2DHMSPP that allows fractional rectangles in the solution called the *two-dimensional fractional strip packing problem* (2DFSPP). We show that a solution for 2DFSPP can be converted into a solution for 2DHMSPP by a careful shifting, re-shaping, and combining of the fractional rectangles to form whole rectangles while increasing the height of the solution by at most $\frac{3}{2}h_{max} + \epsilon$. Our analysis is nearly tight as it is not hard to see that there are instances for which the corresponding fractional and integral solutions differ by h_{max}.

We also give an algorithm for the case when $K = 4$ that computes solutions of value at most $OPT + \frac{7}{3}h_{max} + \epsilon$, and an algorithm that for any fixed K computes solutions of height at most $OPT + \lfloor \frac{3}{4}Kh_{max} \rfloor + h_{max} + \epsilon$. In addition we performed an experimental evaluation of our algorithm for $K = 3$ and our results show that our algorithm has much better than the above theoretical upper bound.

The rest of the paper is organized in the following way. In Section 2 we describe how to compute a near optimum solution for 2DFSPP. In Sections 3-5 we present our algorithm for the case when $K = 3$. In Section 6 we describe a polynomial time implementation of the algorithm. In Section 7 we present our algorithm for the case when $K = 4$. In Section 8 we describe an algorithm for the case when $K > 3$. Finally, in Section 9 we describe our experimental results for the case when $K = 3$.

2 Solving 2DFSPP in Polynomial Time

2DHMSPP can be relaxed to the *two-dimensional fractional strip packing problem* (2DFSPP) by allowing horizontal cuts on the rectangles. A solution to 2DFSPP consists of a set of *configurations*. A *base configuration* C_j consists of a multiset of rectangle types whose total width is at most 1 (see Figure 1). A base configuration can be specified by indicating the number of rectangles of each type T_i in it. For example, the base configuration shown in Figure 1 consists of 4 rectangles of type T_1, 2 rectangles of type T_2, and 3 rectangles of type T_3, so that base configuration can be represented with the tuple (4,2,3).

A group of rectangles following a base configuration can be stacked on top of each other as shown in Figure 1, so that any horizontal line parallel to the base of the strip drawn across any part of the group will intersect the same multiset of rectangle types. This group of rectangles is called a *configuration*. A vertical line drawn across any part of a configuration will intersect either only rectangles of the same type, or empty space. The height of a vertical line intersecting rectangles of a configuration is called the height of the configuration. The configurations are stacked one on top of the other to form a fractional packing (see Figure 2b). Note that the number of possible configurations is $O(n^K)$.

For a configuration C_j let $n_{i,j}$ be the number of rectangles of type T_i in its base configuration, for $i = 1, 2, ..., k$. Let x_j be a variable denoting the height of C_j. Let J be the set of all possible configurations. 2DFSPP can be expressed as the following linear program, hereafter referred to as linear program (1):

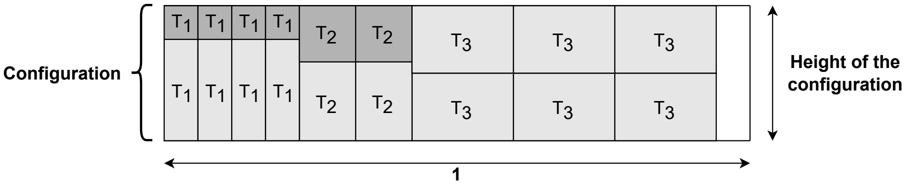


Fig. 1: A configuration with *base configuration* (4,2,3). The fractional rectangles are shaded in a darker color.

$$
\begin{aligned}
\text{Minimize: } & \sum_{C_j \in J} x_j \\
\text{Subject to: } & \sum_{C_j \in J} x_j n_{i,j} \geq n_i h_i, \text{ for each rectangle type } T_i \\
& x_j \geq 0, \text{ for each } j \in J
\end{aligned} \tag{1}
$$

where n_i is the number of rectangles of type T_i and h_i is the height of each rectangle of type T_i. The objective function is to minimize the total height of the packing.

We denote with $OPT(I)$ the height of an optimal packing for instance I of 2DHMSPP and denote with $LIN(I)$ an optimal solution to the corresponding instance of 2DFSPP. It is not hard to see that $LIN(I) \leq OPT(I)$.

Note that 2DFSPP is identical to the fractional bin packing problem; in the latter problem a base configuration is a set of items that fit within a single bin and a solution to linear program (1) gives the fractional number of bins needed to pack all the items. Therefore, we can use an algorithm of Karmarkar and Karp [12] to compute a basic feasible solution for linear program (1) in time $O(K^9 \log K \log^2 \frac{K}{\epsilon})$ of value at most $LIN(I) + \epsilon$ for any fixed $\epsilon > 0$.

In any basic feasible solution, the number of nonzero variables is at most the number of constraints [10]. Thus, the number of nonzero variables, and therefore, the number of configurations used in a basic feasible solution for linear program (1) is at most the number of rectangle types, K.

A simple algorithm for 2DHMSPP is to compute a basic feasible solution for linear program (1) and replace each fractional rectangle with a whole one of the corresponding type, shifting surrounding rectangles upwards as needed. Since a basic feasible solution for (1) uses at most K configurations and replacing the fractional rectangles with whole ones increases the height of a configuration by at most h_{max}, this algorithm computes a solution to 2DHMSPP of height at most $OPT(I) + Kh_{max} + \epsilon$.

3 Algorithm for 2DHMSPP with Three Rectangle Types

When $K = 3$ a basic feasible solution for linear program (1) consists of at most three configurations. Our algorithm performs several steps described in detail in the next sections: 1) the fractional solution of the linear program is divided in two parts: S_{Common} and $S_{Uncommon}$, and the fractional rectangles in S_{Common} are rounded up; 2) in $S_{Uncommon}$ the rectangles in each configuration are sorted and $S_{Uncommon}$ is further partitioned into vertical sections; 3) the vertical sections are grouped according to the heights of the fractional rectangles in them; and 4) the fractional rectangles in each group are combined and/or rounded into whole ones depending on their heights.

We assume for now that the fractional solution computed by solving linear program (1) consists of three configurations. We will show later how to deal with the case when the fractional solution consists of fewer than three configurations.

3.1 Partitioning the Packing

For notational simplicity, in the sequel we assume $h_{max} = 1$. The three configurations of the solution for linear program (1) are stacked one on top of the other as shown in Figure 2a. Rectangles are rearranged horizontally within the configurations so that rectangles of the same type appearing in all three configurations are placed together in a section on the left side of the packing called S_{Common}. In the remaining portion of the packing, called $S_{Uncommon}$, each rectangle type is packed in at most 2 configurations (see Figure 2b).

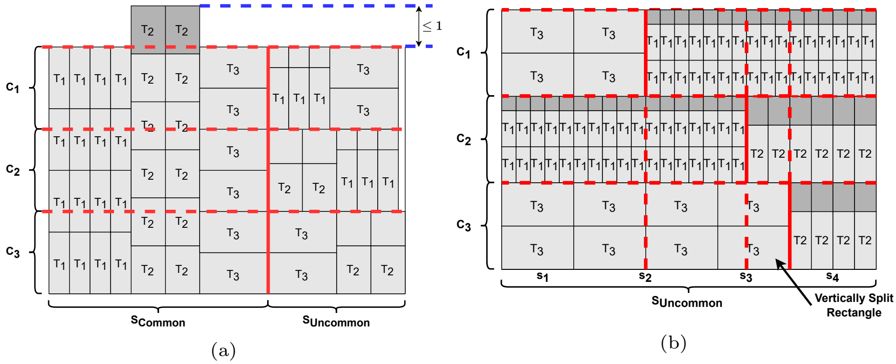


Fig. 2: (a) Rounding the fractional rectangles in S_{Common} increases the height of the packing by at most 1. (b) Within each configuration, the rectangles in $S_{Uncommon}$ are sorted according to their fractional values.

The fractional rectangles in S_{Common} are rounded up to form whole rectangles, increasing the height of the packing by at most 1 (see Figure 2a). In the sequel, we discuss only how to round the fractional rectangles in $S_{Uncommon}$.

Within each configuration, we place the fractional rectangles in $S_{Uncommon}$ at the top of the configuration. Let r be a fractional rectangle. The ratio between the height of r and the height of a rectangle of the same type as r is called the *fractional value* of r. We sort the rectangles so that fractional rectangles are sorted in non-decreasing order of their fractional values (see Figure 2b).

We draw a vertical line at each point where two rectangles of different types are packed side-by-side within a configuration. These vertical lines partition $S_{Uncommon}$ into *vertical sections* (see Figure 2b). Vertical sections are indexed from left to right starting at index 1 for the leftmost section. Within some vertical section s_i, let $C_{1(i)}$, $C_{2(i)}$, and $C_{3(i)}$ refer to the part of C_1, C_2, and C_3 that is located within s_i, respectively.

3.2 Grouping Vertical Sections

Within a vertical section s_i, each configuration has a single rectangle type. Let $f_{1(i)}$, $f_{2(i)}$, and $f_{3(i)}$ represent the fractional values of the fractional rectangles packed in s_i at the top of C_1, C_2, and C_3, respectively.

We classify the vertical sections $s_i \in S_{Uncommon}$ into three cases, depending on the three fractional values $f_{1(i)}$, $f_{2(i)}$, and $f_{3(i)}$ as follows:

- S_{Case1} includes all sections s_i such that $f_{1(i)} + f_{2(i)} + f_{3(i)} \leq 1$.
- S_{Case2} includes all sections s_i such that $f_{1(i)} + f_{2(i)} + f_{3(i)} > 1$ and either $f_{1(i)} + f_{2(i)} \leq 1$, $f_{1(i)} + f_{3(i)} \leq 1$, or $f_{2(i)} + f_{3(i)} \leq 1$.
- S_{Case3} includes all sections s_i such that $f_{1(i)} + f_{2(i)} > 1$, $f_{1(i)} + f_{3(i)} > 1$, and $f_{2(i)} + f_{3(i)} > 1$. Note that for each $s_i \in S_{Case3}$

$$f_{1(i)} + f_{2(i)} + f_{3(i)} > \frac{3}{2} \tag{2}$$

We denote with $B_{i,j}$, for $i,j = 1,2,3$, a vertical division that separates two adjacent vertical sections belonging one to S_{Casei} and the other to S_{Casej}. For example, in Figure 3a the rectangles in C_1 define $B_{1,1}$, the rectangles in C_2 define $B_{2,3}$, and the rectangles in C_3 define $B_{1,2}$. A rectangle r might intersect vertical sections of two or more cases; hereafter, we call such a rectangle a *vertically split* rectangle (see the rectangle with the arrow in Figure 2b).

4 Algorithm for 2DHMSPP with Three Rectangle Types and Two Rectangle Types Per Configuration

We assume for now that within $S_{Uncommon}$ each configuration contains exactly two different rectangle types. We will show later how to deal with the other cases.

4.1 Ordering the Configurations

We order the configurations as follows:

- If S_{Case3} is empty or S_{Case2} is empty then order the configurations so that the rectangles in the bottom configuration define $B_{1,2}$ or $B_{1,3}$, respectively.
- Otherwise order the configurations so that the rectangles in the middle configuration define $B_{2,3}$ and if S_{Case1} and S_{Case2} are not empty the rectangles in the bottom configuration must define $B_{1,2}$. Note that the rectangles in the middle configuration cannot define $B_{2,3}$ and $B_{1,2}$ because the middle configuration has only rectangles of two different types.

After ordering the configurations as above, let the configuration packed at the top be C_1, the one in the middle be C_2, and the one at the bottom be C_3. If we re-order the configurations later on, we will not re-name them; for example, if we re-order the configurations such that C_1 and C_3 swap positions, then C_1 would now be on the bottom.

Having the rectangles in C_2 define $B_{2,3}$, if possible, allows flexibility for shifting the rectangles in $C_2 \cap S_{Case3}$ as we show; for some of our algorithm's cases we shift these rectangles downwards into empty space if the rectangles in $C_3 \cap S_{Case3}$ take up less height than the rectangles in $C_3 \cap S_{Case2}$. Therefore, ordering the configurations in the manner described above is important to our algorithm.

Let a and b be the fractional values of the leftmost and rightmost fractions in C_1, respectively. Let c and d be the fractional values of the leftmost and rightmost fractions in C_2, and let e and f be the fractional values of the leftmost and rightmost fractions in C_3, respectively (see Figure 3).

We use a variable called *count* to track how many wide rectangle types appear in a packing, where a rectangle type is considered to be wide if it is the leftmost type in its configuration and it is packed, at least partially, within S_{Case2} or S_{Case3}. The presence (or absence) of these wide rectangle types is important in deciding whether we can re-use the empty space left behind when fractional rectangles are shifted around in S_{Case1} and S_{Case2} as we later explain. We initialize variable *count* to 0. If any fractional rectangles with fractional value a are packed within any vertical section of S_{Case2} or S_{Case3}, we increase the value of *count* by one. If any fractional rectangles with fractional value c are packed within any section of S_{Case2} or S_{Case3}, we increase the value of *count* by one.

4.2 Pairing Configurations

Our algorithm for rounding fractional rectangles sometimes needs to *pair* the two configurations at the top of the packing. To explain how configurations are packed, assume that C_1 is the configuration at the top of the packing and C_2 is the middle configuration. When pairing C_1 with C_2 (see Figure 3a), we flip C_1 upside down. Let F_1 be the set of fractional rectangles in each vertical section $s_i \in S_{Case1}$, and let F_2 be the set of fractional rectangles from C_1 and C_2 in each vertical section $s_i \in S_{Case2}$ where $f_{1(i)} + f_{2(i)} \leq 1$. We remove the sets F_1 and F_2 from their original positions in the packing. If $F_1 \cup F_2$ is not empty we shift up the remaining rectangles in C_1 so that the tops of the topmost rectangles in C_1 lie on a common line and the distance between C_1 and C_2 in vertical section s_1 is 1. This creates a region in S_{Case1} and S_{Case2} of height at most 1 between C_1 and C_2 where we will pack F_1 and F_2; we call this region C_{A1} (see Figure 3a). If $F_1 \cup F_2$ is empty, then region C_{A1} has initial height zero, but its height might be increased later as explained below.

We re-shape each fractional rectangle $r \in F_1 \cup F_2$ so that its area does not change but it has the full height of a rectangle of the same type as r.

Lemma 1 *Let C_1 and C_2 be paired as described above. The re-shaped fractional rectangles in $F_1 \cup F_2$ can be packed in region C_{A1}.*

Proof Let vertical section $s_i \in S_{Case1}$ have width W_i and let $C_{A1(i)}$ be the part of C_{A1} within s_i. The total empty area A_i in $C_{A1(i)}$ is $A_i \geq W_i * 1 = W_i$. Since each of $C_{1(i)}$, $C_{2(i)}$, and $C_{3(i)}$ has only one fractional rectangle type, the total area a_i of the fractional rectangles in $C_{1(i)}$, $C_{2(i)}$, and $C_{3(i)}$ is

$$a_i \leq (W_i * f_{1(i)}) + (W_i * f_{2(i)}) + (W_i * f_{3(i)}) \leq W_i \leq A_i,$$

as the height of each rectangle is at most 1 and $f_{1(i)} + f_{2(i)} + f_{3(i)} \leq 1$ for $s_i \in S_{Case1}$.

A similar argument can be made for the vertical sections $s_i \in S_{Case2}$ for which $f_{1(i)} + f_{2(i)} \leq 1$. □

Corollary 1 *After re-shaping the fractional rectangles in $F_1 \cup F_2$ we can pack them in C_{A1} so that there is at most one fractional rectangle of each type in C_{A1}.*

Proof We combine the fractional rectangles in $F_1 \cup F_2$ such that a whole rectangle is formed whenever a sufficient number of pieces of the same type have been packed. When fractional rectangles of the same type do not form a whole rectangle, they merge to become one larger fractional rectangle. Therefore, at most one fractional rectangle of each type may remain. By Lemma 1 the rectangles can be packed in C_{A1}. □

We *round up* the fractional rectangles from C_1 and C_2 in each vertical section $s_i \in S_{Case2}$ where $f_{1(i)} + f_{2(i)} > 1$ and for each vertical section $s_i \in S_{Case3}$. Rounding up a fractional rectangle r means replacing it with a whole rectangle of the same type as r and shifting rectangles up as needed to make room for the whole rectangle. When shifting rectangles from C_1 we need ensure that the tops of the topmost rectangles in C_1 lie on a common line. Finally, we round up the fractional rectangles in $C_3 \cap S_{Case2}$ (see Figure 3a).

Note that after pairing two configurations and re-shaping rectangles some whole rectangles might be vertically split by the boundaries $B_{1,2}$ and $B_{2,3}$. Because of the way in which region C_{A1} was defined, the two pieces of a whole rectangle that is vertically split by any of those boundaries are placed side-by-side forming a whole rectangle. However, pieces of fractional rectangles that are vertically split might be placed in different parts of the packing. Later we show how to shift these fractional pieces to form whole rectangles.

4.3 Rounding Fractional Rectangles

We provide different algorithms for rounding fractional rectangles into whole ones based on which of S_{Case1}, S_{Case2}, and S_{Case3} are not empty and what the value of *count* is.

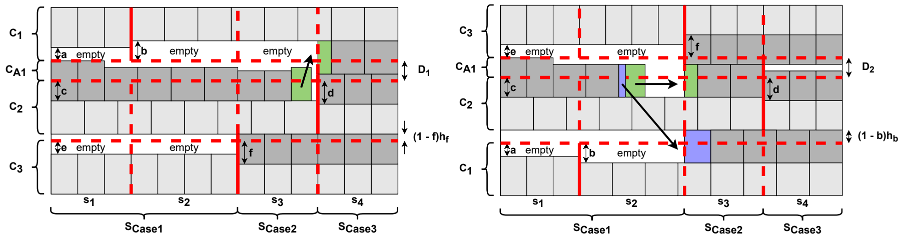
Lemma 2 *If none of S_{Case1}, S_{Case2}, and S_{Case3} are empty, then count > 0.*

Proof Assume that $count = 0$ and none of S_{Case1}, S_{Case2}, and S_{Case3} are empty. Because of how we ordered the configurations, the rectangles in C_2 define the boundary $B_{2,3}$ and therefore at least one of the fractional rectangles in C_2 with fractional value c must be packed in S_{Case2}, contradicting the assumption that $count = 0$. □

By Lemma 2, we do not need consider the case when none of S_{Case1}, S_{Case2}, and S_{Case3} are empty and $count = 0$. The cases we must consider are described below.

For simplicity and without loss of generality, in the sequel we assume that none of the configurations computed by solving linear program (1) contain any empty space, so the width of the base configuration of each configuration C is equal to 1. Additionally, we assume that for some configuration C, if its leftmost and rightmost rectangle types t_1 and t_2 are both in $S_{Case1} \cup S_{Case2}$, then $f_1 h_1 < f_2 h_2$ where f_1 and f_2 are the fractional values of the fractional rectangles of type t_1 and t_2 respectively and h_1 and h_2 are the corresponding heights of the whole rectangles. Note that if the opposite is true the analysis is very similar, so we omit it.

Let h_i be the height of the rectangles corresponding to fractional value i, for $i = a$, b, c, d, e, and f.



(a) (b)

Fig. 3: $count = 1$ and $f_{1(i)} + f_{2(i)} \leq 1$ for all vertical sections $s_i \in S_{Case2}$.

4.4 None of S_{Case1}, S_{Case2}, and S_{Case3} are empty, $count = 1$, and $f_{1(i)} + f_{2(i)} \leq 1$ for all vertical sections $s_i \in S_{Case2}$.

Lemma 3 *If none of S_{Case1}, S_{Case2}, and S_{Case3} are empty, $count = 1$, and $f_{1(i)} + f_{2(i)} \leq 1$ for all vertical sections $s_i \in S_{Case2}$, then there is an algorithm that produces an integer packing of height at most $\frac{3}{2}$ plus the value of the solution for linear program (1).*

Proof Our algorithm will produce two solutions and choose the one with shorter height. For the first solution, pair C_1 and C_2 and re-shape, pack, and round rectangles as explained in Section 4.2 (see Figure 3a). The height increase in S_{Case1} and S_{Case2} caused by creating C_{A1} is at most $h_1 - ah_a - ch_c \leq h_1 - ch_c$, where $h_1 = max\{h_a, h_b, h_c, h_e\}$ (note that C_{A1} re-uses the space that was occupied by the fractional rectangles of fractional values a and c).

In S_{Case3}, the height increase caused by rounding up the fractional rectangles with fractional values b and d is at most $(1-b)h_b+(1-d)h_d$; hence the height increase caused by pairing C_1 and C_2 is at most $D_1 = max\{h_1-ch_c, (1-b)h_b+(1-d)h_d\}$. The height increase caused by rounding up the fractional rectangles in C_3 with fractional value f is at most $(1-f)h_f$ (see Figure 3a). Therefore, the total height increase is at most $max\{\Delta_A, \Delta_B\}$, where $\Delta_A = h_1 - ch_c + (1-f)h_f \leq 2 - f - ch_c$, as $h_1 \leq 1$ and $h_f \leq 1$ and $\Delta_B = (1-b)h_b + (1-d)h_d + (1-f)h_f \leq 3 - b - d - f < \frac{3}{2}$ as $h_b \leq 1, h_d \leq 1$, and $b + d + f > \frac{3}{2}$ by (2).

For the second solution, re-order the configurations so that fractional rectangles with fractional value a appear in the bottom configuration, and fractional rectangles with fractional value e appear in the top configuration, then pair C_2 and C_3 (note that these are now the top two configurations) and re-shape, pack, and round rectangles as explained in Section 4.2 (see Figure 3b). We only consider the case when $f+c > 1$ (see Figure 3b); the case when $f + c \leq 1$ is similar.

The height increase caused by creating C_{A1} is at most $h_2-ch_c-eh_e$, where $h_2 = max\{h_a, h_b, h_c, h_e\}$. In S_{Case2} and S_{Case3}, the height increase caused by rounding up the fractional rectangles with fractional values c, d, and f is at most $max\{(1-c)h_c, (1-d)h_d\}+(1-f)h_f$; hence the height increase caused by pairing C_2 and C_3 is at most $D_2 = max\{h_2-ch_c-eh_e, max\{(1-c)h_c, (1-d)h_d\}+(1-f)h_f\}$. The height increase caused by rounding up fractional rectangles in C_1 with fractional value b

is at most $(1-b)h_b$. Therefore, the total height increase is at most $max\{\Delta_C, \Delta_D\}$, where $\Delta_C = (1-b)h_b + h_2 - ch_c - eh_e$ and $\Delta_D = (1-b)h_b + max\{(1-c)h_c, (1-d)h_d\} + (1-f)h_f$.

Selecting the better of the two solutions produces an increase in the height of the solution by $max\{min\{\Delta_A, \Delta_C\}, min\{\Delta_A, \Delta_D\}, min\{\Delta_B, \Delta_C\}, min\{\Delta_B, \Delta_D\}\}$.

- $min\{\Delta_A, \Delta_C\}$: $\Delta_A = 2-f-ch_c$ and $\Delta_C = (1-b)h_b+max\{h_c, h_e\}-ch_c-eh_e$. Note that $\Delta_A \leq 2-f$ and since $h_b, h_c, h_e \leq 1$ then $\Delta_C \leq (1-b)+(1-e) = 2-b-e$. Since $f+b > 1$ as fractional rectangles with fractional values f and b appear in S_{Case3} then either $f > \frac{1}{2}$ or $b > \frac{1}{2}$ and so $min\{\Delta_A, \Delta_C\} \leq \frac{3}{2}$.
- $min\{\Delta_A, \Delta_D\}$: $\Delta_A = 2-f-ch_c$ and $\Delta_D = (1-b)h_b + max\{(1-c)h_c, (1-d)h_d\} + (1-f)h_f$. Recall our assumption that $f+c > 1$ (the case when $f+c \leq 1$ is similar), therefore either $f > \frac{1}{2}$ or $c > \frac{1}{2}$. If $f > \frac{1}{2}$ then $\Delta_A < \frac{3}{2} - ch_c < \frac{3}{2}$. If $c > \frac{1}{2}$ then $\Delta_D \leq (1-b)+(1-f)+max\{1-c, 1-d\} = 2-b-f+max\{$1 - c, 1 - d$\}$:
  - If $1-c > 1-d$ then $\Delta_D \leq 3-b-f-c < 3-\frac{1}{2}-b-f < \frac{3}{2}$ as $b+f > 1$.
  - If $1-d > 1-c$ then $\Delta_D \leq 3-b-d-f \leq \frac{3}{2}$ by (2).

  Therefore, $min\{\Delta_A, \Delta_D\} \leq \frac{3}{2}$.
- $min\{\Delta_B, \Delta_C\} \leq \frac{3}{2}$ and $min\{\Delta_B, \Delta_D\} \leq \frac{3}{2}$ because $\Delta_B \leq \frac{3}{2}$.

□

Observe that in the first solution, depicted in Figure 3a, there might be a fractional rectangle r in C_1 that is vertically split by $B_{2,3}$ such that one piece of r is re-shaped and packed as explained in Section 4.2, while the other piece is rounded up to the height of a rectangle of the same type as r. These pieces are marked in Figure 3a. Note that the two pieces can be placed beside each other to form a whole rectangle without further increasing the height of the packing. Similarly the fractional rectangles in Figure 3b can be combined to form whole rectangles without affecting the height of the packing. In the sequel we will not explicitly explain how fractional rectangles that are vertically split are combined to form whole rectangles, instead the figures will show how to do this.

4.5 None of S_{Case1}, S_{Case2}, and S_{Case3} are empty, $count = 1$, and $f_{1(i)} + f_{2(i)} > 1$ for at least one vertical section $s_i \in S_{Case2}$.

Lemma 4 *If none of S_{Case1}, S_{Case2}, and S_{Case3} are empty and $count = 1$, then C_1's rectangles cannot define $B_{2,2}$, $B_{2,3}$, or $B_{3,3}$.*

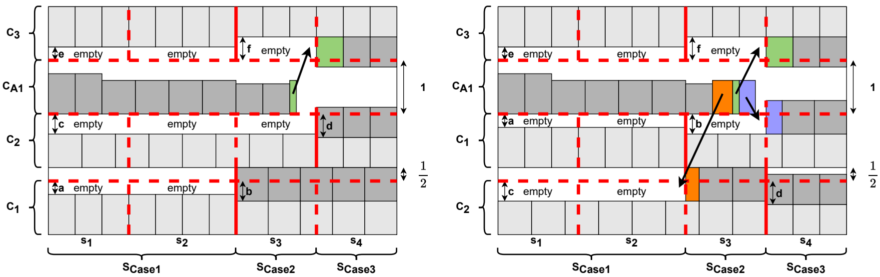
Proof Note that if C_1's rectangles defined $B_{2,2}$, then the value of *count* would be 2 because rectangles in C_1 with fractional value a would appear in S_{Case2} and since the rectangles in C_2 define $B_{2,3}$ then rectangles with fractional value c would also appear in S_{Case2}. Similarly, it is not possible that C_1's rectangles define boundaries $B_{2,3}$ or $B_{3,3}$. □

Lemma 5 *If none of S_{Case1}, S_{Case2}, and S_{Case3} are empty, $count = 1$, and $f_{1(i)} + f_{2(i)} > 1$ for at least one $s_i \in S_{Case2}$, then there is an algorithm that produces an integer packing of height at most $\frac{3}{2}$ plus the value of the solution for linear program (1).*

Proof By Lemma 4, C_1's rectangles cannot define $B_{2,2}$, $B_{2,3}$, or $B_{3,3}$. Note that if C_1's rectangles defined $B_{1,1}$, then $f_{1(i)} + f_{2(i)} \leq 1$ for all vertical sections $s_i \in S_{Case2}$ since the rectangles in C_2 define $B_{2,3}$ and thus fractional rectangles with fractional values b and c would appear within S_{Case1} and so $b+c$ would be at most 1. Therefore, the rectangles in C_1 and C_3 must create a coinciding boundary $B_{1,2}$ so that $b + c$ could be larger than 1, as required by the Lemma.

Since $b+c > 1$, then $b > \frac{1}{2}$ and/or $c > \frac{1}{2}$. If $b > c$, then re-order the configurations so that fractional rectangles with fractional value b appear in the bottom configuration. Pair C_2 and C_3 and re-shape, pack, and round rectangles as explained Section 4.2. The height increase caused by pairing C_2 and C_3 is at most 1: for sections s_i where $f_{1(i)} + f_{2(i)} \leq 1$ the height increase caused by creating C_{A1} is at most 1, and for sections s_i where $f_{1(i)} + f_{2(i)} > 1$ the height increase caused by rounding up $f_{1(i)}$ and $f_{2(i)}$ is also at most 1. The height increase caused by rounding up fractional rectangles with fractional value b is at most $\frac{1}{2}$ (see Figure 4a) so the total height increase is at most $\frac{3}{2}$.

If $c > b$, then re-order the configurations so that fractional rectangles with fractional value c appear in the bottom configuration, pair C_1 and C_3, and re-shape, pack, and round rectangles as explained in Section 4.2. The height increase caused by pairing C_1 and C_3 is at most 1 and the height increase caused by rounding up fractional rectangles with fractional value c is at most $\frac{1}{2}$ (see Figure 4b). □



(a) (b)

Fig. 4: *count* $= 1$ and $f_{1(i)} + f_{2(i)} > 1$ for at least one vertical section $s_i \in S_{Case2}$.

4.6 None of S_{Case1}, S_{Case2}, and S_{Case3} are empty, $count = 2$, and $f_{1(i)} + f_{2(i)} \leq 1$ for all vertical sections $s_i \in S_{Case2}$.

Lemma 6 *If none of S_{Case1}, S_{Case2}, and S_{Case3} are empty, $count = 2$, and $f_{1(i)} + f_{2(i)} \leq 1$ for all vertical sections $s_i \in S_{Case2}$, then there is an algorithm that produces an integer packing of height at most $\frac{3}{2}$ plus the value of the solution for linear program (1).*

Proof Note that the rectangles in C_1 cannot create $B_{1,1}$ as otherwise $count < 2$.

- If $(1-f)h_f \leq \frac{1}{2}$ then re-order the configurations so that fractional rectangles with fractional value f appear in the bottom configuration. If $(1-f)h_f > \frac{1}{2}$:
  - If $(1-a)h_a \leq \frac{1}{2}$ re-order the configurations so that fractional rectangles with fractional value a appear in the bottom configuration.
  - If $(1-c)h_c \leq \frac{1}{2}$ re-order the configurations so that fractional rectangles with fractional value c appear in the bottom configuration.

  Pair the top two configurations, and re-shape, pack, and round fractional rectangles as explained in Section 4.2. These solutions are very similar as that in Figure 4a. The height increase caused by pairing the top two configurations is at most 1. The height increase caused by rounding up the fractional rectangles in the bottom configuration is at most $\frac{1}{2}$. To see this note that if $(1-f)h_f > \frac{1}{2}$ then $h_f > \frac{1}{2}$ and $f < \frac{1}{2}$ as $h_f \leq 1$; furthermore, $b > \frac{1}{2}$ and $d > \frac{1}{2}$ since $b + f > 1$ and $d + f > 1$ (as fractional values b, d, and f appear in S_{Case3}). So the total height increase is at most $\frac{3}{2}$.
- If $(1-f)h_f > \frac{1}{2}$, $(1-c)h_c > \frac{1}{2}$, and $(1-a)h_a > \frac{1}{2}$, then $h_a > \frac{1}{2}$, $h_c > \frac{1}{2}$, and $h_f > \frac{1}{2}$. Pair C_1 and C_2 and re-shape, pack, and round fractional rectangles as explained in Section 4.2 (see Figure 5). The height increase in S_{Case1} and S_{Case2} caused by creating C_{A1} is at most $h_1 - ah_a - ch_c$, where $h_1 = max\{h_a, h_b, h_c, h_e\}$. In S_{Case3}, the height increase caused by rounding up the fractional rectangles with fractional values b and d is at most $(1-b)h_b + (1-d)h_d$; hence the height increase caused by pairing C_1 and C_2 is at most $D_1 = max\{h_1 - ah_a - ch_c, (1-b)h_b + (1-d)h_d\}$. The height increase caused by rounding up the fractional rectangles in C_3 with fractional value f is at most $(1-f)h_f$. Therefore, the total height increase is at most $max\{\Delta_A, \Delta_B\}$, where:
  - $\Delta_A = h_1 - ah_a - ch_c + (1-f)h_f = h_1 + h_f - (ah_a + ch_c + fh_f) \leq 2 - \frac{1}{2}(a+c+f) < \frac{3}{2}$, as $h_1 \leq 1$, $h_a > \frac{1}{2}$, $h_c > \frac{1}{2}$, $1 \geq h_f > \frac{1}{2}$, and $a + c + f > 1$, and
  - $\Delta_B = (1-b)h_b + (1-d)h_d + (1-f)h_f \leq (1-b) + (1-d) + (1-f) = 3 - b - d - f < \frac{3}{2}$ as $h_b \leq 1, h_d \leq 1$, $h_f \leq 1$, and $b + d + f > \frac{3}{2}$ by (2).

□

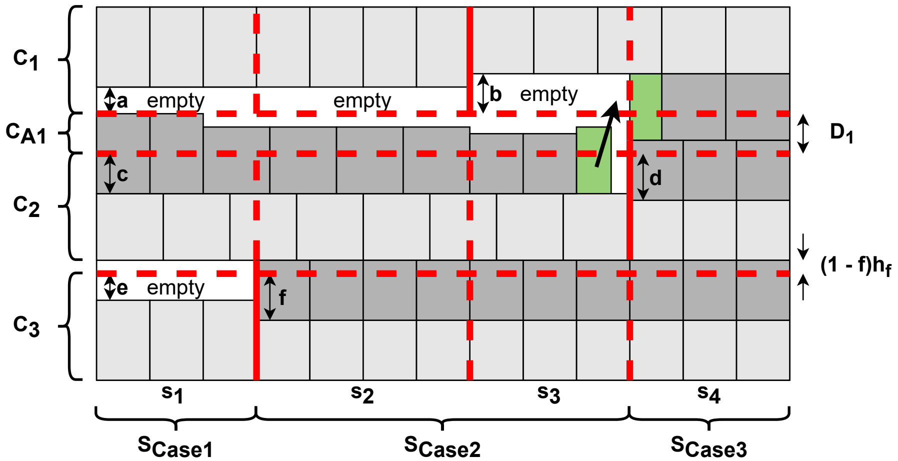


Fig. 5: *count* $= 2$, $f_{1(i)} + f_{2(i)} \leq 1$ for all vertical sections $s_i \in S_{Case2}$, $(1-f)h_f > \frac{1}{2}$, $(1-c)h_c > \frac{1}{2}$, and $(1-a)h_a > \frac{1}{2}$.

4.7 None of S_{Case1}, S_{Case2}, and S_{Case3} are empty, $count = 2$, and $f_{1(i)} + f_{2(i)} > 1$ for at least one vertical section $s_i \in S_{Case2}$.

Lemma 7 *If none of S_{Case1}, S_{Case2}, and S_{Case3} are empty, $count = 2$, and $f_{1(i)} + f_{2(i)} > 1$ for at least one vertical section $s_i \in S_{Case2}$, then there is an algorithm that produces an integer packing of height at most $\frac{3}{2}$ plus the value of the solution for linear program (1).*

Proof Note that if C_1's rectangles defined $B_{2,3}$ or $B_{3,3}$, then $f_{1(i)} + f_{2(i)} \leq 1$ for all vertical sections $s_i \in S_{Case2}$ since the rectangles in C_2 define $B_{2,3}$ and therefore rectangles with fractional values a and c would appear within S_{Case1} (so $a + c \leq 1$) and they would be the only fractional values in $S_{Case2} \cap (C_1 \cup C_2)$. Additionally, note that C_1's rectangles cannot define $B_{1,1}$ or $B_{1,2}$, as otherwise *count* could not have value 2. Therefore, C_1's rectangles must define boundary $B_{2,2}$ so that $f_{1(i)} + f_{2(i)} > 1$ for at least one vertical section $s_i \in S_{Case2}$ and $b + c > 1$, as required by the Lemma.

Similar to the analysis in the proof of Lemma 6, if $(1-f)h_f \leq \frac{1}{2}$, or if $(1-f)h_f > \frac{1}{2}$ and $(1-c)h_c \leq \frac{1}{2}$ or $(1-a)h_a \leq \frac{1}{2}$, then we re-order the configurations so that the fractional value f, c, or a appears in the bottom configuration, respectively. The height increase caused by pairing the top two configurations is at most 1. The height increase caused by rounding up fractional rectangles in the bottom configuration is at most $\frac{1}{2}$, and so the total height increase is at most $\frac{3}{2}$.

Hence, we only need to consider the case when $(1-f)h_f > \frac{1}{2}$, $(1-c)h_c > \frac{1}{2}$, and $(1-a)h_a > \frac{1}{2}$. Note that then $f < \frac{1}{2}$, $c < \frac{1}{2}$, $a < \frac{1}{2}$, $h_c > \frac{1}{2}$, $h_a > \frac{1}{2}$, and $b > \frac{1}{2}$ as $b + c > 1$. Consider the fractional values a, c, and f, and re-order the configurations so that the two largest fractional values among them are in the bottom and middle configurations. If $a \geq c \geq f$ or $c \geq a \geq f$, then ensure that fractional value a appears in the bottom configuration. If $a \geq f \geq c$, $f \geq a \geq c$, $c \geq f \geq a$, or $f \geq c \geq a$, then ensure that fractional value f appears in the bottom configuration. Pair the top two configurations, and re-shape, pack, and round fractional rectangles as explained in Section 4.2. We consider below just the case when $a \geq c \geq f$; the other cases are similar.

Observe that $a + c > \frac{2}{3}$ as $a + c + f > 1$ (see Figure 6). The height increase in S_{Case1} and S_{Case2} caused by creating C_{A1} is at most $h_1 - ch_c - eh_e \leq h_1 - ch_c$, where $h_1 = max\{h_a, h_c, h_e, h_f\}$. In S_{Case3}, the height increase caused by rounding up the fractional rectangles with fractional values d and f is at most $(1-d)h_d + (1-f)h_f$.

Note that $(1-a)h_a > (1-b)h_b$, as $b > \frac{1}{2}$ and so $(1-b)h_b \leq \frac{1}{2}$ but $(1-a)h_a > \frac{1}{2}$. Thus the fractional rectangles with fractional value d (and the whole rectangles of the same type beneath them in the middle configuration) can be shifted downwards into the empty space above the fractional rectangles with fractional value b (see Figure 6). Hence, the height increase caused from pairing the top two configurations is at most $D_1 = max\{h_1 - ch_c, (1-d)h_d + (1-f)h_f - ((1-a)h_a - (1-b)h_b)\}$.

The height increase caused by rounding up the fractional rectangles in the bottom configuration with fractional value a is at most $(1-a)h_a$. Therefore, the total height increase is at most $max\{\Delta_A, \Delta_B\}$, where:

- $\Delta_B = (1-d)h_d + (1-f)h_f - ((1-a)h_a - (1-b)h_b) + (1-a)h_a \leq (1-d) + (1-f) + (1-b) = 3 - b - d - f \leq \frac{3}{2}$ by (2), as h_b, h_d, and h_f are at most 1.
- $\Delta_A = h_1 - ch_c + (1-a)h_a \leq 2 - a - ch_c$ as $h_1 \leq 1$ and $h_a \leq 1$. Since Δ_A is a decreasing function on $a + c$ and $a + c > \frac{2}{3}$ then an upper bound for the value of Δ_A can be obtained when $a + c = \frac{2}{3}$ and so $a = \frac{2}{3} - c$, therefore $\Delta_A \leq 2 - \frac{2}{3} + c - ch_c = \frac{4}{3} + c - ch_c < \frac{4}{3} + c - \frac{c}{2(1-c)}$ because $(1-c)h_c > \frac{1}{2}$. Then $\Delta_A < \frac{4}{3} + \frac{c - 2c^2}{2(1-c)}$. The right hand side of this inequality takes its maximum value when $c = 1 - \frac{\sqrt{2}}{2}$ and so $\Delta_A < \frac{4}{3} + \frac{3}{2} - \sqrt{2} = \frac{17}{6} - \sqrt{2} < \frac{3}{2}$. □

4.8 Remaining Cases

In the previous section we considered the case when none of S_{Case1}, S_{Case2}, or S_{Case3} was empty. We briefly discuss below the remaining cases:

- If S_{Case1} is not empty, but S_{Case2} and S_{Case3} are both empty, pair C_1 and C_2 re-shape and pack all fractional rectangles in C_{A1} as explained in Section 4.2. Therefore we obtain a solution of height at most 1 plus the value of the solution for linear program (1).
- If S_{Case2} is not empty, but S_{Case1} and S_{Case3} are both empty, then $count > 0$ and we can use Lemmas 3-7.
- If S_{Case3} is not empty, but S_{Case1} and S_{Case2} are both empty, round up all fractional rectangles. Since $f_{1(i)} + f_{2(i)} + f_{3(i)} > \frac{3}{2}$ for every vertical section $s_i \in S_{Case3}$ then we obtain a solution of height at most $\frac{3}{2}$ plus the value of the solution for linear program (1).
- If both S_{Case1} and S_{Case2} are not empty, but S_{Case3} is empty, and $count = 0$, there must be only one vertical section $s_i \in S_{Case2}$ as otherwise the boundary $B_{2,2}$ must exist, but that would mean that at least one fractional

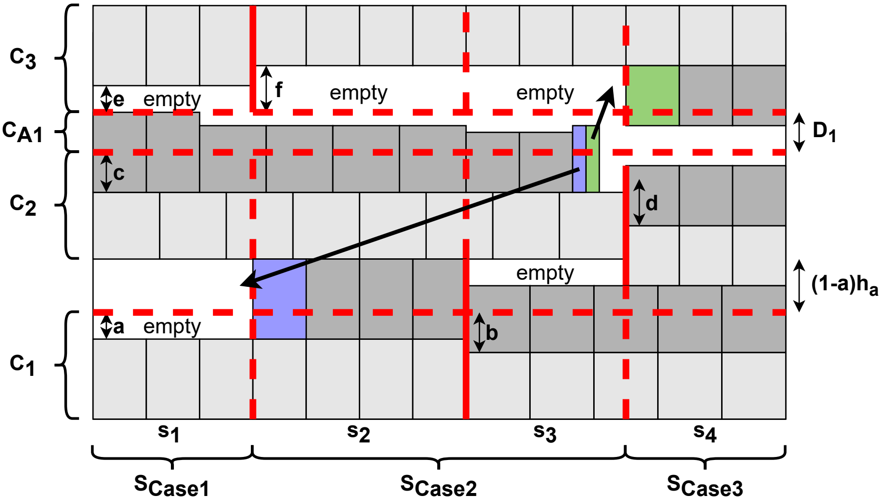
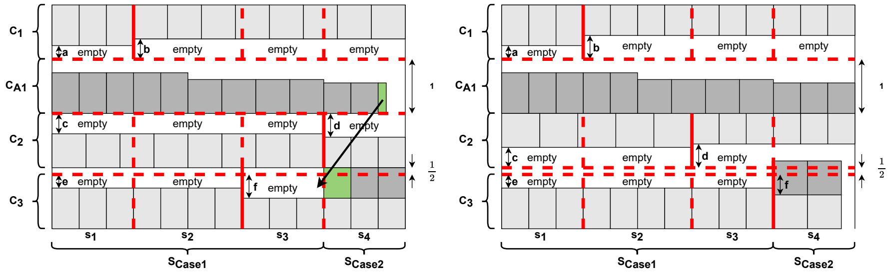


Fig. 6: $count = 2$, $f_{1(i)} + f_{2(i)} > 1$ for at least one vertical section $s_i \in S_{Case2}$, $(1-f)h_f > \frac{1}{2}$, $(1-c)h_c > \frac{1}{2}$, $(1-a)h_a > \frac{1}{2}$, and $a \geq c \geq f$.



(a) (b)

Fig. 7: If $S = S_{Case1} \cup S_{Case2}$ and $count = 0$, then there is exactly one vertical section $s_i \in S_{Case2}$. (a) The largest fraction is more than $\frac{1}{2}$, and (b) the largest fraction is less than $\frac{1}{2}$.

rectangle with fractional value a or c must be within S_{Case2} and therefore $count$ would have value larger than zero.

– If the largest fractional value in the section $s_i \in S_{Case2}$ is more than $\frac{1}{2}$, re-order the configurations so that the fractional rectangles with that fractional value appear in the bottom configuration and then pair the top two configurations, and re-shape, pack, and round fractional rectangles as explained in Section 4.2 (see Figure 7a). The height increase caused by creating C_{A1} is at most 1 and the height increase caused by rounding up the fractional rectangles in the bottom configuration is at most $\frac{1}{2}$.

- Otherwise, re-order the configurations such that for the section $s_i \in S_{Case2}$ the fractional value in the top configuration in S_{Case2} is the smallest, and the boundary defined by the rectangles in the bottom configuration does not occur to the left of the boundary defined by the rectangles in the middle configuration. Pair the top two configurations and re-shape, pack, and round fractional rectangles as explained in Section 4.2, then flip the middle configuration upside down. Note that the rounded-up rectangles in S_{Case2} in the bottom configuration can use the empty space left behind by the fractional rectangles in the middle configuration (see Figure 7b).

  The height increase caused by creating C_{A1} is at most 1. Note that since the middle configuration is flipped upside down, the empty space leftover after removing the fractional rectangles from the middle configuration can be used by the rounded up rectangles in the bottom configuration; therefore, the height increase caused by rounding up the fractional rectangles in the bottom configuration is at most $\frac{1}{2}$, as the fractional values the middle and bottom configurations sum to more than $\frac{1}{2}$, so the total height increase is at most $\frac{3}{2}$.

- If both S_{Case1} and S_{Case2} are not empty, but S_{Case3} is empty, and $count > 0$, then we can use Lemmas 3-7.
- If both S_{Case1} and S_{Case3} are not empty, but S_{Case2} is empty, and $count = 0$, then re-order the configurations so that the fractional rectangles in the bottom configuration are the largest of fractional values b, d, and f. Note that fractional values a and c are not within S_{Case3}, as $count = 0$, and the rectangles in the bottom configuration create $B_{1,3}$, so fractional value e is not within S_{Case3} either. Pair the top two configurations, and re-shape, pack, and round fractional rectangles as explained in Section 4.2. The height increase from pairing the top two configurations is at most 1, and the height increase from rounding up the fractional rectangles in S_{Case3} in the bottom configuration is at most $\frac{1}{2}$, so the total height increase is at most $\frac{3}{2}$.
- If both S_{Case1} and S_{Case3} are not empty, but S_{Case2} is empty, and $count > 0$, then we can use Lemmas 3-7.
- If both S_{Case2} and S_{Case3} are not empty, but S_{Case1} is empty, then $count = 2$, and we can use Lemmas 3-7.

Theorem 1 *If $K = 3$ and the fractional solution computed by solving linear program (1) has exactly three configurations, and if each of those configurations has exactly two different rectangle types in $S_{Uncommon}$, then there is an algorithm that produces an integer packing of height at most $\frac{3}{2}$ plus the value of the solution for linear program (1).*

5 Algorithm for Three Configurations and Three Rectangle Types in a Configuration

In this section we consider the case when the fractional solution obtained from solving linear program (1) has three configurations, one configuration has exactly three rectangle types, one configuration has exactly two rectangle types, and one configuration has exactly one rectangle type. The algorithms described in this section are modifications of the algorithms described in the previous section to account for the existence of a configuration with three rectangle types. Note that only a single configuration in $S_{Uncommon}$ can pack three rectangle types, as otherwise at least one of the rectangle types would be common to all three configurations.

5.1 Ordering the Configurations

We order the configurations as follows:

- The top configuration contains only one rectangle type.
- The rectangles in the middle configuration define $B_{2,3}$, if it exists. Note that the rectangles in the middle configuration can define both $B_{1,2}$ and $B_{2,3}$ if it contains three rectangle types.

After ordering the configurations as above, let the configuration packed at the top be C_1, the one in the middle be C_2, and the one at the bottom be C_3.

Let a be the fractional value of the fractional rectangles in C_1. If C_2 contains three rectangle types, then let b, c, and d be the fractional values in C_2 and let e and f be the fractional rectangles in C_3. Otherwise, if C_2 contains two rectangle types, then let b and c be the fractional values in C_2 and let d, e, and f be the fractional rectangles in C_3.

Initialize variable *count* to 0. Increase *count* in the following way:

- If any fractional rectangles with fractional value a are packed within any vertical section of S_{Case2} or S_{Case3}, increase the value of *count* by one.
- If any fractional rectangles with fractional value b are packed within any vertical section of S_{Case2} or S_{Case3}, increase the value of *count* by one.
- If C_2 contains three rectangle types:
  - If any fractional rectangles with fractional value e are packed within any vertical section of S_{Case2} or S_{Case3}, increase the value of *count* by one.
- If C_2 contains two rectangle types:
  - If any fractional rectangles with fractional value d are packed within any vertical section of S_{Case2} or S_{Case3}, increase the value of *count* by one.

We provide different algorithms for rounding fractional rectangles into whole ones based on which of S_{Case1}, S_{Case2}, and S_{Case3} are not empty and what the value of *count* is. Note that because C_1 has only one rectangle type, if none of S_{Case1}, S_{Case2}, and S_{Case3} are empty, then $count > 0$.

5.2 None of S_{Case1}, S_{Case2}, and S_{Case3} are empty, and $count = 1$.

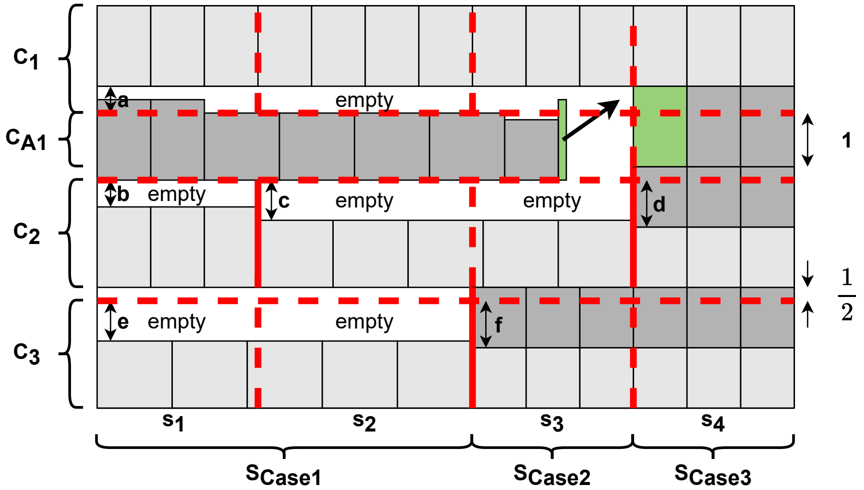


Fig. 8: None of S_{Case1}, S_{Case2}, and S_{Case3} are empty, $count = 1$, $f_{1(i)} + f_{2(i)} \leq 1$ for all vertical sections $s_i \in S_{Case2}$, and $(1-a)h_a > \frac{1}{2}$ so $f > \frac{1}{2}$.

Lemma 8 *If none of S_{Case1}, S_{Case2}, and S_{Case3} are empty, $count = 1$ and $f_{1(i)} + f_{2(i)} \leq 1$ for all vertical sections $s_i \in S_{Case2}$, then there is an algorithm that produces an integer packing of height at most $\frac{3}{2}$ plus the value of the solution for linear program (1).*

Proof Since the rectangles in C_2 define $B_{2,3}$, if C_2 had only two rectangle types then $count > 1$; hence C_2 must have three rectangle types. So, in C_2 fractional rectangles with fractional value b are in S_{Case1} (but not S_{Case2}, as otherwise $count > 1$), fractional rectangles with fractional value c are in S_{Case2} (but not S_{Case3}, as these rectangles define $B_{2,3}$), and only the fractional rectangles with fractional value d are in S_{Case3}. In C_3 fractional value e cannot be within S_{Case2} or S_{Case3}, as otherwise $count > 1$.

If $(1-a)h_a \leq \frac{1}{2}$ then re-order the configurations so that fractional rectangles with fractional value a appear in the bottom configuration; otherwise $(1-a)h_a > \frac{1}{2}$, so $a < \frac{1}{2}$ and $f > \frac{1}{2}$ because $a + f > 1$ as both fractional values appear in S_{Case3}. Pair the top two configurations, and re-shape, pack, and round fractional rectangles as explained in Section 4.2. The height increase caused by pairing the top two configurations is at most 1. The height increase caused by rounding up fractional rectangles with fractional value a (if $(1-a)h_a \leq \frac{1}{2}$) or f (if $(1-a)h_a > \frac{1}{2}$) is at most $\frac{1}{2}$ so the total height increase is at most $\frac{3}{2}$. □

5.3 None of S_{Case1}, S_{Case2}, and S_{Case3} are empty, $count = 2$, and $f_{1(i)} + f_{2(i)} \leq 1$ for all vertical sections $s_i \in S_{Case2}$.

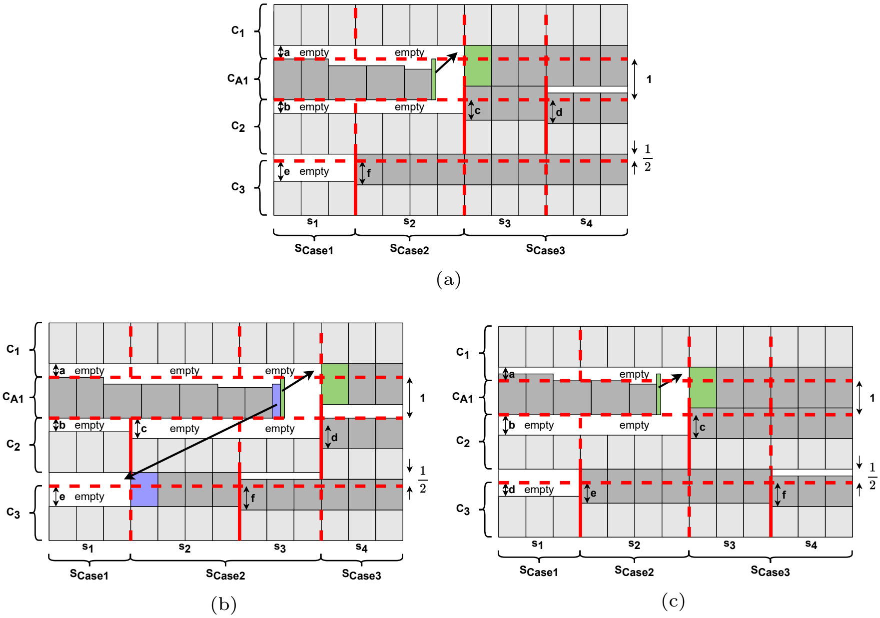


Fig. 9: $S = S_{Case1} \cup S_{Case2} \cup S_{Case3}$, $count = 2$, and $f_{1(i)} + f_{2(i)} \leq 1$ for all vertical sections $s_i \in S_{Case2}$. (a) C_2's leftmost fractional value is packed within S_{Case2} and $(1-a)h_a > \frac{1}{2}$. (b) C_3's leftmost fractional value is packed within S_{Case2} and $(1-a)h_a > \frac{1}{2}$, $(1-e)h_e < \frac{1}{2}$, and $f > \frac{1}{2}$. (c) C_2 has only two rectangle types and $(1-a)h_a > \frac{1}{2}$, $(1-e)h_e < \frac{1}{2}$, and $f > \frac{1}{2}$.

Lemma 9 *If none of S_{Case1}, S_{Case2}, and S_{Case3} are empty, $count = 2$ and $f_{1(i)} + f_{2(i)} \leq 1$ for all vertical sections $s_i \in S_{Case2}$, then there is an algorithm that produces an integer packing of height at most $\frac{3}{2}$ plus the value of the solution for linear program (1).*

Proof First assume that C_2 has three rectangle types and C_3 has two rectangle types. We need to consider two cases.

- C_2's leftmost fraction is packed within S_{Case2}. Therefore, C_2 cannot create either $B_{1,1}$ or $B_{1,2}$, C_3 must define $B_{1,2}$, and fractional value e does not appear in S_{Case2} or S_{Case3} (see Figure 9a). We process the fractional rectangles using the same approach as in Lemma 8.

- C_3's leftmost fraction is packed within S_{Case2}. Therefore, C_2 must define $B_{1,2}$ and C_3 could create either $B_{2,2}$, $B_{2,3}$, or $B_{3,3}$ (see Figure 9b). If $(1-a)h_a \le \frac{1}{2}$ then re-order the configurations so that fractional rectangles with fractional value a appear in the bottom configuration. Otherwise, if $(1-a)h_a > \frac{1}{2}$ and $(1-c)h_c \le \frac{1}{2}$ (or $(1-e)h_e \le \frac{1}{2}$), then re-order the configurations so that fractional rectangles with fractional value c (or e) appear in the bottom configuration. Pair the top two configurations, and re-shape, pack, and round fractional rectangles as explained in Section 4.2. The height increase caused by pairing the top two configurations is at most 1. The height increase caused by rounding up fractional rectangles in the bottom configuration is at most $\frac{1}{2}$. To see this note that if $(1-a)h_a > \frac{1}{2}$ then $h_a > \frac{1}{2}$ and $a < \frac{1}{2}$ as $h_a \le 1$; therefore, when this happens $d > \frac{1}{2}$ and $f > \frac{1}{2}$ since $a+d > 1$ and $a+f > 1$ (as fractional values a, d, and f appear in S_{Case3}). So the total height increase is at most $\frac{3}{2}$. Hence, we only need to consider the case when $(1-a)h_a > \frac{1}{2}$, $(1-c)h_c > \frac{1}{2}$, and $(1-e)h_e > \frac{1}{2}$, which can be addressed using the approach from the proof of Lemma 7 and it increases the height of the packing by at most $\frac{3}{2}$.

Assume now that C_2 has two rectangle types and C_3 has three rectangle types. Note that fractional rectangles in C_2 with fractional value b are packed within S_{Case2}, because the rectangles in C_2 define boundary $B_{2,3}$, and so C_3 must create $B_{1,2}$ (see Figure 9c).

Again, similar to the analysis above, if $(1-a)h_a \le \frac{1}{2}$ then re-order the configurations so that fractional rectangles with fractional value a appear in the bottom configuration. Otherwise, if $(1-a)h_a > \frac{1}{2}$ and $(1-b)h_b \le \frac{1}{2}$ (or $(1-e)h_e \le \frac{1}{2}$), then re-order the configurations so that fractional rectangles with fractional value b (or e) appear in the bottom configuration. Pair the top two configurations, and re-shape, pack, and round fractional rectangles as explained in Section 4.2. The height increase caused by pairing the top two configurations is at most 1. The height increase caused by rounding up fractional rectangles in the bottom configuration is at most $\frac{1}{2}$. To see this note that if $(1-a)h_a > \frac{1}{2}$ then $h_a > \frac{1}{2}$ and $a < \frac{1}{2}$ as $h_a \le 1$; therefore, when this happens $c > \frac{1}{2}$ and $f > \frac{1}{2}$ since $a+c > 1$ and $a+f > 1$ (as fractional values a, c, and f appear in S_{Case3}). So the total height increase is at most $\frac{3}{2}$.

Finally, when $(1-a)h_a > \frac{1}{2}$, $(1-b)h_b > \frac{1}{2}$, and $(1-e)h_e > \frac{1}{2}$, using the approach from the proof of Lemma 7 increases the height of the packing by at most $\frac{3}{2}$. □

5.4 None of S_{Case1}, S_{Case2}, and S_{Case3} are empty, $count = 2$, and $f_{1(i)} + f_{2(i)} > 1$ for at least one vertical section $s_i \in S_{Case2}$.

Lemma 10 *If $count = 2$, and $f_{1(i)} + f_{2(i)} > 1$ for at least one vertical section $s_i \in S_{Case2}$, then there is an algorithm that produces an integer packing of height at most $\frac{3}{2}$ plus the value of the solution for linear program (1).*

Proof Note that if C_2 has two rectangle types then $f_{1(i)} + f_{2(i)} \le 1$ for all vertical sections $s_i \in S_{Case2}$ since the rectangles in C_2 define boundary $B_{2,3}$ and so fractional values a and b would appear within S_{Case1} and S_{Case2} and so $a + b$ would be at most 1; therefore, C_2 must have three rectangle types.

Additionally, note that since C_2 has three rectangle types, its rectangles must also define $B_{1,2}$ or $B_{2,2}$, as otherwise $f_{1(i)} + f_{2(i)} \leq 1$ for all sections $s_i \in S_{Case2}$. The rectangles in C_2 cannot define $B_{1,1}$ as then the rectangles in C_3 would have to define $B_{1,2}$ and so *count* would have value 1.

We first consider when the rectangles in C_2 define $B_{1,2}$, which means that the rectangles in C_3 define either $B_{2,2}$, $B_{2,3}$, or $B_{3,3}$, all of which are handled the same way.

Since $a + c > 1$, then $a > \frac{1}{2}$ and/or $c > \frac{1}{2}$.

- If $a > \frac{1}{2}$ then re-order the configurations so that fractional rectangles with fractional value a appear in the bottom configuration, pair the top two configurations, and re-shape, pack, and round fractional rectangles as explained in Section 4.2. The height increase caused by pairing the top two configurations is at most 1. The height increase caused by rounding up fractional rectangles with fractional value a is at most $\frac{1}{2}$ so the total height increase is at most $\frac{3}{2}$.
- If $a < \frac{1}{2}$ then $c > \frac{1}{2}$; also $d > \frac{1}{2}$, and $f > \frac{1}{2}$ as fractional values a, d, and f appear in S_{Case3}. Re-order the configurations so that fractional rectangles with fractional value c appear in the bottom configuration, pair the top two configurations, and re-shape, pack, and round fractional rectangles as explained in Section 4.2. The height increase caused by pairing the top two configurations is at most 1. The height increase caused by rounding up fractional rectangles with fractional values c and d is at most $\frac{1}{2}$ so the total height increase is at most $\frac{3}{2}$.

For the case when the rectangles in C_2 define $B_{2,2}$, which means that the rectangles in C_3 define $B_{1,2}$, we use the same approach as in Lemma 8. □

5.5 Remaining Cases

When only S_{Case1} is not empty, or when only S_{Case3} is not empty, we can use the algorithms described in Section 4 for the same cases. Note that when only S_{Case2} or S_{Case3} are empty, then *count* > 0 since there is a configuration containing a single rectangle type, and hence the algorithms from Lemmas 8-10 can be used. When only S_{Case2} is not empty or when only S_{Case1} is empty, then *count* > 0 and the algorithms from Lemmas 8-10 can be used.

Theorem 2 *If $K = 3$ and the fractional solution computed by solving linear program (1) has exactly three configurations, one configuration has three rectangle types, one configuration has two rectangle types, and one configuration has only one rectangle type, then there is an algorithm that produces an integer packing of height at most $\frac{3}{2}$ plus the value of the solution for linear program (1).*

5.6 Fewer Than Three Configurations

We have described how to round a fractional packing with exactly three configurations computed by solving linear program (1). When the fractional packing has fewer than three configurations we need to group the vertical sections in a different manner. When there are only two configurations, a vertical section s_i is classified as S_{Case1} if $f_{1(i)} + f_{2(i)} \leq 1$ and classified as S_{Case2} if $f_{1(i)} + f_{2(i)} > 1$. Pair the two configurations and re-shape, pack, and round fractional rectangles as explained in Section 4.2. Note that the height increase caused by pairing the two configurations is at most 1.

When there is only one configuration, all fractional rectangles are rounded up for a height increase of the packing of at most 1.

5.7 Differing Number of Rectangle Types in Each Configuration

We have described how to round $S_{Uncommon}$ when each configuration has exactly two rectangle types, or when one configuration has three rectangle types, one configuration has two rectangle types, and one configuration has only one rectangle type. In all of the other remaining possible combinations of the number of rectangle types in each configuration there is at least one configuration with only a single rectangle type, and so the approach used in Sections 5.1-5.6 can be applied for all of these remaining cases.

6 Polynomial Time Implementation

Recall that the input to 2DHMSPP is represented as a list of $3K$ numbers, not a list specifying the dimensions of n rectangles; therefore, any algorithm that specifies individual locations of rectangles in a solution for 2DHMSPP will not run in polynomial time.

We represent a configuration as a list of $O(K)$ numbers: for $1 \leq i \leq K$ we specify the rectangle type T_i, the number of rectangles of type T_i packed side-by-side, and the number of rectangles of type T_i packed on top of each other (note that this last number might not be integer).

Since there are at most K configurations, and we create at most one additional configuration by creating C_{A1} during the rounding process, then at most $O(K^2)$ numbers are needed to specify the packing in $S_{Uncommon}$. Similarly, the packing in S_{Common} is specified using at most $O(K)$ numbers, for a total of $O(K^2)$ numbers to specify the entire packing.

The number of rectangles of type T_i that are packed side-by-side in S_{Common} is equal to the minimum of the number of rectangles of type T_i that are packed in each of C_1, C_2, and C_3. The number of rectangles of type T_i that are packed vertically in S_{Common} is equal to the rounded up sum of the number of rectangles of type T_i that are packed one-on-top of the other in each of C_1, C_2, and C_3. Therefore, finding the number of rectangles of each type that belong in S_{Common} requires $O(K^2)$ operations.

Processing S_{Common} requires $O(K)$ operations as for $1 \leq i \leq K$ our algorithm only needs to round up the fractional values for each rectangle type T_i. Sorting the rectangles in each configuration in $S_{Uncommon}$ by their fractional values requires $O(K^2)$ operations.

Ordering the configurations as specified in Sections 4 and 5 requires $O(K)$ operations, computing the value of the *count* variable requires $O(K)$ operations, and checking which of the cases specified in the lemmas of Sections 4 and 5 are present in the fractional packing requires $O(K)$ operations. Reshaping, packing, and rounding fractional rectangles as described in Section 4.2 requires $O(K^2)$ operations. Finally, packing leftover vertically split fractional rectangles as shown in the figures requires $O(K)$ operations.

Note that the above analysis holds regardless of how many rectangle types are in each configuration of $S_{Uncommon}$.

Theorem 3 *There is a polynomial time algorithm for 2DHMSPP with three rectangle types that computes solutions of value at most $OPT + \frac{3}{2} + \epsilon$ for $\epsilon > 0$.*

Proof As shown in Section 2, an optimal fractional solution to 2DFSPP can be computed in polynomial time. Our algorithm transforms fractional packings obtained by solving linear program (1) into integer packings with height of at most $\frac{3}{2} + \epsilon$ plus the height of the corresponding fractional packing, where ϵ is a positive constant. Finally, as shown above our algorithm can be implemented in polynomial time. □

7 4-Type Algorithm

When $K = 4$ a basic feasible solution for linear program (1) consists of at most four configurations. Our algorithm for this case performs the same four steps as for the case when $K = 3$.

When there are only one or two configurations, the fractional rectangles can be rounded as described in Section 5.6. Note that when $K = 4$ but there are only three configurations in the fractional solution of linear program (1), we cannot use our 3-type algorithm described above, as that algorithm takes advantage of where the at most three case boundaries $B_{i,j}, i \neq j$ are located, but when $K = 4$ there can be up to eight boundaries, and these are not accounted for in the algorithms we described above. Therefore, when $K = 4$ but there are three configurations, we pair the top two configurations as described in Section 4.2 and round up the fractional rectangles in the bottom configuration to produce a packing of height at most 2 plus the value of the solution for linear program (1). In the sequel we only consider the case where the solution of linear program (1) has 4 configurations.

7.1 Grouping Vertical Sections

Recall that within a vertical section s_i, each configuration has a single rectangle type. Let i be the smallest section index for which the sum of the smallest three fractions in section s_i is more than 1, if such a section exists; otherwise, we set $i = 0$. We order the configurations so that $f_{1(i)} \leq f_{2(i)} \leq f_{3(i)} \leq f_{4(i)}$, where $f_{1(i)}$, $f_{2(i)}$, $f_{3(i)}$, and $f_{4(i)}$ represent the fractional values of the fractional rectangles packed in s_i of C_1, C_2, C_3, and C_4, respectively.

We classify the vertical sections $s_i \in S_{Uncommon}$ into 4 cases, depending on the four fractional values $f_{1(i)}$, $f_{2(i)}$, $f_{3(i)}$, and $f_{4(i)}$ as follows:

- S_{Case1} includes all sections s_i such that $f_{1(i)} + f_{2(i)} + f_{3(i)} + f_{4(i)} \leq 1$.
- S_{Case2} includes all sections s_i such that $f_{1(i)} + f_{2(i)} + f_{3(i)} + f_{4(i)} > 1$ and $f_{1(i)} + f_{2(i)} + f_{3(i)} \leq 1$.
- S_{Case3} includes all sections s_i such that $f_{1(i)} + f_{2(i)} + f_{3(i)} > 1$ and $f_{1(i)} + f_{2(i)} \leq 1$.
- S_{Case4} includes all sections s_i such that $f_{1(i)} + f_{2(i)} > 1$.

If no section s_i exists for which the sum of the smallest three fractions is more than 1, then cases S_{Case3} and S_{Case4} will be empty.

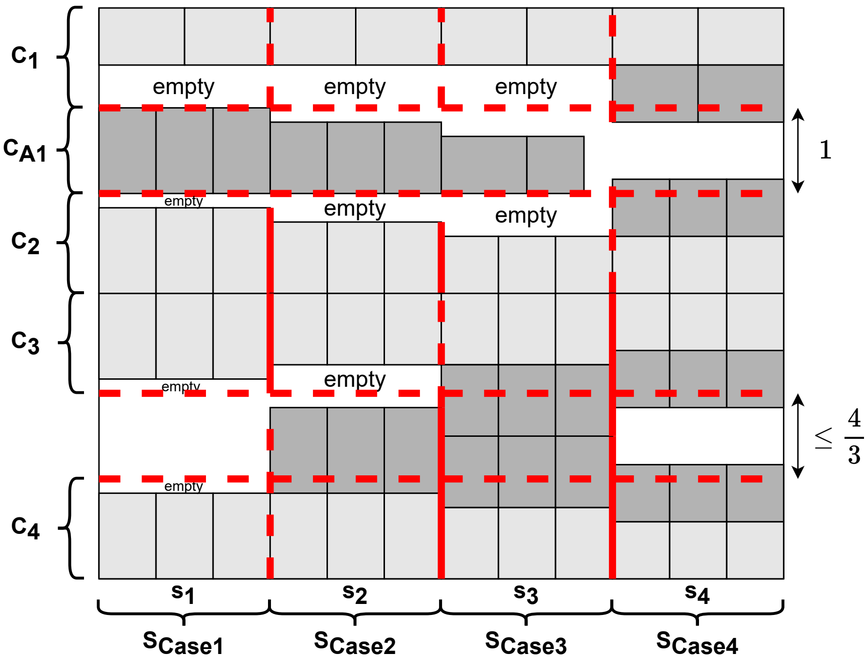


Fig. 10: When $K = 4$ and the fractional packing has exactly four configurations, our algorithm partitions the packing into at most 4 cases.

7.2 Case1: $f_{1(i)} + f_{2(i)} + f_{3(i)} + f_{4(i)} \leq 1$

For every section $s_i \in S_{Case1}$, we remove the fractional rectangles in C_1, C_2, C_3, and C_4 (see Figure 10), including the parts r_{Case1} for vertically split fractional rectangles. We re-shape the fractional rectangles so that they have the full height of a rectangle of the same type but only a fraction of its width, and then we pack them side-by-side in C_{A1}: to create C_{A1} all rectangles in C_1 are shifted upwards, including rectangles in S_{Case2}, S_{Case3}, and S_{Case4}, until there is empty space of height 1 between C_1 and C_2 in section s_i. After shifting the rectangles the tops of the topmost rectangles in C_1 must lie on a common line.

The creation of C_{A1} increases the height of the packing by at most 1.

7.3 Case2: $f_{1(i)} + f_{2(i)} + f_{3(i)} + f_{4(i)} > 1$ and $f_{1(i)} + f_{2(i)} + f_{3(i)} \leq 1$

For every section $s_i \in S_{Case2}$, we remove the fractional rectangles in C_1, C_2, and C_3 (see Figure 10), including the parts r_{Case2} for vertically split fractional rectangles. We re-shape the fractional rectangles so that they have the full height of a rectangle of the same type but only a fraction of its width, and then we pack them side-by-side in C_{A1} as described above. Fractional rectangles in C_4 are rounded up.

The creation of C_{A1} and rounding fractional rectangles in C_4 increases the height of the packing by at most 2.

7.4 Case3: $f_{1(i)} + f_{2(i)} + f_{3(i)} > 1$ and $f_{1(i)} + f_{2(i)} \leq 1$

For every section $s_i \in S_{Case3}$, we remove the fractional rectangles in C_1 and C_2 (see Figure 10), including the parts r_{Case3} for vertically split fractional rectangles. We re-shape the fractional rectangles so that they have the full height of a rectangle of the same type but only a fraction of its width, and then we pack them side-by-side in C_{A1} as described above. Fractional rectangles in C_3 and C_4 are rounded up.

Note that we ordered the configurations based on the fractional values of the fractional rectangles in the leftmost section s_i of S_{Case3}, so $f_{1(i)} + f_{2(i)} + f_{3(i)} > 1$ and $f_{1(i)} \leq f_{2(i)} \leq f_{3(i)} \leq f_{4(i)}$. Hence, $f_{4(i)} \geq f_{3(i)} > \frac{1}{3}$; therefore, rounding up the fractional rectangles in C_3 and C_4 increases the height of the packing by at most $\frac{4}{3}$, and when including the height increase caused by creating C_{A1} the total height increase for this case is at most $\frac{7}{3}$.

7.5 Case4: $f_{1(i)} + f_{2(i)} > 1$

For every section $s_i \in S_{Case4}$ the fractional rectangles in C_1 and C_2 are rounded up, increasing the height of the packing by at most 1 (see Figure 10). Additionally, the fractional rectangles in C_3 and C_4 are rounded up, and by the same reasoning shown for S_{Case3}, the height increase is at most $\frac{4}{3}$. Therefore, the height increase for this case is at most $\frac{7}{3}$.

Theorem 4 *If $K = 4$ there is an algorithm that produces an integer packing of height at most $\frac{7}{3}$ plus the value of the solution for linear program (1).*

8 K-Type Algorithm

Our algorithm for the case when $K > 4$ also performs four steps. The first two steps are the same as the cases when $K = 3$ and $K = 4$. However, we perform one additional pre-processing step: if there are any rectangle types whose widths are greater than half the width of the strip, we place these rectangles leftmost within their configurations. Additionally, we order the configurations so that configurations containing these wide rectangles are placed at the bottom of the packing so that two configurations containing wide rectangles of the same type are put in adjacent positions. Observe that this ensures that wide rectangles are whole (see Figure 11).

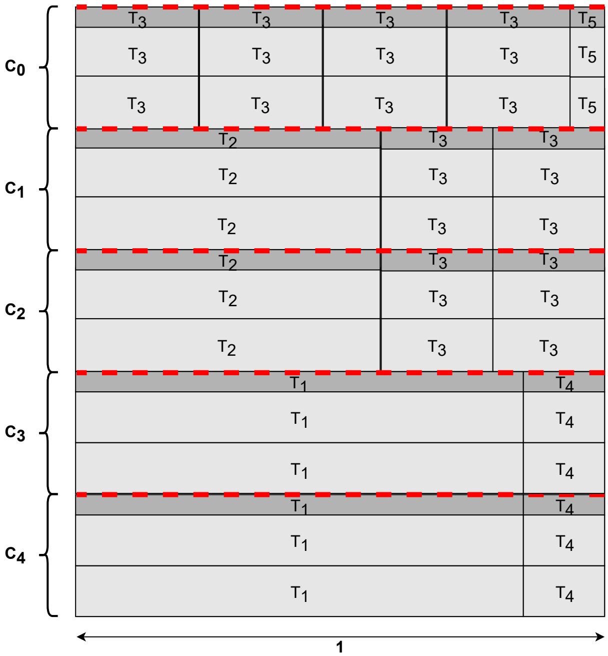


Fig. 11: Configurations with wide rectangle types are adjacent and near the bottom.

8.1 Pairing Configurations

If there are an odd number of configurations, let C_0 be the configuration at the top of the packing, and let each subsequently lower configuration be C_1, C_2, ..., C_{K-1}, respectively. Otherwise, if there are an even number of configurations, let them be C_1, C_2, ..., C_K, respectively, from top to bottom. Pair configurations C_{2i-1} and C_{2i} for $i = 1, 2, ..., \lfloor \frac{K}{2} \rfloor$. Add a region $R_{j,j+1}$ of height 1 between each pair of configurations C_j and C_{j+1}, shifting rectangles upwards as necessary, but ensure that the rectangles whose widths are greater than half the width of the strip still remain at the bottom of the packing (see Figure 12). If there is an odd number of configurations, the final configuration (topmost) will simply have all of its fractional rectangles rounded up.

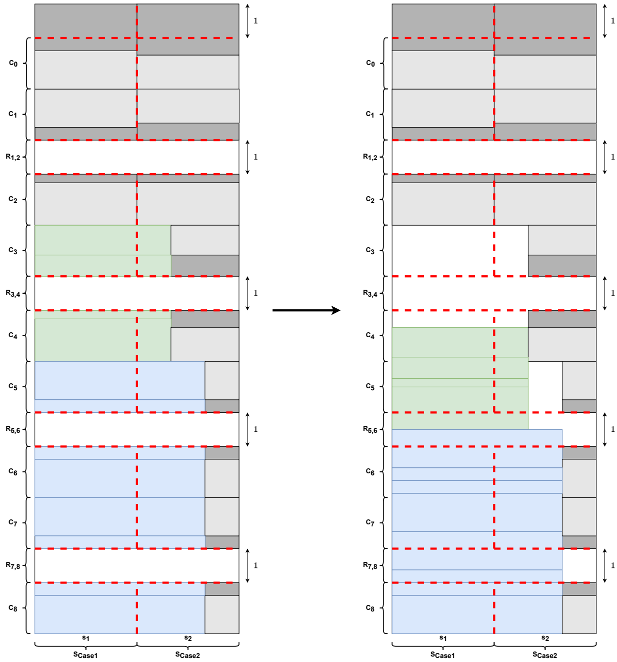


Fig. 12: The K configurations are stacked; if there are an odd number of configurations, the topmost configuration is rounded up instead of paired. After creating the regions $R_{j,j+1}$, the rectangles whose widths are greater than half the width of the strip are again grouped at the bottom of the configuration.

For every paired configurations C_j and C_{j+1}, a vertical section s_i is classified as S_{Case1} if $f_{j(i)} + f_{j+1(i)} \leq 1$ and classified as S_{Case2} if $f_{j(i)} + f_{j+1(i)} > 1$.

Note that since rectangle types whose widths are greater than half the width of the strip were packed together at the bottom of the packing and are already whole, these rectangles are not considered for the remainder of this section.

8.2 Processing Fractional Rectangles

Consider paired configurations C_j and C_{j+1}. For any vertically split fractional rectangle $r \in S_{Case1} \cap S_{Case2}$, put the fractional piece r_{Case2} of r located in S_{Case2} into a set F. Round up the remaining fractional rectangles contained in S_{Case2}.

Add to the set F all the fractional rectangles from each vertical section $s_i \in S_{Case1}$. Using fractional rectangles from F, form as many whole rectangles as possible. Note that all fractional rectangles in S_{Common} and S_{Case2} (excluding the pieces r_{Case2}) are rounded up and therefore represent an integer number of whole rectangles of each type. Since there was an integer number of whole rectangles given as input to 2DHMSPP, then the fractional rectangles in F must yield an integer number of whole rectangles of each type. Therefore, any leftover fractional rectangles in F must have been used to round up other rectangles and can be discarded.

8.3 Packing Rectangles into the Regions $R_{j,j+1}$

Consider one by one the regions $R_{j,j+1}$. Pack the rectangles from F one by one into $R_{j,j+1}$ until the next rectangle r does not fit. Split r and pack in $R_{j,j+1}$ the largest fraction of r that fits; the other piece of r is put back in F. Note that either $R_{j,j+1}$ is completely full (width-wise) or the set F is empty. If F is not empty, continue packing rectangles from F starting with the fractional piece of r, if any, into the remaining regions in the same manner. Note that the rectangles from F must fit within these regions as we did not leave empty space (width-wise) in any region and the total width of the rectangles in F was at most the total width of all the regions combined.

Lemma 11 *After packing the whole rectangles from F into the regions $R_{j,j+1}$ as described above, at most $\lfloor \frac{K}{2} \rfloor - 1$ rectangles were split.*

Proof When rectangles from F are packed into the first region $R_{j,j+1}$, at most one fractional rectangle is leftover (the final rectangle that did not fit in the region). This fractional rectangle combines with the fractional rectangle packed at the beginning of the next region to form a whole rectangle. Combining the fractional rectangle located at the end of a region with the fractional rectangle located at the beginning of the next region accounts for $\lfloor \frac{K}{2} \rfloor - 1$ whole rectangles, as the final region will only have a fractional rectangle at the beginning of the region and not at the end of it. □

8.4 Packing the Split Rectangles

We create additional regions R_1, R_2, ..., $R_{\lfloor \frac{1}{4}K \rfloor}$ at the top of the packing of width equal to the width of the strip to pack the rectangles that were split (see Figure 13). These regions have width 1, the same as the rectangular strip, instead of having just the width of $S_{Uncommon}$. Since the height of S_{Common} is increased by at most 1, then as long as $K > 2$ these regions are located above S_{Common}. Note that S_{Common} could be empty, so that the width of $S_{Uncommon}$ is equal to the width of the full strip.

Lemma 12 *The $\lfloor \frac{K}{2} \rfloor - 1$ split rectangles can be packed using at most $\lfloor \frac{K}{4} \rfloor$ additional regions of height 1.*

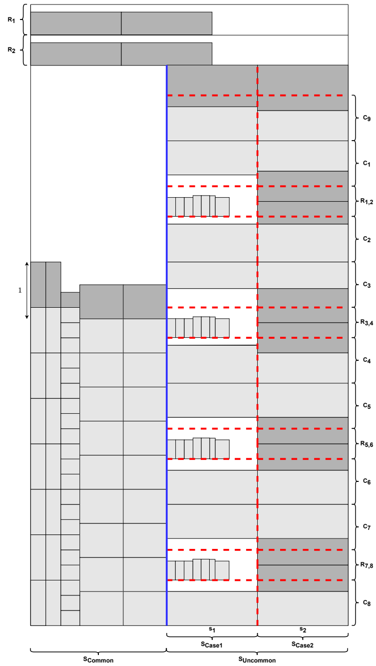


Fig. 13: The configurations have been paired, the fractional rectangles have been processed, and the split rectangles have been made whole and packed in regions placed at the top of the packing.

Proof Note that none of the split rectangles are wide, hence we can pack at least two of these split rectangles into each region. Since there are fewer than $\lfloor \frac{K}{2} \rfloor$ whole rectangles, packing them at least two to a region will use at most $\lfloor \frac{K}{4} \rfloor$ additional regions. □

Note that if K is even, then the height increase of $S_{Uncommon}$ is at most $\lfloor \frac{K}{2} \rfloor$ from the regions created between each pair of configurations and an additional $\lfloor \frac{K}{4} \rfloor$ from the final regions added to the top of the packing. If K is odd, then the height increase of $S_{Uncommon}$ is at most $\lfloor \frac{K}{2} \rfloor + \lfloor \frac{K}{4} \rfloor + 1$, where the term 1 is from rounding up the un-paired configuration.

Theorem 5 *If $K > 3$ then there is an algorithm that produces an integer packing of height at most $OPT + \lfloor \frac{3}{4} K \rfloor + 1$ plus the value of the solution for linear program (1).*

9 Experimental Results

We compared our rectangle packing algorithm for the case when the input contains three types of rectangles with the fractional packings produced by solving linear program (1). We implemented our algorithm for 3 rectangle types using Java. The commercial integer and linear program solver Cplex 12.7, configured using default settings, was used to compute optimal fractional solutions. For each test instance we pre-computed the list of possible *base configurations* to provide to the linear program.

Our algorithm for three rectangle types produces integer packings of height at most $\frac{3}{2}h_{max}+\epsilon$ plus the height of the fractional packing where ϵ is a positive constant, but as we show, its experimental performance is much better than its theoretical upper bound.

9.1 Input Data

We used randomly generated sets of rectangles of 3 types to evaluate our algorithm. Note that the running time of our algorithm depends on K, so the number of rectangles in the input does not have much effect on the running time of the algorithm. For each rectangle type, we randomly generate a width, a height, and a multiplicity, but we performed different tests changing the intervals over which we selected the random values. The width and height of the rectangles were always rounded to two decimal places. For every test case we generated one thousand trials.

The structure of the fractional packing impacts how well our algorithm performs. When all of the heights of the fractional rectangles are nearly the full height of their corresponding rectangle types, our algorithm simply rounds them up and computes near-optimum solutions. In contrast, when some of the heights of the fractional rectangles are much smaller than the heights of their corresponding rectangle types, our algorithm needs to apply a combination of rounding techniques. Therefore, when analyzing the results, we divide the test cases into groups based on the structure of the fractional packing.

9.2 Test Cases

We studied the impact that rectangle type width has on the performance of our algorithm. For $i = 1, 2, ..., 10$, we generated packings where the upper bound on the randomly generated widths was $\frac{1}{i}$. For example, when $i = 5$ the widths of the rectangle types were randomly generated from the interval from 0.01 to 0.20. The running time of our algorithm quickly increases when we decrease the upper bound for the rectangle widths because the need to pre-compute the base configurations, so we limited the maximum value of i to be 10 for the majority of our test caes. We also performed a smaller number of experiments where the widths of the rectangle types were randomly generated from the interval 0.01 to 0.05.

We studied the impact that rectangle type height has on the performance of our algorithm. For each value of i noted above, we chose height intervals of size 0.10, 0.25, 0.50, and 1. The minimum height of a rectangle type was 0.01. We include the following height intervals:

- $0.9 - 0.1j$ to $1 - 0.1j$ for $j = 0, 1, ..., 9$.
- $0.75 - 0.25j$ to $1 - 0.25j$ for $j = 0, 1, 2, 3$.

- $0.50 - 0.50j$ to $1 - 0.50j$ for $j = 0, 1$.
- 0.01 to 1.

We summarize the results below.

9.3 Results

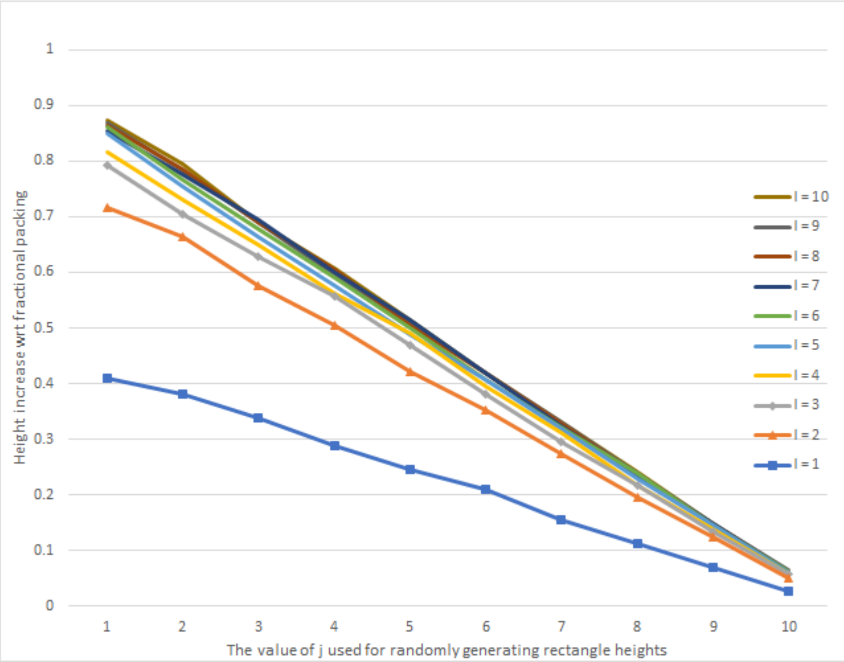


Fig. 14: Average height increase with respect to optimal fractional packing. Each data line represents a different value used for i when selecting the rectangle widths and the x-axis shows the value used for j when selecting the height from the upper bound interval $1 - 0.1j$ and the lower bound $0.9 - 0.1j$. From left to right the chart shows taller to shorter rectangles, and from top to bottom the series of lines show narrower to wider rectangles.

In this section we present only a sample of our experimental results, but the observations that we make in this section will cover all of our experiments [1].

Figure 14 shows how the widths and heights of the rectangle types impact the height of the packing computed by our algorithm. On the x-axis, each label represents the value of j used for that test case, and the height was randomly generated using the interval from $0.9 - 0.1j$ to $1 - 0.1j$. On the y-axis, each label represents the mean of the difference between the height of the

[1]The complete results are available at www.csd.uwo.ca/~ablochha/2DHMSPP_Journal_RawData.pdf

Test Case	Description	Trials	Avg	Min	Max
1	Configurations have 3, 2, and 1 rectangle types	163	0.913	0.48	1.3
2	Configurations have 3, 1, and 1 rectangle types	41	0.834	0.48	1.16
3	Configurations have 2, 2, and 2 rectangle types	389	0.941	0.34	1.38
4	Configurations have 2, 2, and 1 rectangle types	326	0.877	0.19	1.35
5	Configurations have 2, 1, and 1 rectangle types	71	0.828	0.44	1.26
6	Configurations have 1, 1, and 1 rectangle types	10	0.718	0.15	0.98
7	Our Algorithm Total	1000	0.901	0.15	1.38
8	Simple Algorithm Total	1000	2.041	0.65	2.9

Table 1: One thousand trials. Widths are between 0.01 and 0.05, heights are between 0.90 and 1. The results are separated into categories depending on how many different rectangle types appear in each configuration, and within these categories the mean average height increase, minimum height increase, and maximum height increase is listed with respect to the height of an optimal fractional packing. Note that in all 1000 trials the fractional packing contained three configurations.

packing computed by our algorithm and the height of the fractional packing obtained by solving linear program (1); the mean is taken over the thousand tests performed for each value of i. Each individual line on the figure represents the change in value of j for a particular value of i (recall that the width of the rectangles was randomly generated using the interval from 0.01 to $\frac{1}{i}$). So, looking at the chart from left to right shows results of our test cases of taller to shorter rectangles, and looking at the series of lines from top to bottom shows results of our test cases of narrower to wider rectangles.

Rectangle Heights. The height increase in a packing caused by rounding up a fractional rectangle depends on the height of the rectangle type (see Figure 14). Instances generated using shorter rectangle types resulted in our algorithm producing solutions that were closer to the optimal fractional solutions. The correlation between the height of the rectangles and the height increase of the packing was observed in each of our tests cases. Note that if the heights of all the rectangle types are the same, then the fractional values for each fractional rectangle will also be the same (see the full results), which leads to a simpler problem.

Rectangle Widths. Our results do not include the trivial case when all rectangle types have widths larger than $\frac{1}{2}$; however, we did consider cases when some of the rectangle types have widths larger than $\frac{1}{2}$. Within a particular height interval, the instances that contained rectangles wider than $\frac{1}{2}$ have the lowest height increase with respect to the optimal fractional packing. To see this, observe in Figure 14 the data points corresponding to $j = 1$ on the x-axis; the bottommost line represents the results where $i = 1$ and the maximum rectangle width was 1, and each line above the blue line represents a larger and larger value of i. Our results show that for many of the fractional packings that include one or more rectangle types wider than $\frac{1}{2}$ each of the configurations contain a single rectangle type, S_{Case2} and S_{Case3} are both empty, or they have fewer than three configurations (see the full results). Each of these situations are simple to solve and most of their solutions increase the height by less than 1.

As we reduced the maximum width allowed for each rectangle type, the solutions computed by our algorithm had heights that were further away from the height of an optimal fractional packing. Note that in the full results you can see that the fraction of the instances that had three configurations increased when we reduced the maximum width (recall that the theoretical upper bound on the height increase is worse for rounding three configurations). For the instances where the maximum rectangle width was $\frac{1}{10}$, the average height increase of our algorithm nearly reached its maximum for all of our test cases (nearly 0.9 more than the height of an optimal fractional packing).

To get instances that pushed our algorithm towards its theoretical upper bound, we generated inputs that contained rectangle types that are tall and narrow. In Table 1 we show results for a test case where rectangle widths were randomly chosen from the interval 0.01 to 0.05. Under the column labeled "Test Case" we include a number so that we can refer to it easliy, and under the column labeled "Description" we give a description of the data that is included for that test case. For example, test case 1 includes all of the trials (for width between 0.01 and 0.05 and height between 0.90 and 1) where there was a configuration with three different rectangle types, another configuration with two different rectangle types, and a configuration with only a single rectangle type. Test case 7 includes all of the results from the 1000 trials using our algorithm, while test case 8 includes all of the results from the 1000 trials using a simple algorithm that only rounds up fractional rectangles. Under the "Trials" column, we list the number of instances that are included in each test case, and under the "Avg", "Min", and "Max" columns we list the mean average height increase, minimum height increase, and maximum height increase, respectively, within each test case with respect to the height of an optimal fractional packing. Note that in all 1000 instances shown in Table 1 the fractional packing contained three configurations.

The results shown in Table 1 include some of the largest height increases with respect to the fractional packing that we were able to produce in our testing (recall that the running time of our algorithm increases as the rectangles become more narrow). Observe that in test cases 2, 5, and 6, when two of the three configurations have only a single rectangle type each, the problem becomes simpler: the boundaries between the cases are limited to the configuration that has multiple rectangle types, and often one of the configurations with a single rectangle type can be rounded up without increasing the height by a large amount. As seen in the table, 122 of the 1000 trials were this simpler version of the problem and their average height increases (0.834, 0.828, and 0.718 for test cases 2, 5, and 6, respectively) were the lowest within the table.

The instances from Table 1 that have the highest average height increases are in test cases 1 and 3, with height increases of 0.913 and 0.941, respectively. Recall that these test cases are the most complicated versions of the problem and required multiple different algorithms (described in the previous sections) to transform the fractional rectangles into whole ones. When $S_{Uncommon}$ has the maximum number of rectangle types (6) in its configurations ($3+2+1$ or $2+2+2$), the instance is the most difficult to solve. As seen in the table, 552 of the 1000 trials were this more complicated version of the problem, which represents a majority of the instances. We did not perform additional experiments with even narrower rectangles because of the increased running time.

9.4 Final Observation

We compared our 2DHMSPP algorithm for three types against optimal fractional solutions computed by Cplex. Even though our algorithm has a worst case performance of $1.5+\epsilon$ plus the height of an optimal fractional packing, its average performance was significantly better. Our algorithm produces solutions that are closest to the optimal fractional packings on instances where the rectangle types are short and wide and produces solutions that are furthest from the optimal where the rectangles are tall and narrow. Moreover, for instances that have at most two configurations our algorithm performs significantly better than when there are three configurations.

Declarations

Funding

Andrew Bloch-Hansen received funding from the Natural Sciences and Engineering Research Council of Canada for the research that lead to these results (6636-548083-2020).
Roberto Solis-Oba received funding from the Natural Sciences and Engineering Research Council of Canada for the research that lead to these results (1234-567890-0000).

References

[1] Baker, B., Coffman, E., and Rivest, R.: Orthogonal packings in two dimensions. SIAM Journal on Computing **9**(4), 846-855 (1980).

[2] Bloch-Hansen, A.: High multiplicity strip packing. Msc. Thesis, Western University (2019).

[3] Bloch-Hansen, A., Solis-Oba, R., Yu, A.: High multiplicity strip packing with three rectangle types. Paper presented at Combinatorial Optimization: 7th International Symposium, ISCO 2022, Online, 2022.

[4] Coffman, E., Garey, M., Johnson, D., and Tarjan, R.: Performance bounds for level-oriented two-dimensional packing algorithms. SIAM Journal on Computing **9**(4), 808-826 (1980).

[5] De La Vega, W., Lueker, G.: Bin packing can be solved within $1 + \epsilon$ in linear time. Combinatorica **1**(4), 349-355 (1981).

[6] De La Vega, F., and Zissimopoulos, V.: An approximation scheme for strip packing of rectangles with bounded dimensions. Discrete Applied Mathematics **82**(1-3), 93-101 (1998).

[7] Garey, M., and Johnson, D.: Computers and intractability. Vol. 174. San Francisco: freeman (1979).

[8] Harren, R., van Stee, R.: Improved absolute approximation ratios for two-dimensional packing problems. Approximation, Randomization, and Combinatorial Optimization. Algorithms and Techniques, 177-189. Springer, (2009).

[9] Harren, R., Jansen, K, Prädel, L, Van Stee, R.: A $(\frac{5}{3} + \epsilon)$-approximation for strip packing. Computational Geometry **47**(2), 248–267 (2014).

[10] Karloff, H.: Linear programming. Springer Science & Business Media (2008).

[11] Jansen, K., and Solis-Oba, R.: Rectangle packing with one-dimensional resource augmentation. Discrete Optimization **6**(3), 310-323 (2009).

[12] Karmarkar, N., Karp, R.: An efficient approximation scheme for the one-dimensional bin-packing problem. In: 23rd Annual Symposium on Foundations of Computer Science, 312–320. IEEE (1982).

[13] Kenyon, C., and Rémila, E.: A near-optimal solution to a two-dimensional cutting stock problem. Mathematics of Operations Research **25**(4), 645-656 (2000).

[14] Schiermeyer, I.: Reverse-fit: A 2-optimal algorithm for packing rectangles. European Symposium on Algorithms, 290-299. Springer, Berlin, Heidelberg (1994).

[15] Sleator, D.: A 2.5 times optimal algorithm for packing in two dimensions. Information Processing Letters. **10**(1), 37-40 (1980).

[16] Sviridenko, M.: A note on the Kenyon-Remila strip-packing algorithm. Information Processing Letters. **112**(1-2), 10-12 (2012).

[17] Steinberg, A.: A strip-packing algorithm with absolute performance bound 2. SIAM Journal on Computing **26**(2), 401-409 (1997).

[18] Yu, A.: High Multiplicity Strip Packing Problem With Three Rectangle Types. Msc. Thesis, Western University (2019).