Improved Pseudo-Polynomial-Time Approximation for Strip Packing

Waldo Galvez, Salvatore Ingala, Fabrizio Grandoni,
Arindam Khan

IDSIA, USI-SUPSI, Lugano, Switzerland

Strip Packing Problem: (2-D)

Strip Packing Problem: (2-D)

- Input :

Strip Packing Problem: (2-D)

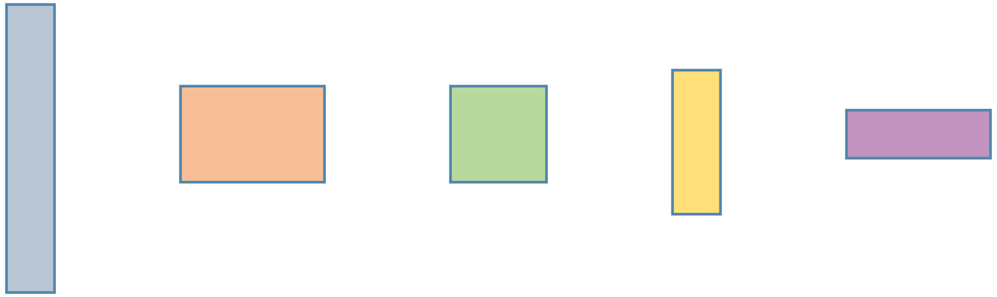
- Input :

  \- Rectangles $R_1, R_2, \ldots, R_n$; Each R_i has integral width and height (w_i, h_i).

Strip Packing Problem: (2-D)

- Input :

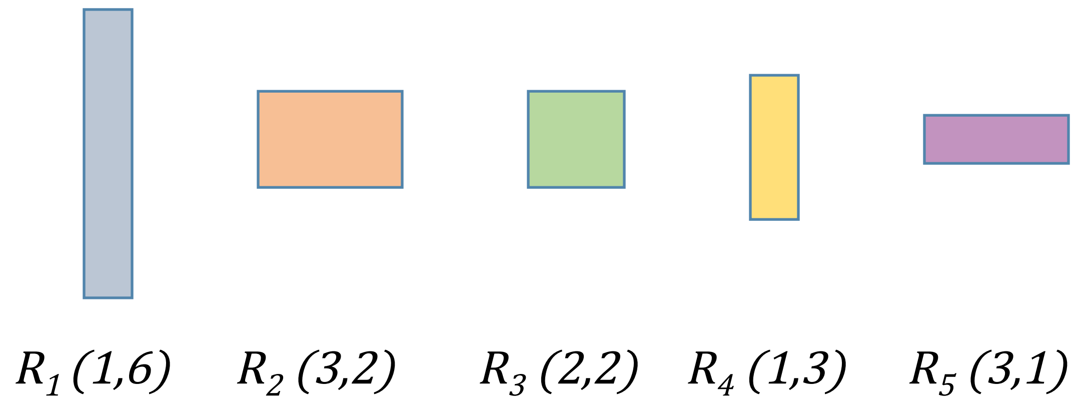
  \- Rectangles $R_1, R_2, \ldots, R_n$; Each R_i has integral width and height (w_i, h_i).



$R_1\ (1,6)$ $R_2\ (3,2)$ $R_3\ (2,2)$ $R_4\ (1,3)$ $R_5\ (3,1)$

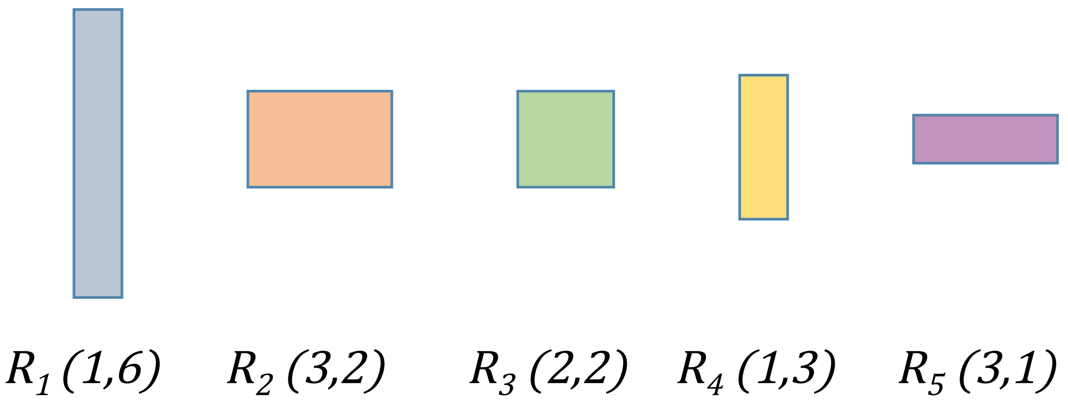
Strip Packing Problem: (2-D)

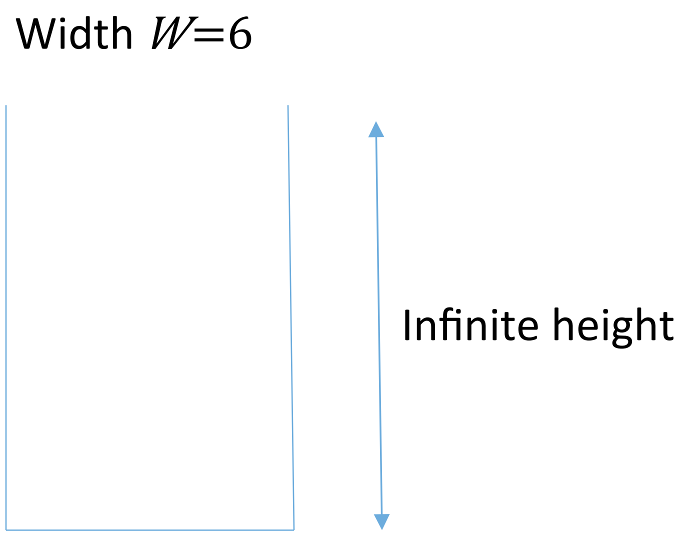
- Input :
  - Rectangles $R_1, R_2, \ldots, R_n$; Each R_i has integral width and height (w_i, h_i).
  - A strip of integral width W and infinite height.



Strip Packing Problem: (2-D)

- Input :
  - Rectangles $R_1, R_2, \ldots, R_n$; Each R_i has integral width and height (w_i, h_i).
  - A strip of integral width W and infinite height.

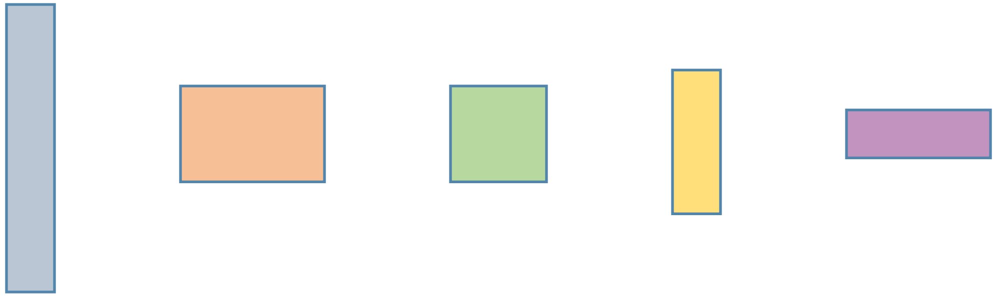




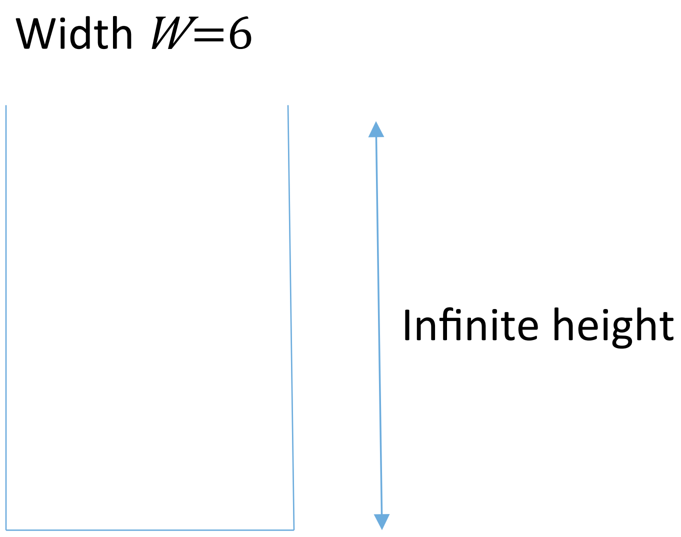
Strip Packing Problem: (2-D)

- Input :
  - Rectangles $R_1, R_2, \ldots, R_n$; Each R_i has integral width and height (w_i, h_i).
  - A strip of integral width W and infinite height.

- Goal :
  - Pack all rectangles minimizing the height of the strip.



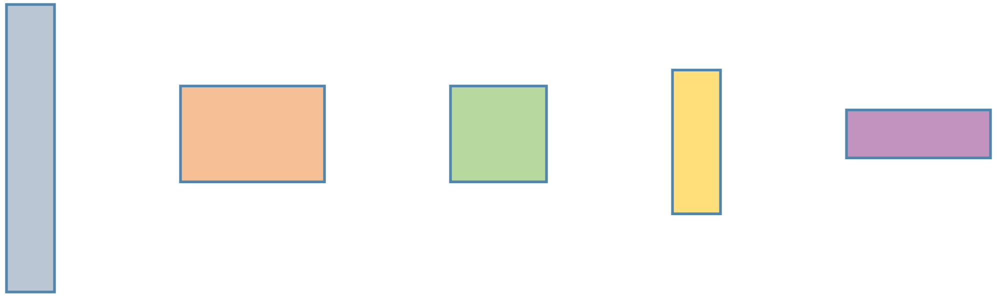
$R_1\,(1,6)$ $R_2\,(3,2)$ $R_3\,(2,2)$ $R_4\,(1,3)$ $R_5\,(3,1)$



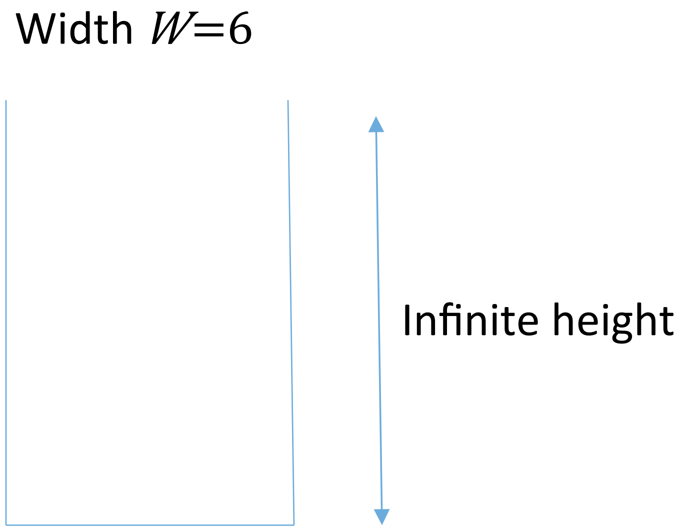
Strip Packing Problem: (2-D)

- Input :
  - Rectangles $R_1, R_2, ..., R_n$; Each R_i has integral width and height (w_i, h_i).
  - A strip of integral width W and infinite height.

- Goal :
  - Pack all rectangles minimizing the height of the strip.
  - Axis-parallel non-overlapping packing.



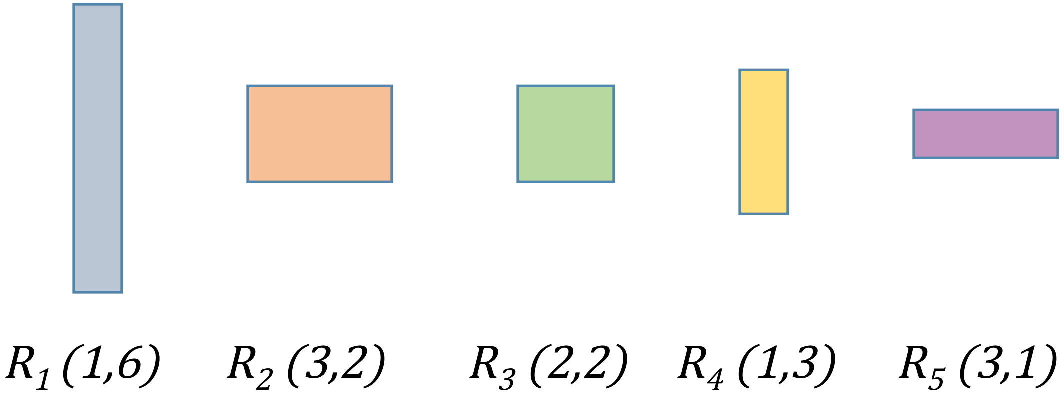
$R_1\ (1,6)$ $R_2\ (3,2)$ $R_3\ (2,2)$ $R_4\ (1,3)$ $R_5\ (3,1)$

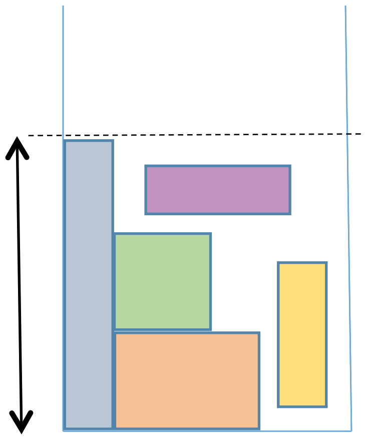


Strip Packing Problem: (2-D)

- Input :
  - Rectangles $R_1, R_2, \ldots, R_n$; Each R_i has integral width and height (w_i, h_i).
  - A strip of integral width W and infinite height.

- Goal :
  - Pack all rectangles minimizing the height of the strip.
  - Axis-parallel non-overlapping packing.

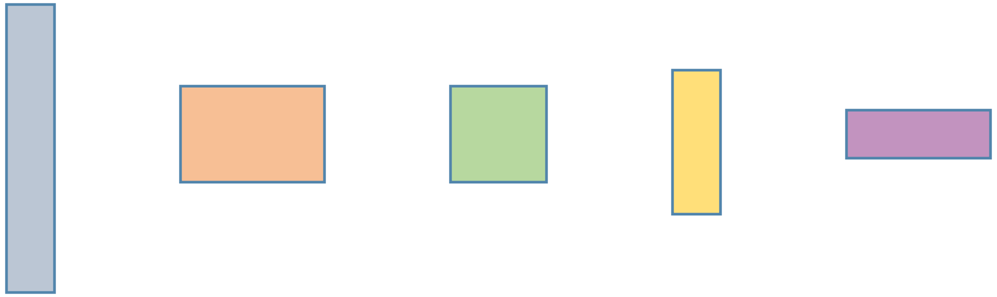




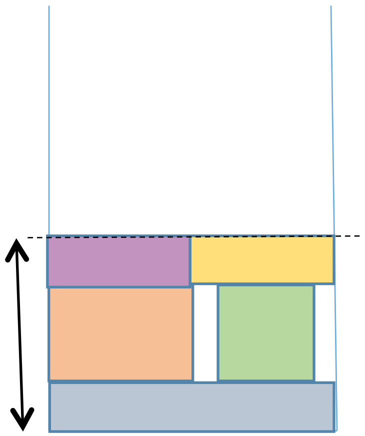
Variant 1:
No rotations are allowed!

Strip Packing Problem: (2-D)

- Input :
  - Rectangles $R_1, R_2, ..., R_n$; Each R_i has integral width and height (w_i, h_i).
  - A strip of integral width W and infinite height.
- Goal :
  - Pack all rectangles minimizing the height of the strip.
  - Axis-parallel non-overlapping packing.



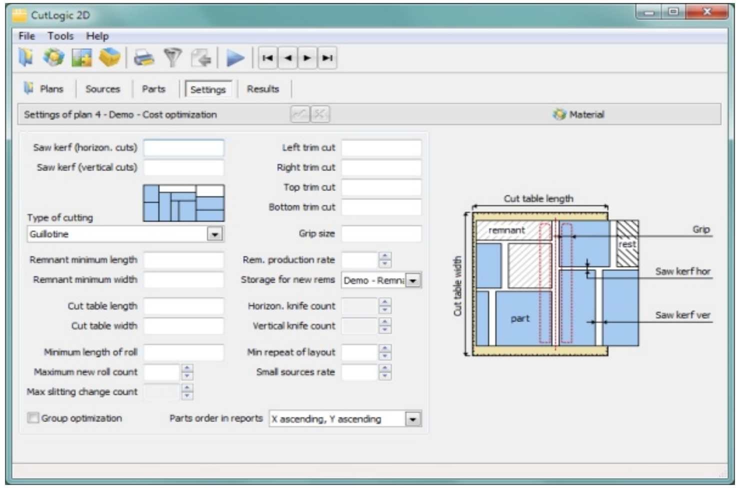
$R_1\,(1,6)$ $R_2\,(3,2)$ $R_3\,(2,2)$ $R_4\,(1,3)$ $R_5\,(3,1)$

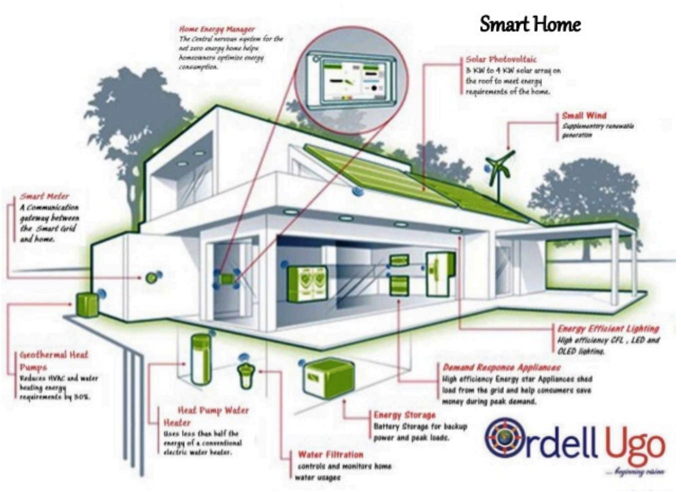


Variant 2:
90° rotations are allowed!

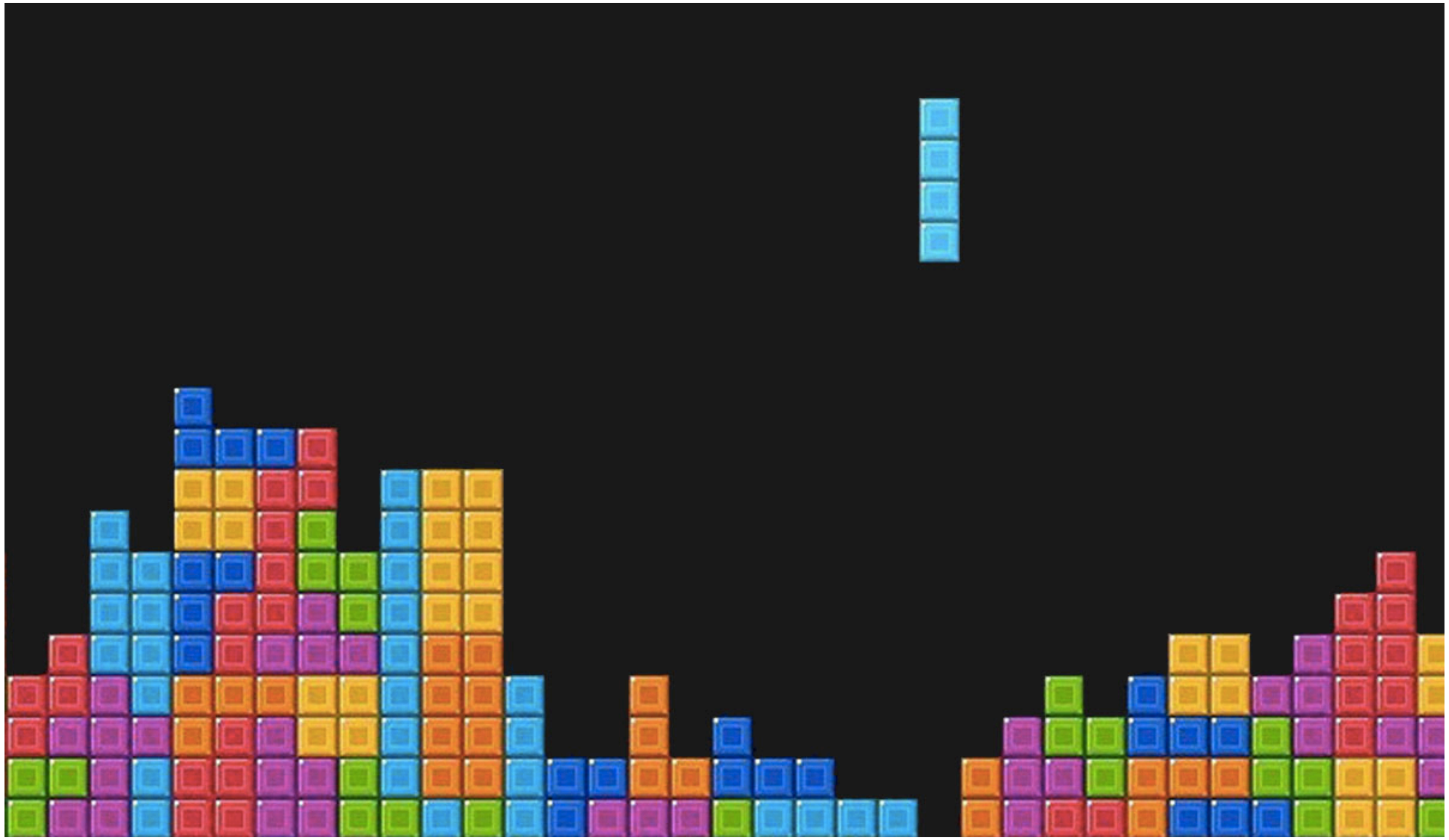
Applications:

- Cutting stock: cloth cutting, steel/wood cutting.
- Logistics and Scheduling:
  memory allocation , truck loading, palletization by robots.
- Recent applications in peak demand reduction in smart-grids.



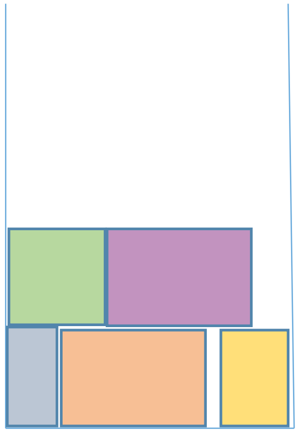



Strip packing is fun!



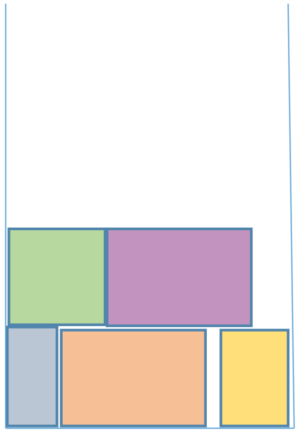
Strip Packing:

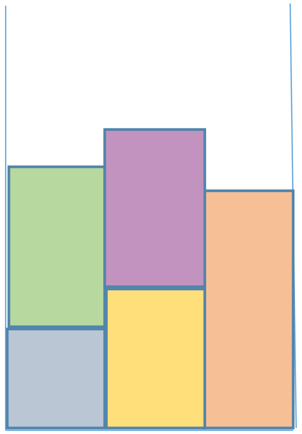
- Strip Packing generalizes
  - bin packing (when all rectangles have same height),



Strip Packing:

- Strip Packing generalizes
  - bin packing (when all rectangles have same height),
  - makespan minimization (when all rectangles have same width).





Related Problems.

- Strip Packing generalizes
  - bin packing (when all rectangles have same height)
  - makespan minimization (when all rectangles have same width)
- Strip Packing is NP-hard.

Hardness.

- Strip Packing generalizes
  - bin packing (when all rectangles have same height)
  - makespan minimization (when all rectangles have same width)
- Strip Packing is NP-hard.
- Reduction from Partition Problem:
  - Can not distinguish in polytime if needs height 2 or 3.
  - Polytime approximation hardness of 3/2 (unless $\mathcal{P}=\mathcal{NP}$).

Hardness.

- Strip Packing generalizes
  - bin packing (when all rectangles have same height)
  - makespan minimization (when all rectangles have same width)
- Strip Packing is NP-hard.
- Reduction from Partition Problem:
  - Can not distinguish in polytime if needs height 2 or 3.
  - Polytime approximation hardness of 3/2 (unless $\mathcal{P}=\mathcal{NP}$).
- Strongly NP-hard: Can not be solved exactly in pseudo-polynomial time (in time $poly(W, h_{max}, n)$ where max rectangle height is h_{max}.
  - No other explicit hardness was known for pseudo-polynomial time.

A tale of approximability.

- Without rotations.
- 2.7-appx. [First-Fit-Decreasing-Height, Coffman-Garey-Johnson-Tarjan '80] …
- 5/3+ε [Harren-Jansen-Pradel-vanStee '14]
- Asymptotic PTAS [Kenyon-Remila '00] – Good when OPT is large!
- Pseudo-polytime (1.4+ε)-appx [Nadiradze-Wiese SODA '16]
- With Rotations.
- Asymptotic PTAS [Jansen-vanStee '05]

Our Results:

- Algorithm:
- (4/3+ε)-approximation algorithm in *poly(W,n)* time.
  - For both the cases without and with 90^{O} rotations.

- A simple *container-based* packing.
- Breaks the barrier of 3/2 for the case with rotations.
- Pushes present techniques to its limits.

Rest of the talk:

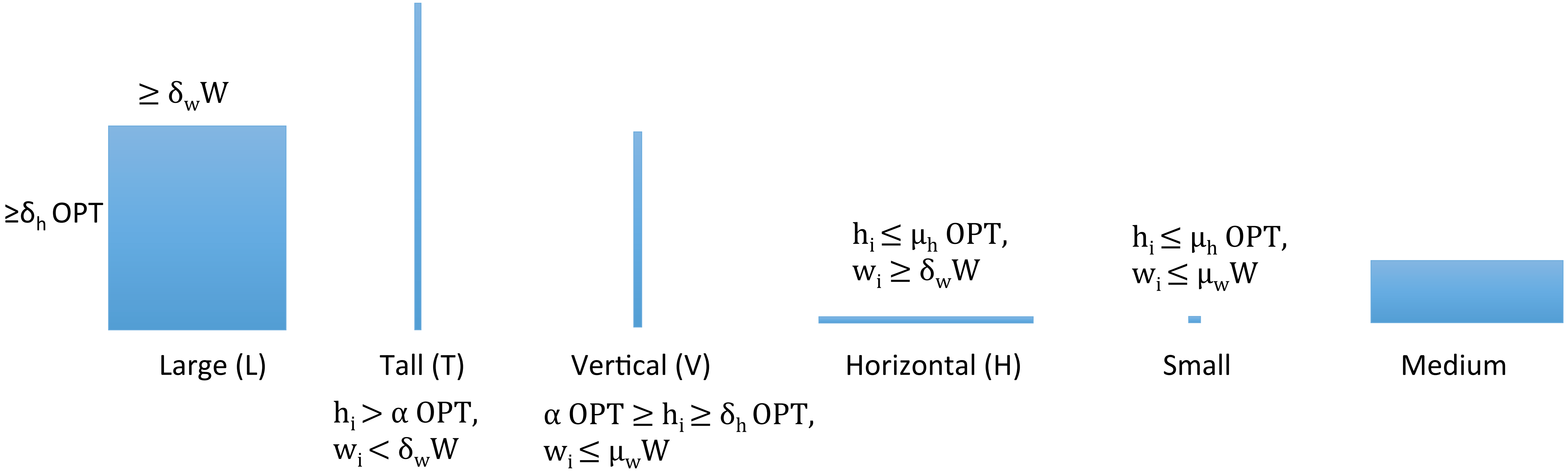
- 1. Existence of a *structured* packing of all rectangles in the strip with height $\leq (4/3+\varepsilon)OPT$.

Rest of the talk:

- 1. Existence of a *structured* packing of all rectangles in the strip with height $\leq(4/3+\varepsilon)OPT$.
- 2. The algorithm finds the best structured packing in time $poly(W, n)$ using a dynamic program.

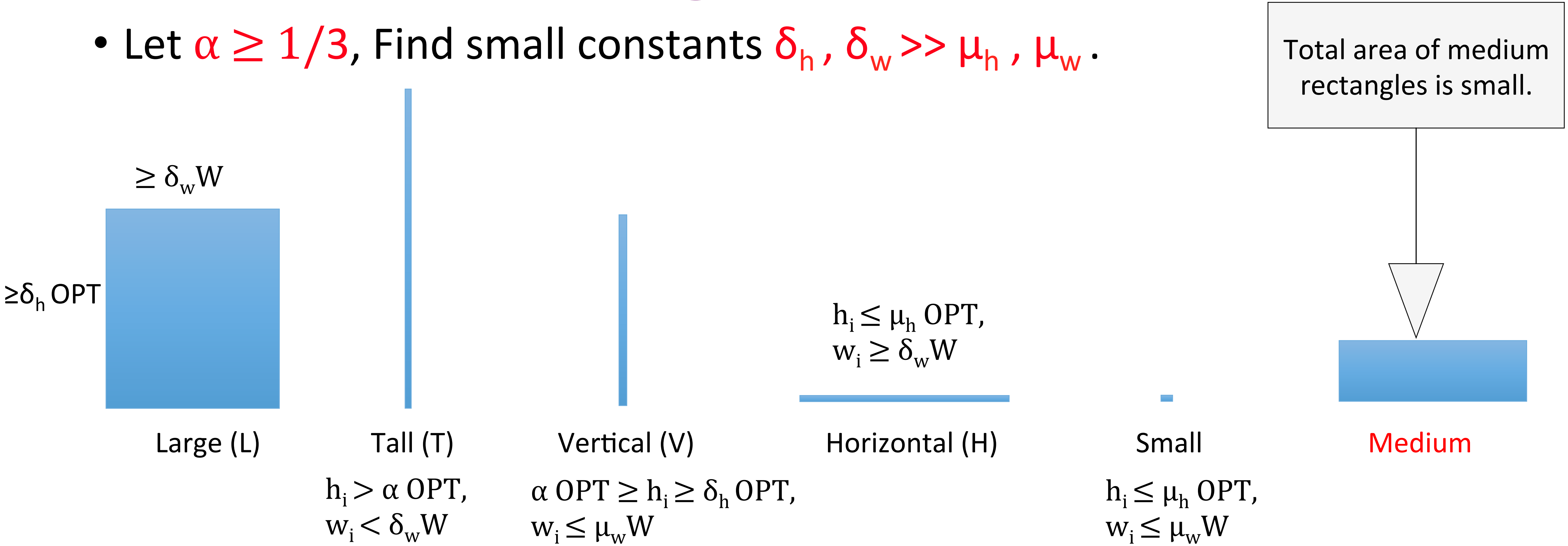
Existence of a structured packing.

- Classification of rectangles.
- Let $\alpha \geq 1/3$, Find small constants δ_h , $\delta_w >> \mu_h$, μ_w .



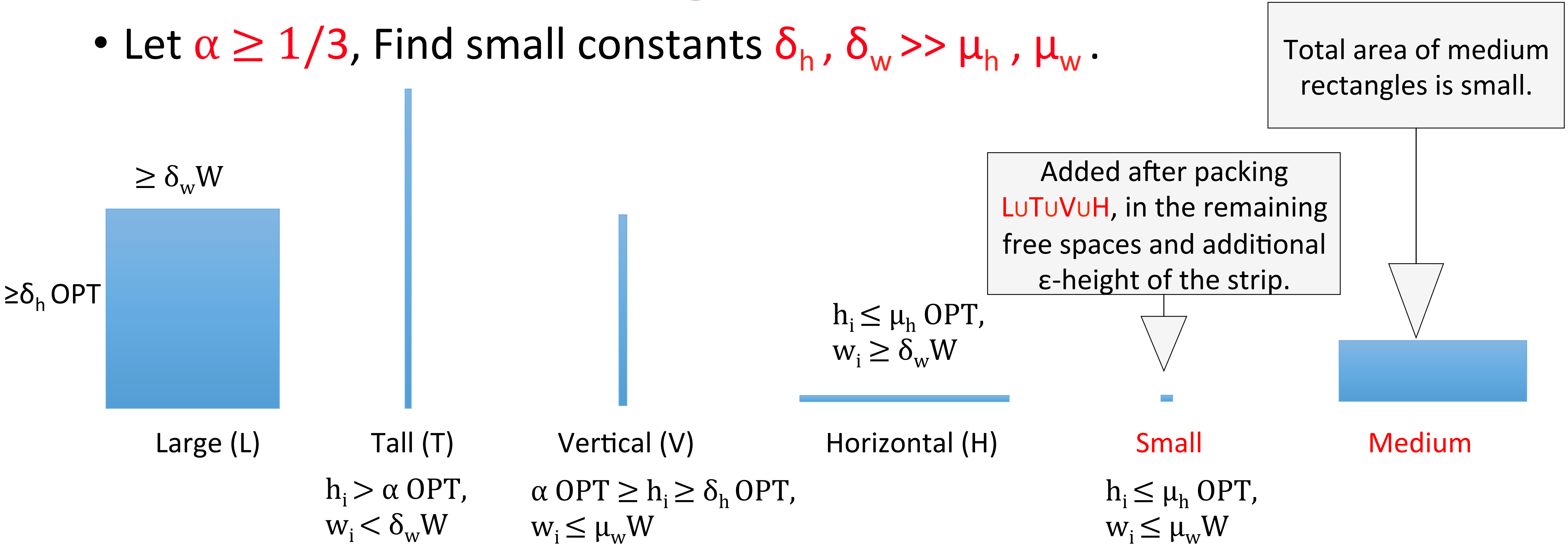
Existence of a structured packing.

- Classification of rectangles.
- Let $\alpha \geq 1/3$, Find small constants δ_h , $\delta_w >> \mu_h$, μ_w .

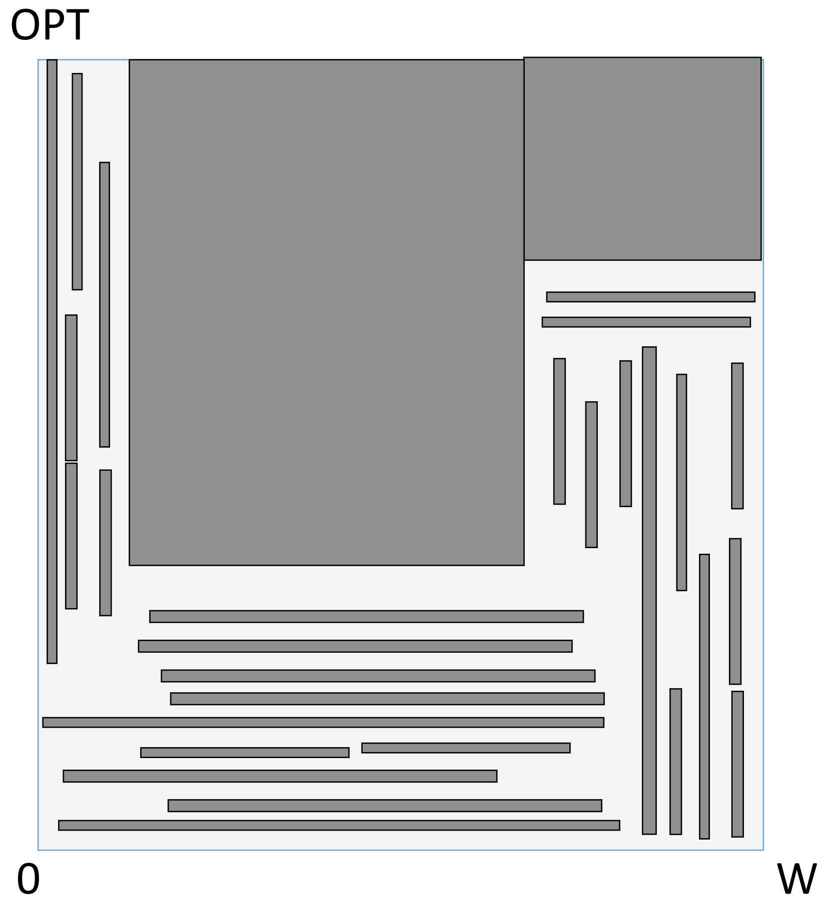


Existence of a structured packing.

- Classification of rectangles.
- Let $\alpha \geq 1/3$, Find small constants δ_h , $\delta_w >> \mu_h$, μ_w .

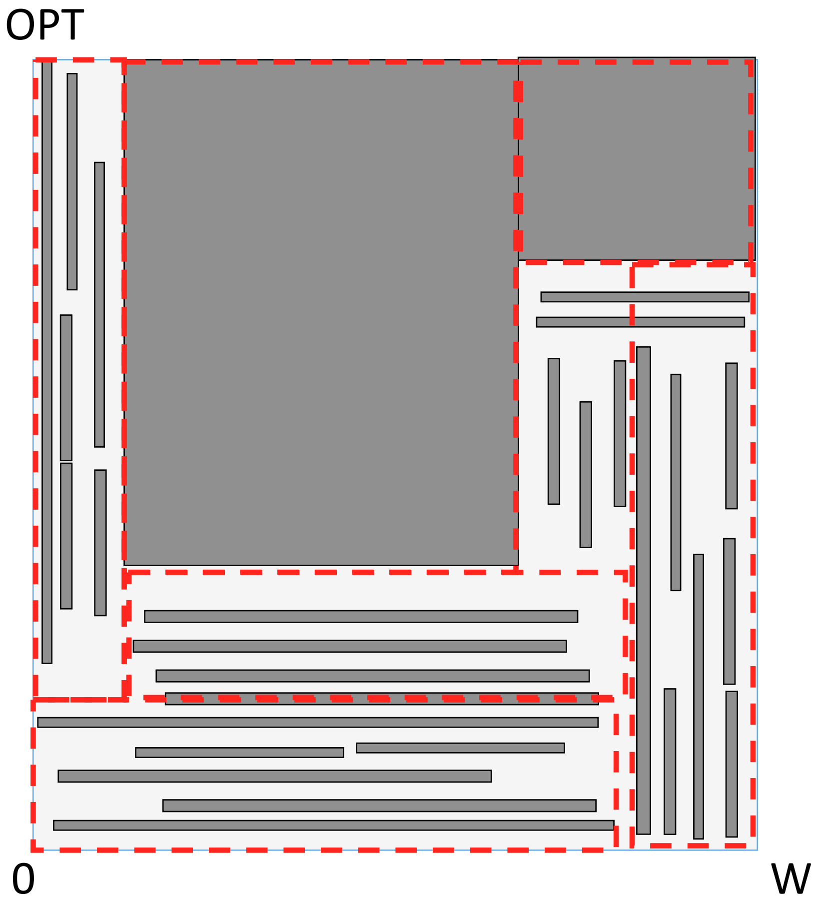


Existence of a structured packing. [Extension of Nadiradze-Wiese]



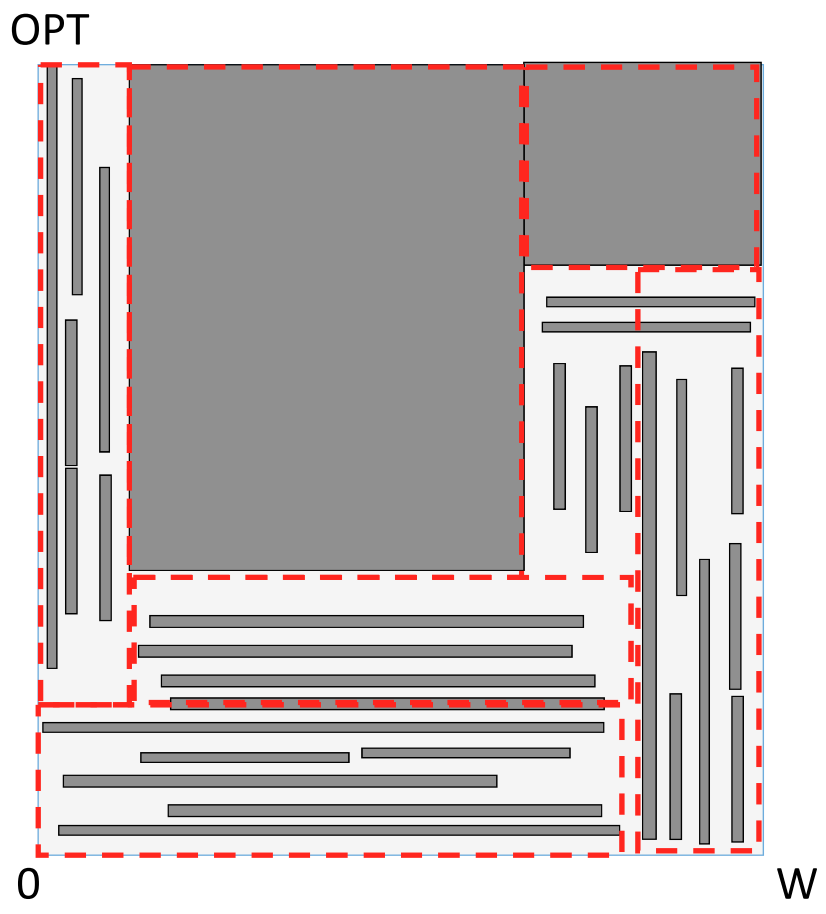
Existence of a structured packing. [Extension of Nadiradze-Wiese]

- There is a partition of [0,W]x[0,OPT] into K=O(1) boxes packing all rectangles in L∪T∪V∪H s.t.



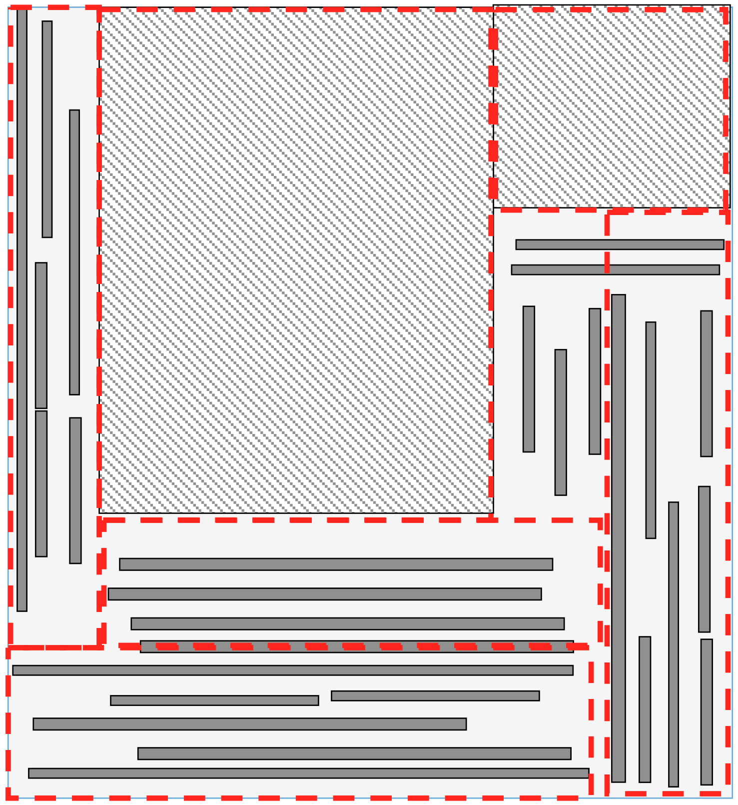
Existence of a structured packing. [Extension of Nadiradze-Wiese]

- There is a partition of [0,W]x[0,OPT] into K=O(1) boxes packing all rectangles in L∪T∪V∪H s.t.
- Each box has size either equal to size of some large rectangle (large box)



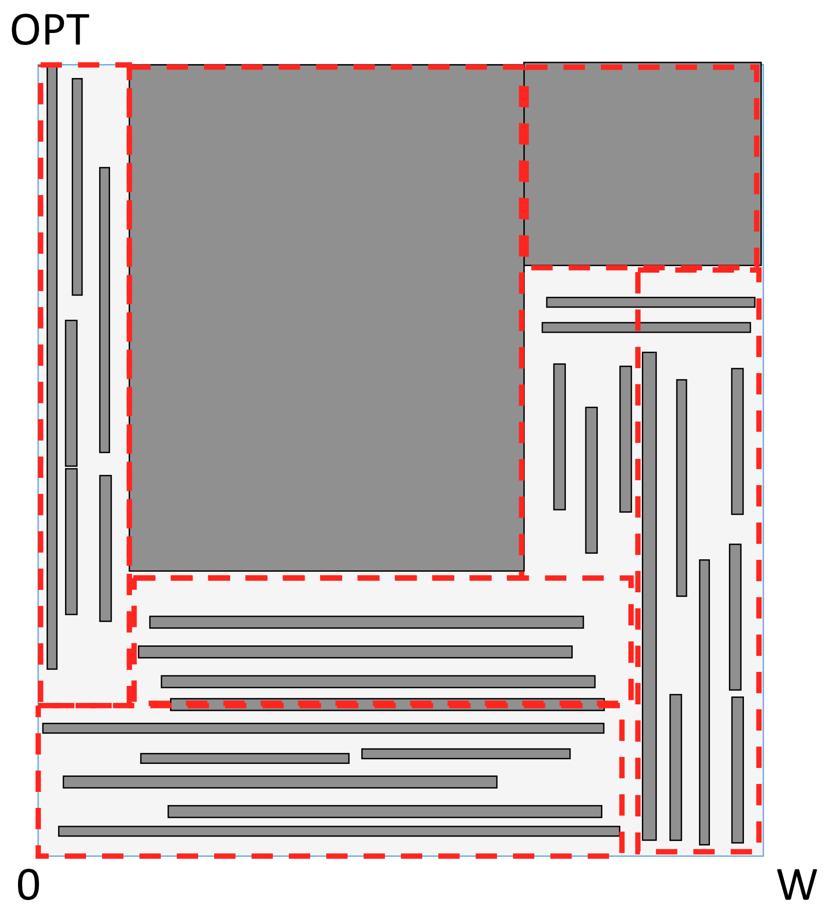
Existence of a structured packing. [Extension of Nadiradze-Wiese]

- There is a partition of [0,W]x[0,OPT] into K=O(1) boxes packing all rectangles in LᴜTᴜVᴜH s.t.
- Each box has size either equal to size of some large rectangle (large box)



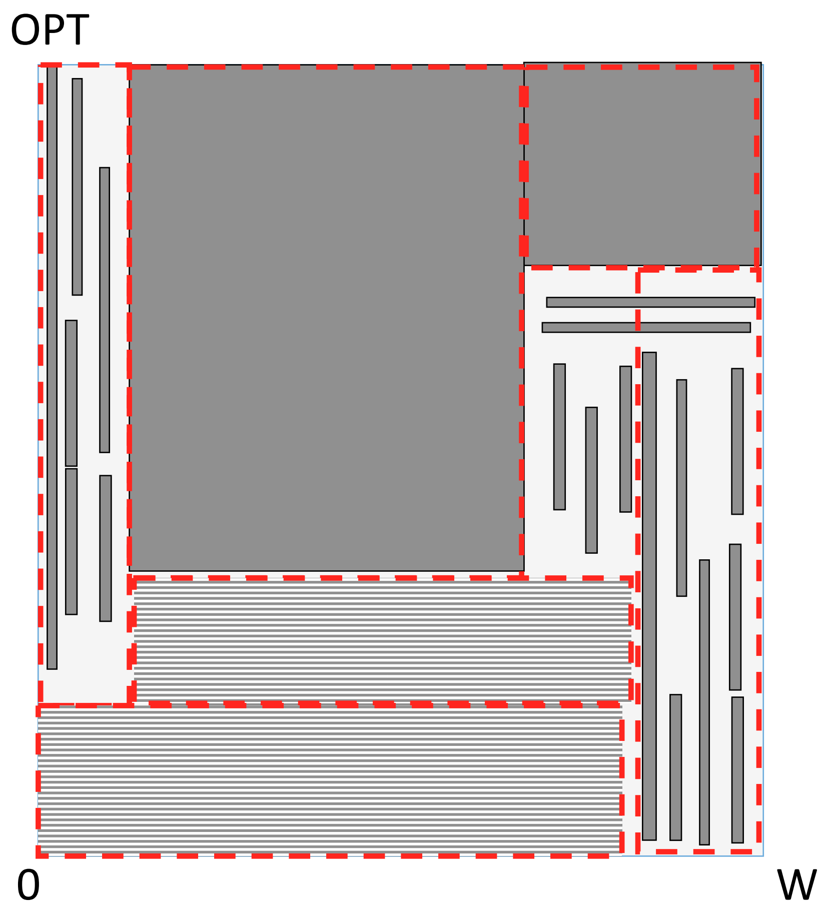
Existence of a structured packing. [Extension of Nadiradze-Wiese]

- There is a partition of [0,W]x[0,OPT] into K=O(1) boxes packing all rectangles in L∪T∪V∪H s.t.
- Each box has size either equal to size of some large rectangle (large box) or height $\leq \delta_h$ OPT (horizontal box)



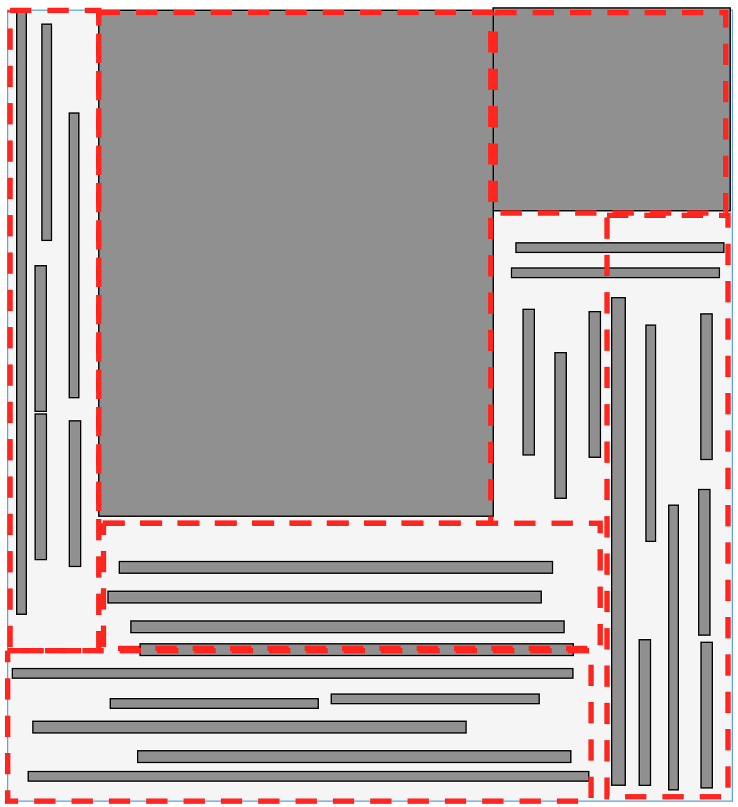
Existence of a structured packing. [Extension of Nadiradze-Wiese]

- There is a partition of [0,W]x[0,OPT] into K=O(1) boxes packing all rectangles in L∪T∪V∪H s.t.
- Each box has size either equal to size of some large rectangle (large box)
  or height $\leq \delta_h$ OPT (horizontal box)



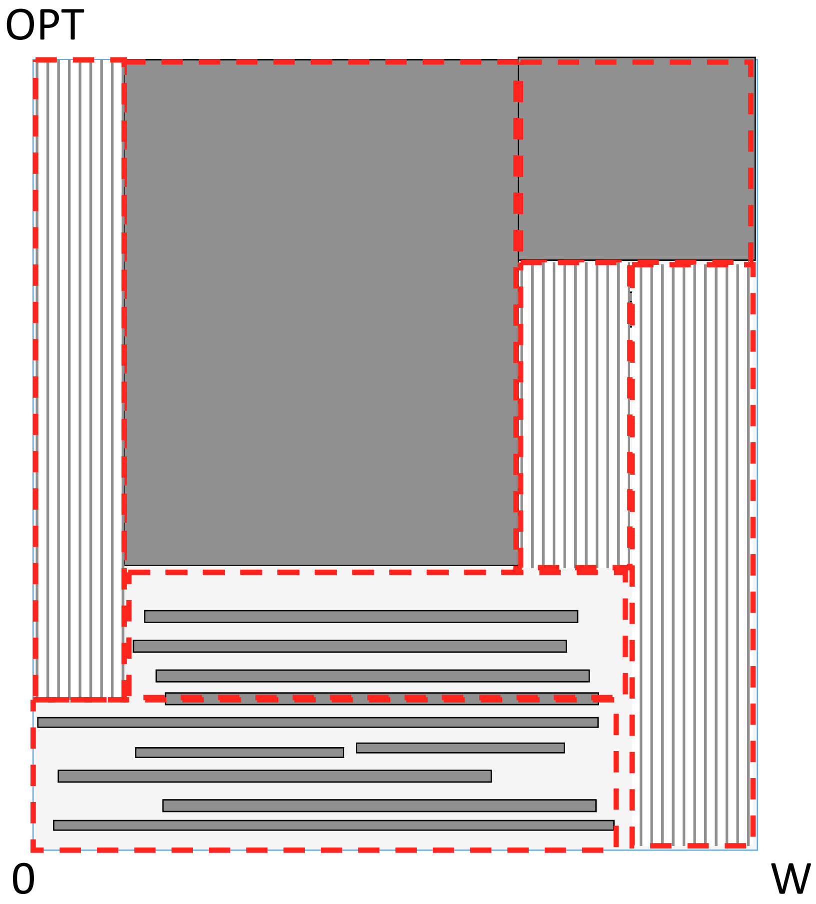
Existence of a structured packing. [Extension of Nadiradze-Wiese]

- There is a partition of [0,W]x[0,OPT] into K=O(1) boxes packing all rectangles in L∪T∪V∪H s.t.
- Each box has size either equal to size of some large rectangle (large box)
  or height $\leq \delta_h$ OPT (horizontal box)
  or width $\leq \delta_w W$ (vertical box).



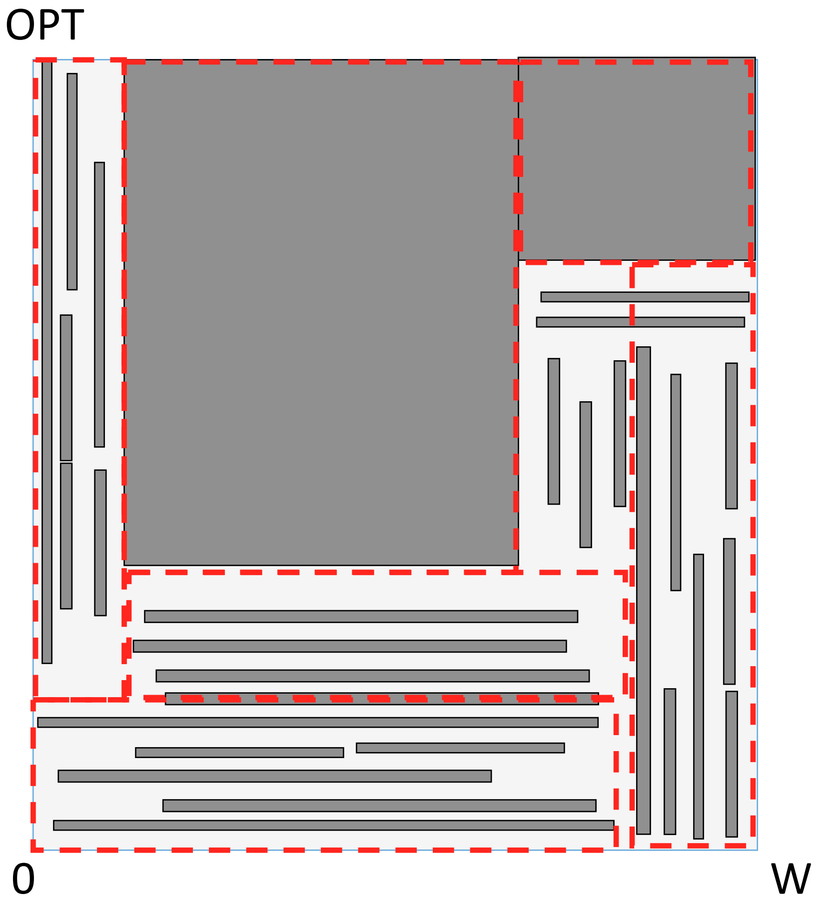
Existence of a structured packing. [Extension of Nadiradze-Wiese]

- There is a partition of [0,W]x[0,OPT] into K=O(1) boxes packing all rectangles in L∪T∪V∪H s.t.
- Each box has size either equal to size of some large rectangle (large box)
  or height $\leq \delta_h$ OPT (horizontal box)
  or width $\leq \delta_w W$ (vertical box).

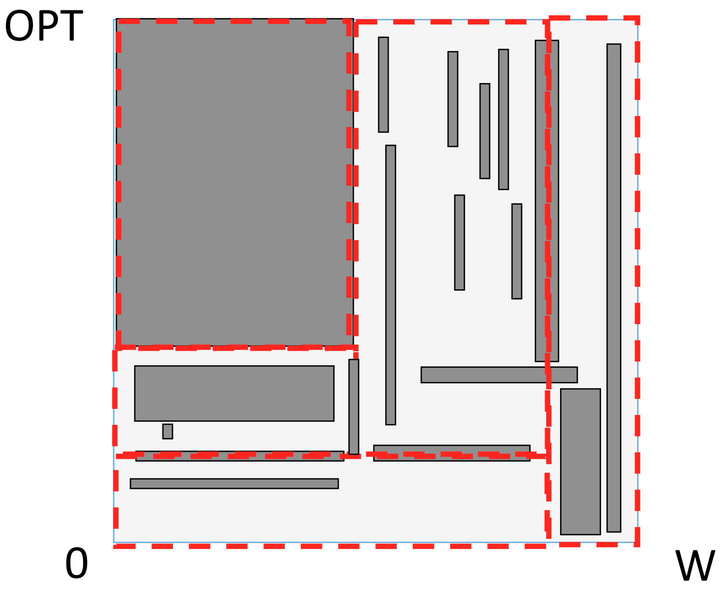


Existence of a structured packing. [Extension of Nadiradze-Wiese]

- There is a partition of [0,W]x[0,OPT] into K=O(1) boxes packing all rectangles in L∪T∪V∪H s.t.
- Each box has size either equal to size of some large rectangle (large box)
  or height $\leq \delta_h$ OPT (horizontal box)
  or width $\leq \delta_w$W (vertical box).
- Each large rectangle is contained in a large box.
- Horizontal rectangles are either contained in a horizontal box or cut by a box. *Area of cut horizontal rectangles* is ≤W.O(ε)OPT
- Tall or vertical rectangles are either contained in a vertical box or vertically cut by a vertical box.

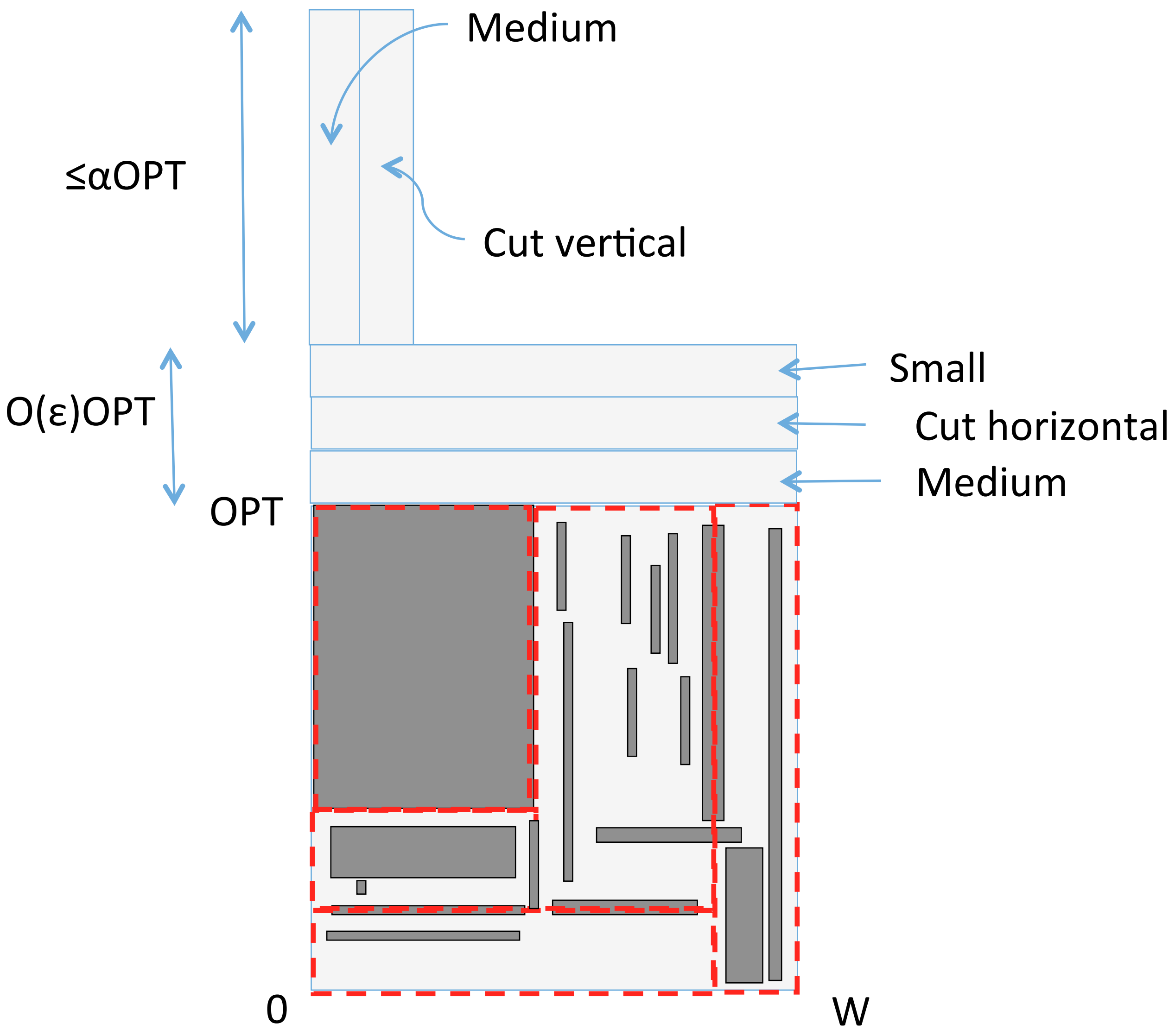


Existence of a structured packing.



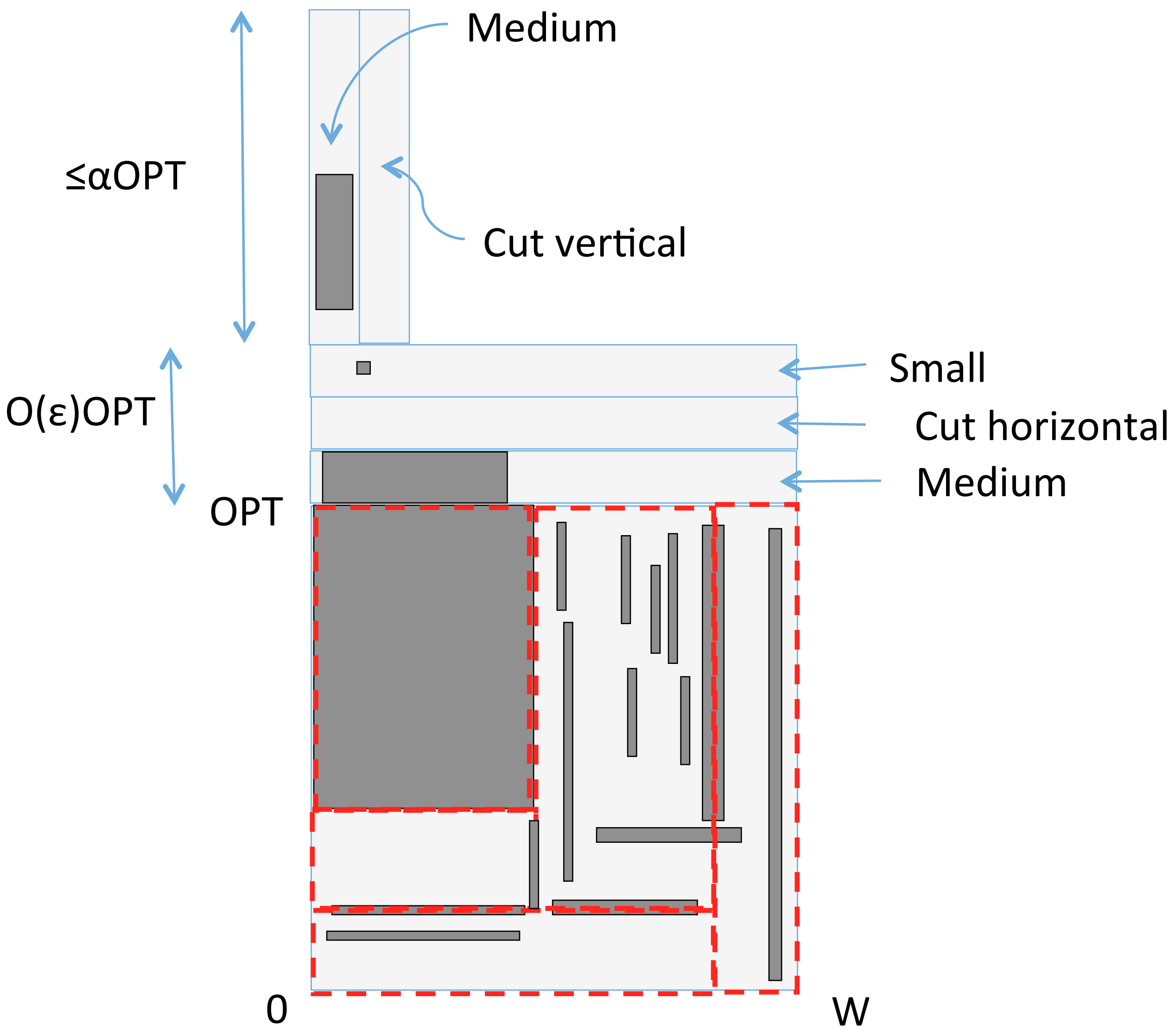
- Problem 1. Rectangles can not be cut.
- Problem 2. How do we find this packing?

Existence of a structured packing.



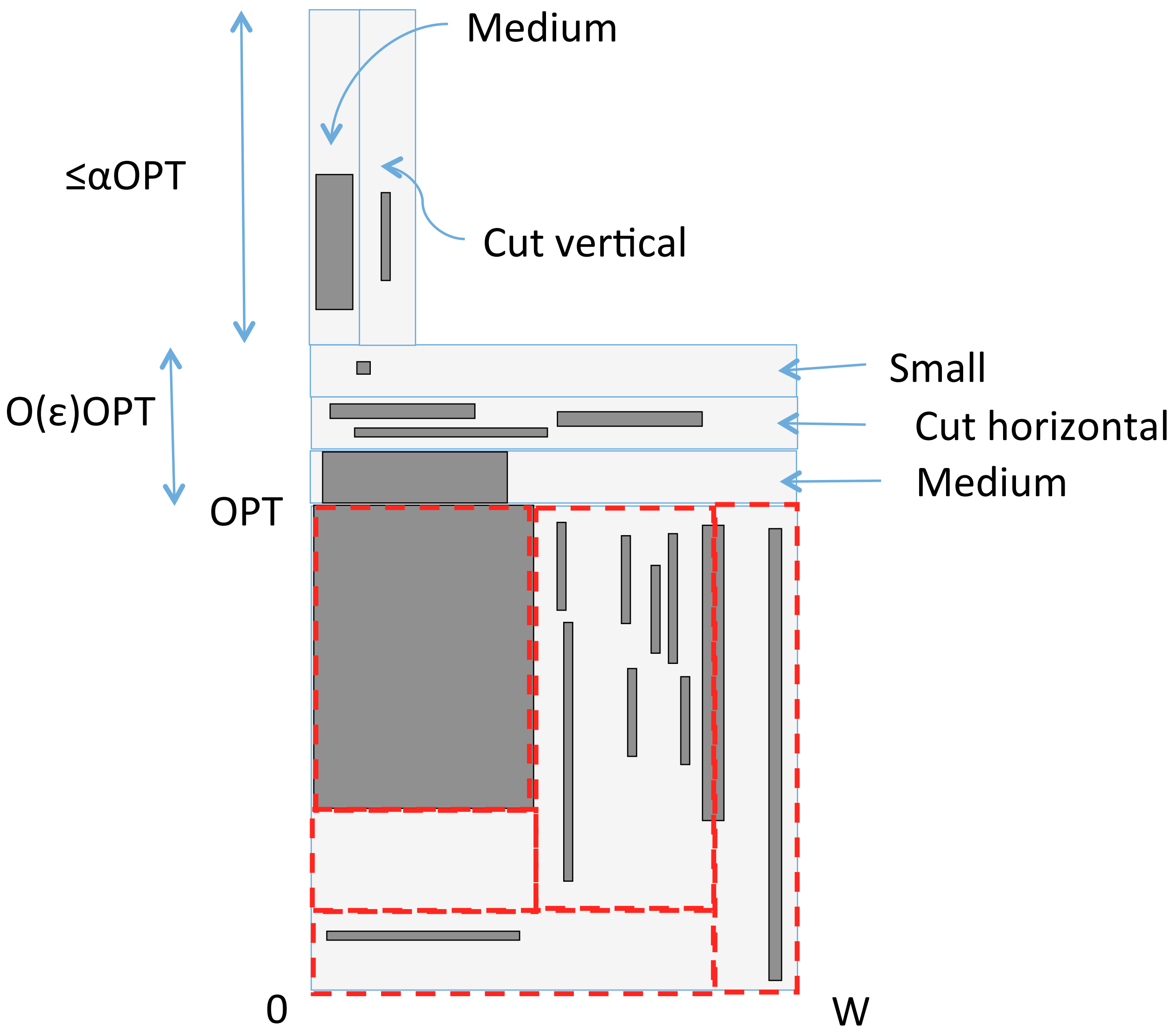
- Problem 1. Rectangles can not be cut.
- Problem 2. How do we find this packing?

Existence of a structured packing.



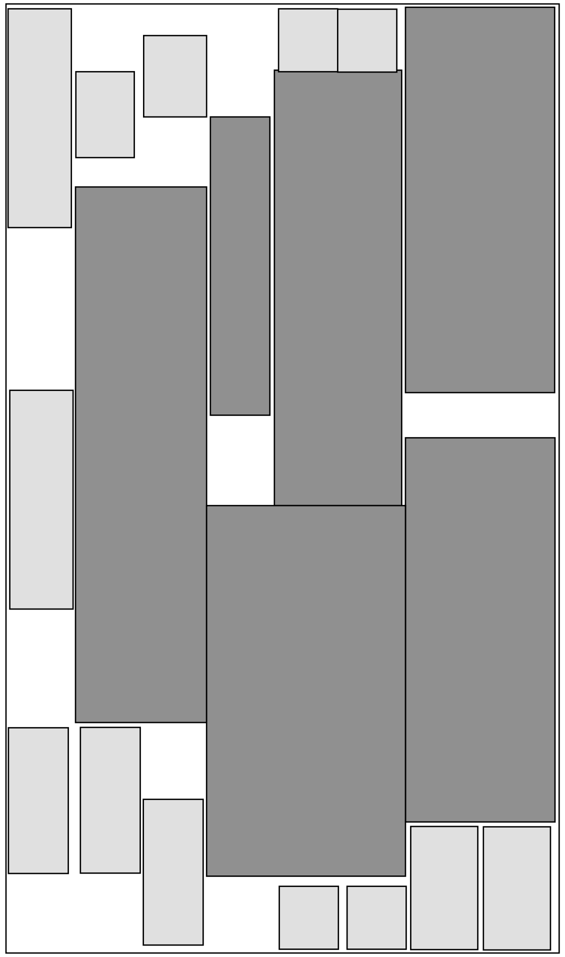
- Problem 1. Rectangles can not be cut.
- Problem 2. How do we find this packing?

Existence of a structured packing.



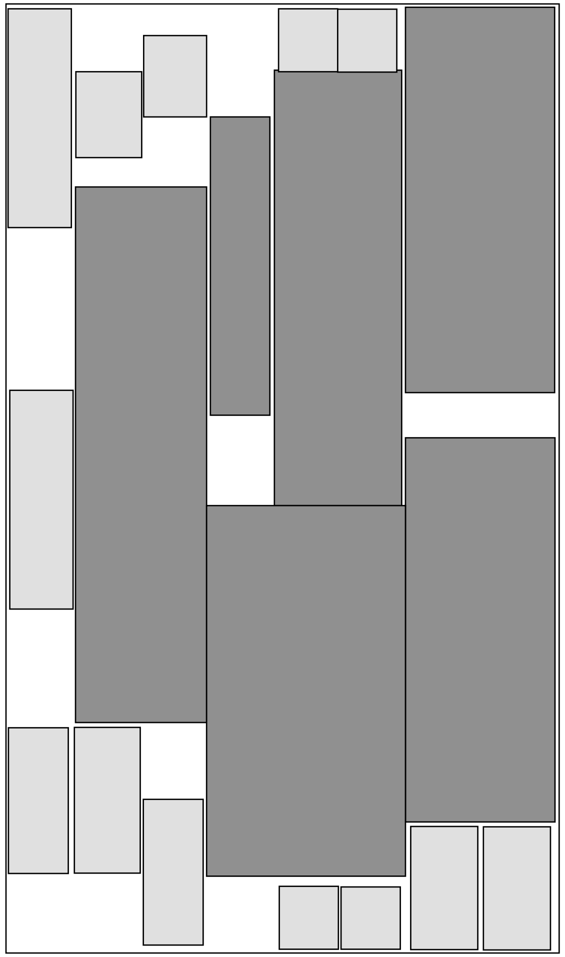
- Now only tall rectangles can be cut.
- One can find packing of horizontal boxes using an LP.
- But still not clear how to find packing of the vertical boxes.

Rearrangement of vertical box

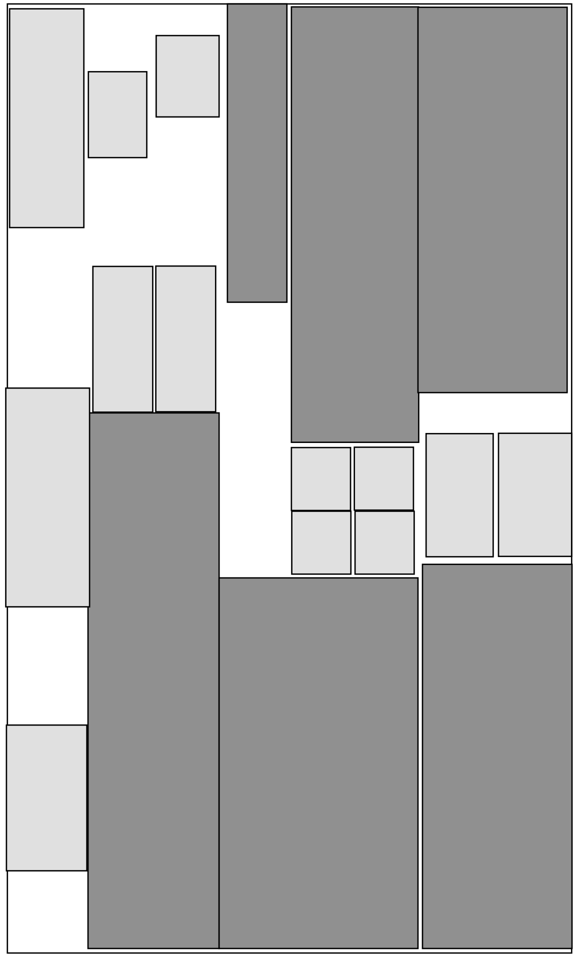


- For simplicity, assume
  - all vertical rectangles have unit width,
  - no tall rectangles are cut,
  - each height is integral multiple of $\gamma\mathrm{OPT}$.
- Tall = dark gray, Vertical = light gray.
- Any vertical line intersects at most two tall (>1/3 OPT) rectangles.
- For each tall rectangle, either top or bottom cannot contain any tall rectangle.
- Shift tall rectangles so that they touch boundary.

Rearrangement of vertical box

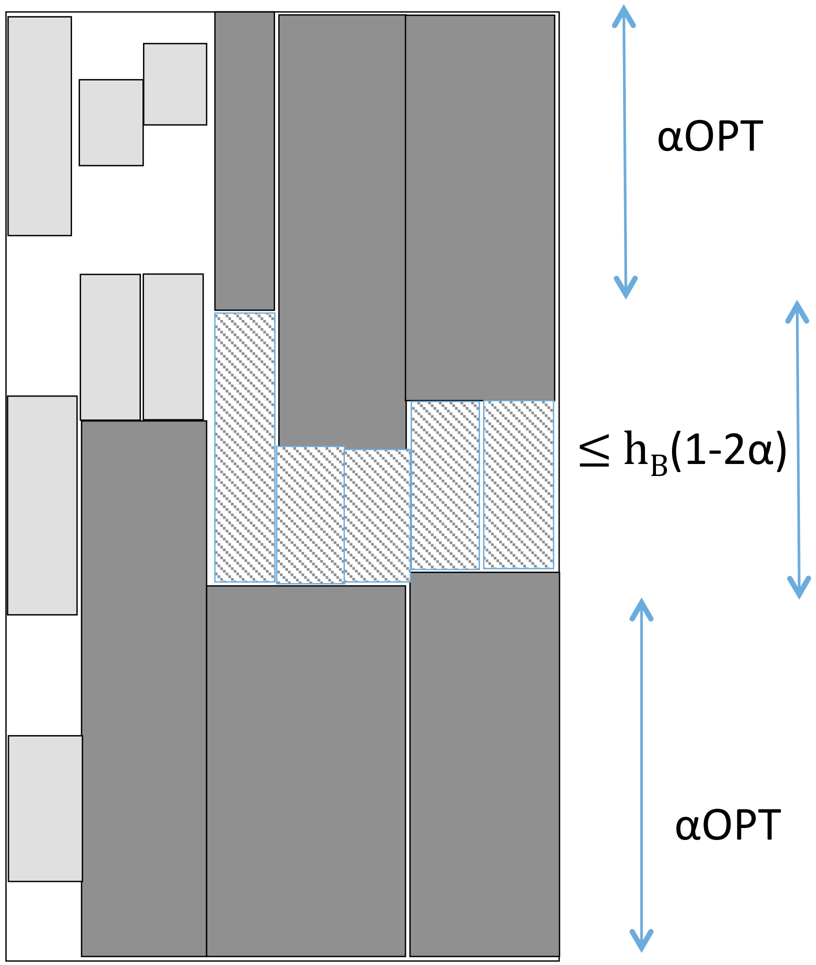


Rearrangement of vertical box



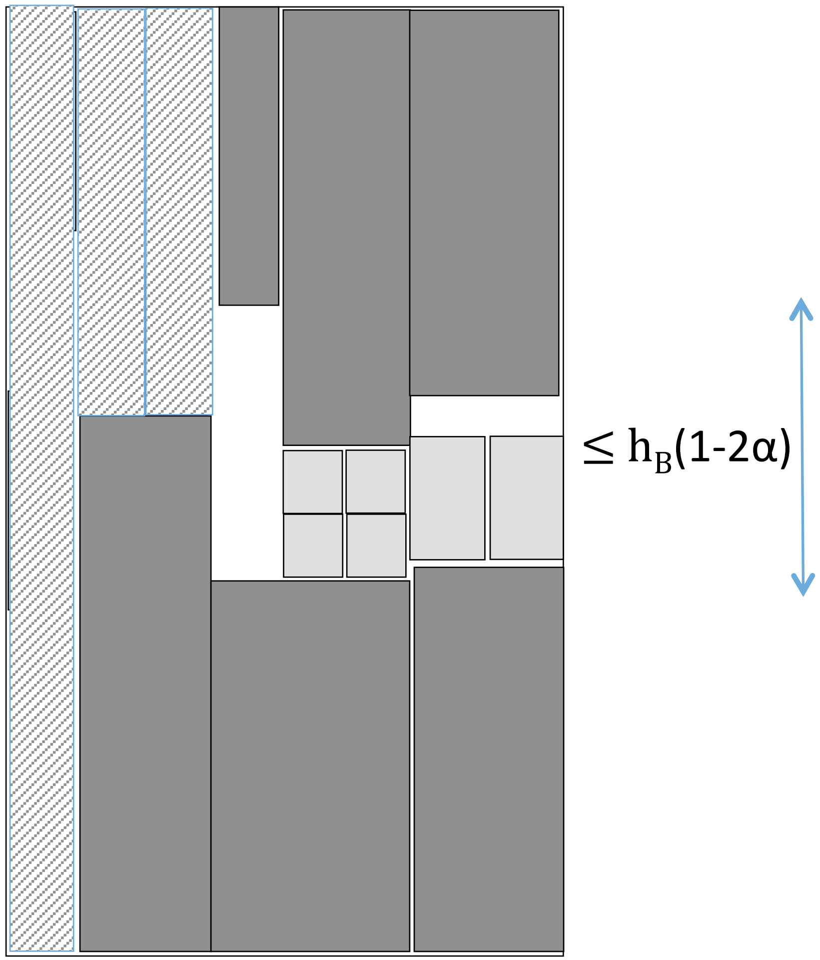
- Box $B:=(w_B, h_B)$, T=Tall rectangles.
- Consider each unit width stripes in B-T.
- Free rectangle: If both the top and bottom sides of the stripe overlaps with T.

Rearrangement of vertical box



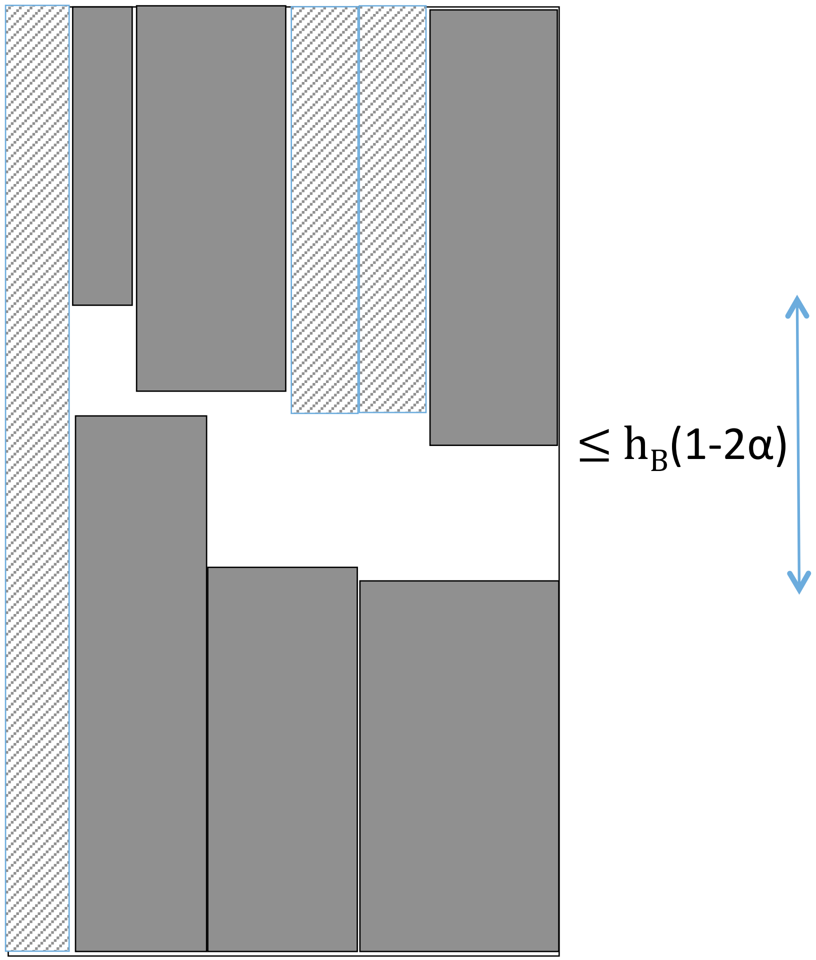
- Box $B{:=}(w_B, h_B)$, T=Tall rectangles.
- Consider each unit width stripes in B-T.
- Free rectangle: If both the top and bottom sides of the stripe overlaps with T.
- Each free rectangle is contained in a strip of width w_B and height at most $h_B\text{-}2\alpha\mathrm{OPT} \leq h_B(1\text{-}2\alpha)$.

Rearrangement of vertical box



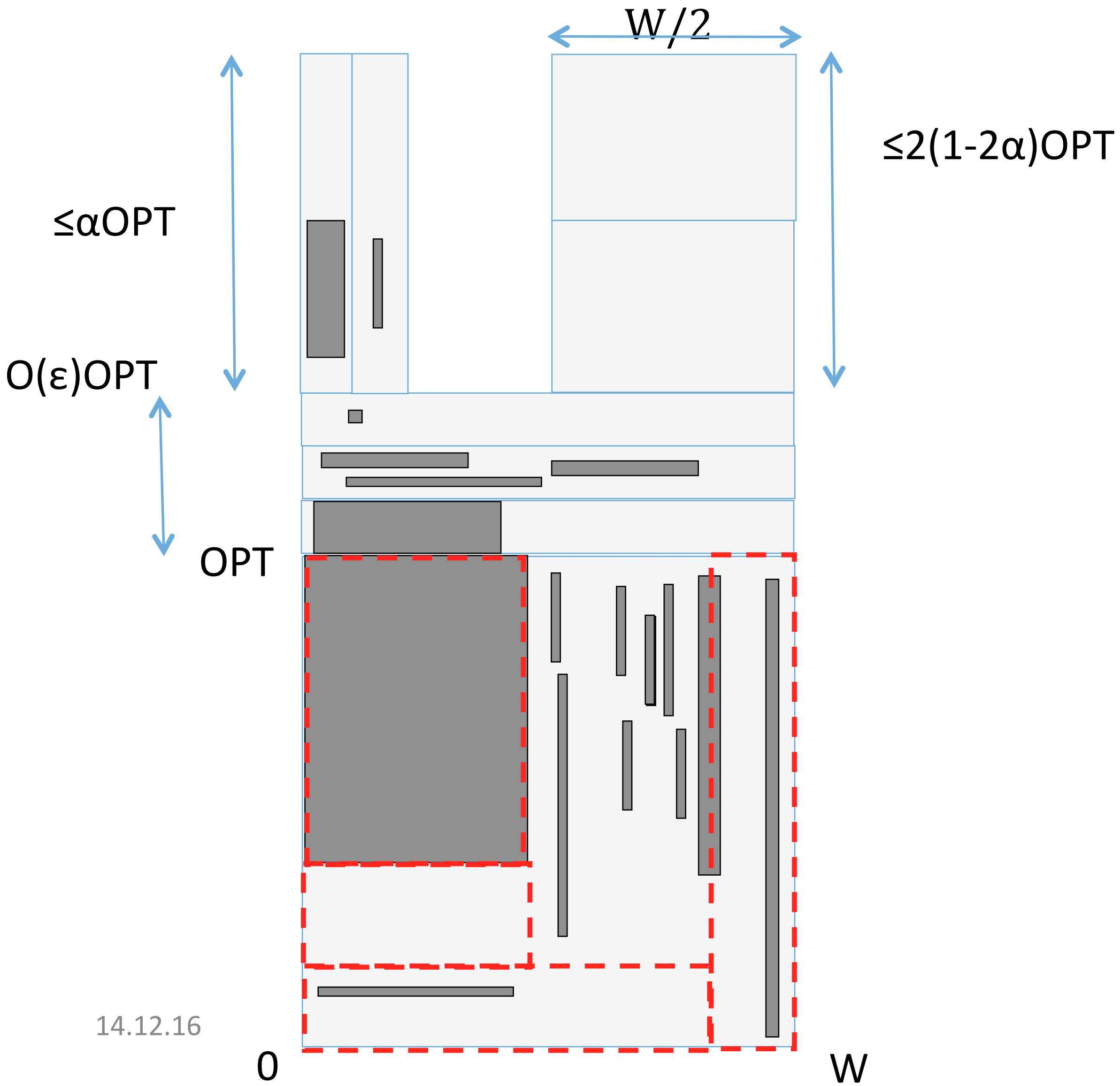
- Box $B:=(w_B, h_B)$, T=Tall rectangles.
- Consider each unit width stripes in $B\text{-}T$.
- Free rectangle: If both the top and bottom sides of the stripe overlaps with T.
- Pseudo rectangle: If at most one of the top and bottom sides of the stripe overlaps with T.

Rearrangement of vertical box

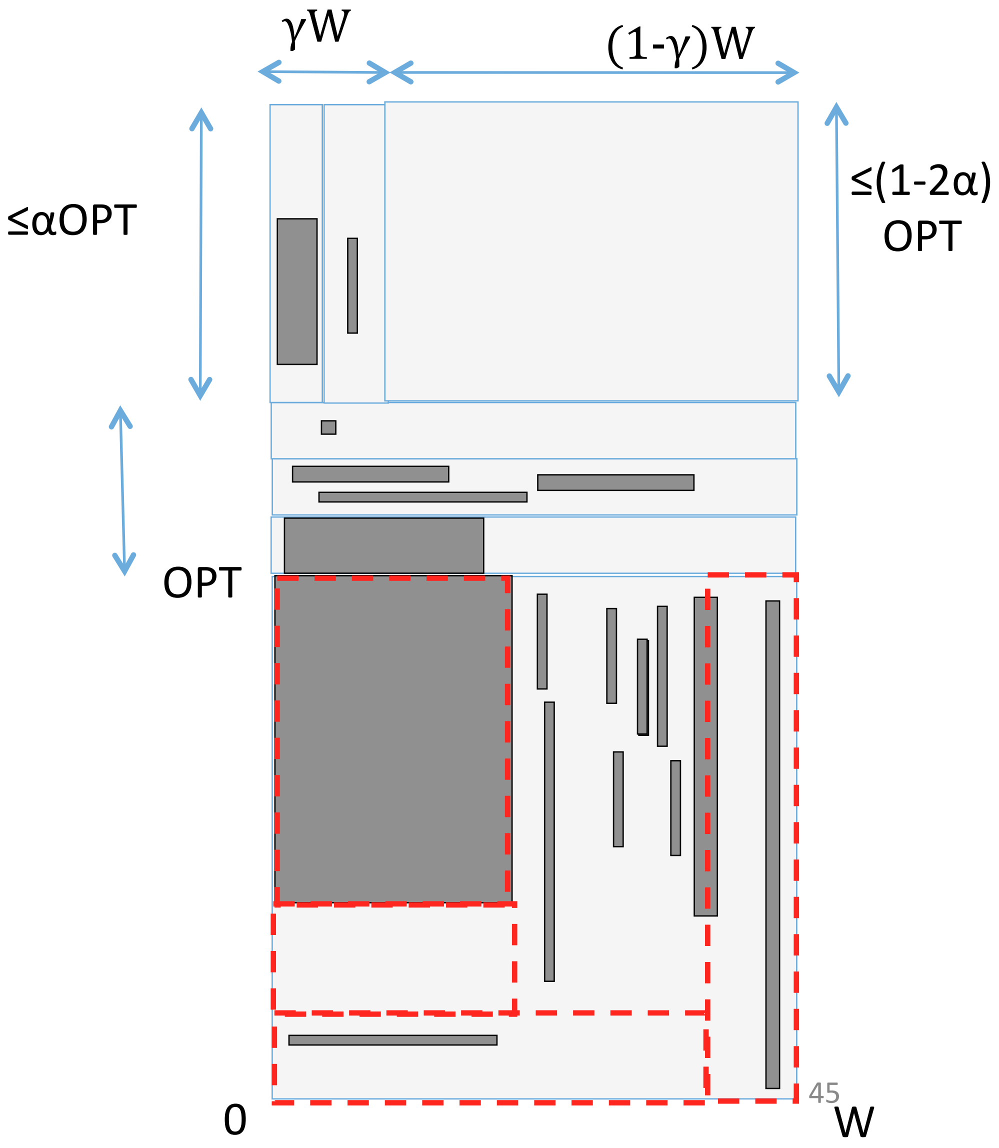


- Remove all free rectangles.
- Rearrange tall and pseudo rectangles. (same heights are grouped together as much as possible).
- Removed free rectangles are packed into two strips $W/2 \times (1\text{-}2\alpha)OPT$.

- Nadiradze-Wiese:
  For α=2/5, α=2(1-2α); 7/5 Approximation.

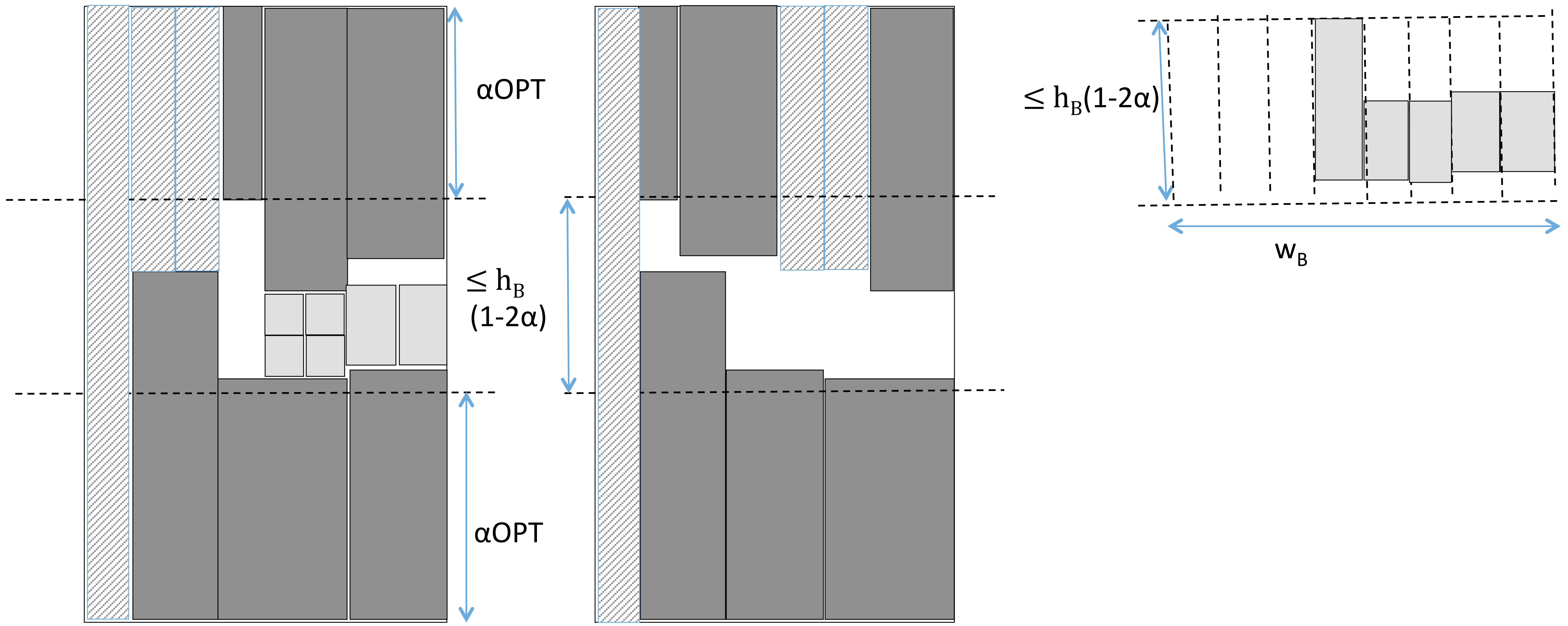


- Our packing: For α=1/3, α=(1-2α); 4/3 Approximation



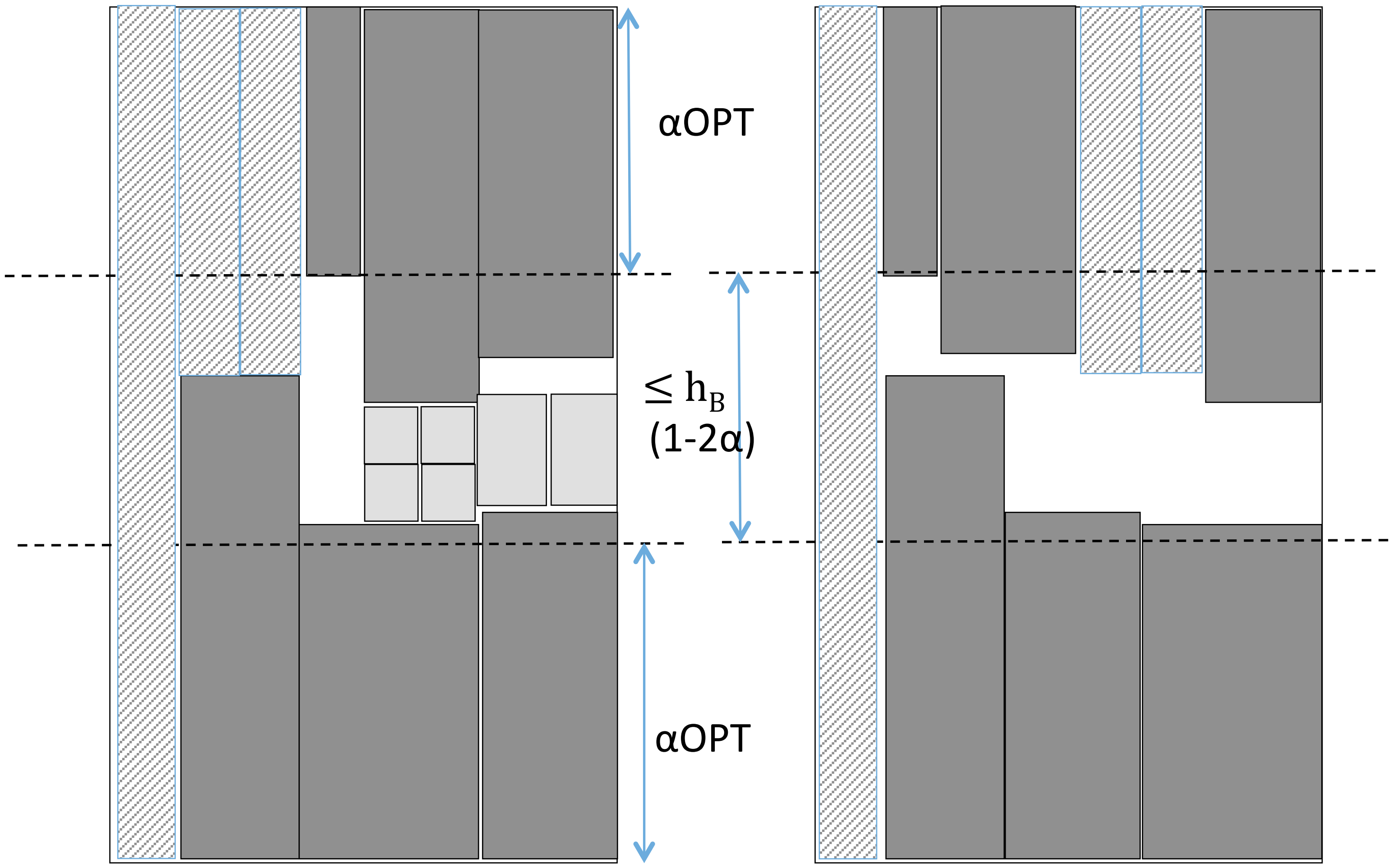
A repacking lemma:

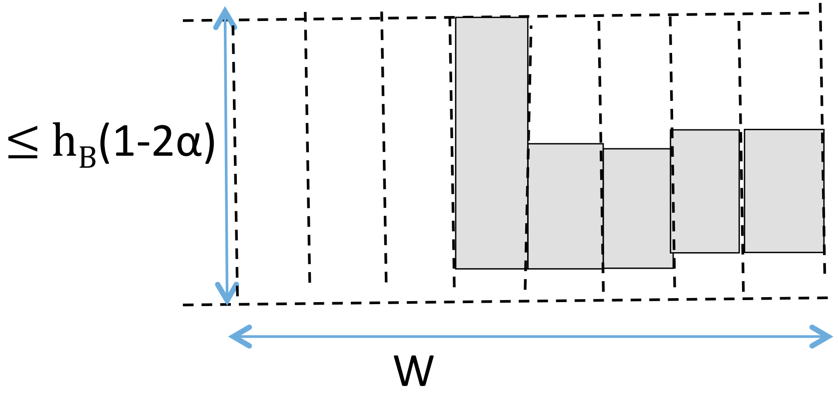
A small fraction of free rectangles can be repacked inside vertical box.



A repacking lemma:

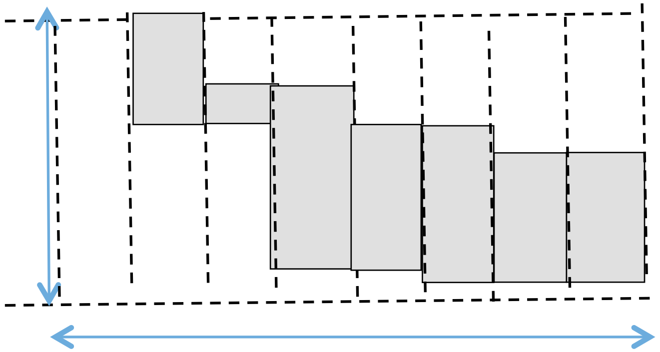
A small fraction of free rectangles can be repacked inside vertical box.





f(i)=height of i'th free rectangle.

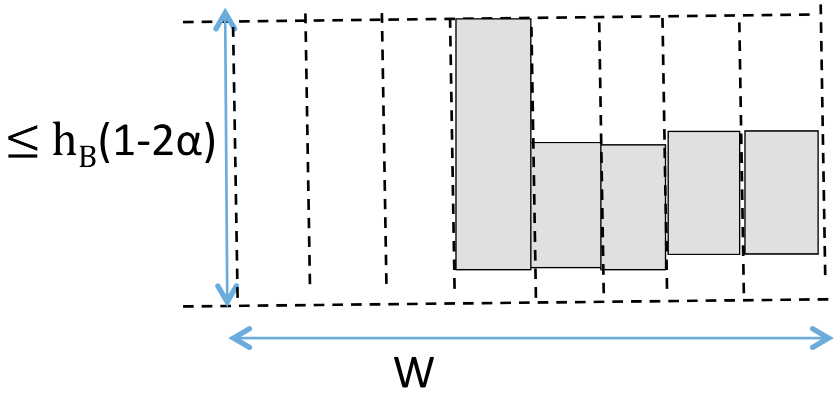
g(i)=height of i'th newly free rectangle



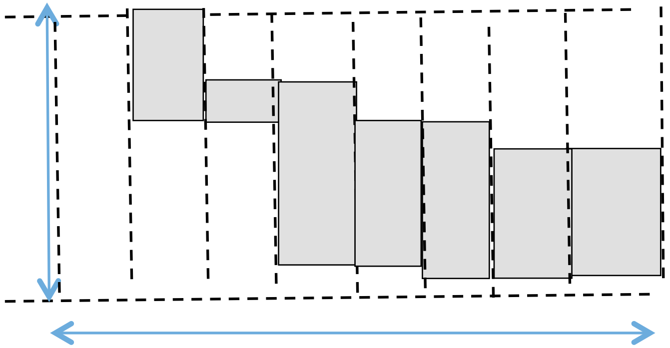
A repacking lemma:

A small fraction of free rectangles can be repacked inside vertical box.

- $\Sigma f(i) = \Sigma g(i)$.
- If $g(i) \geq f(i)$ => We can repack them.



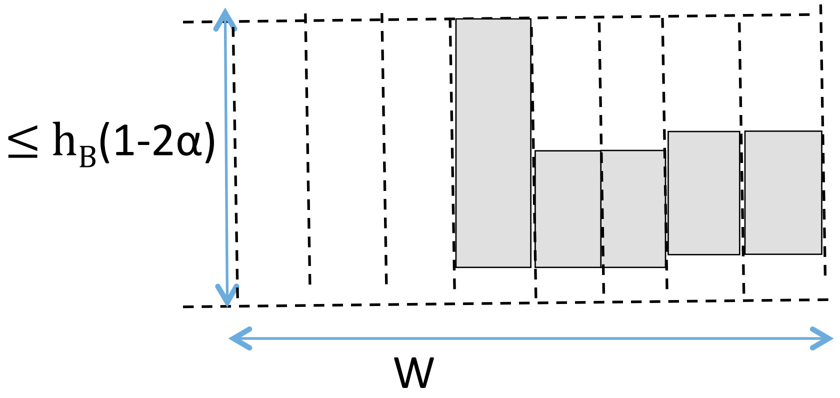
$f(i)$=height of i'th free rectangle.
$g(i)$=height of i'th newly free rectangle



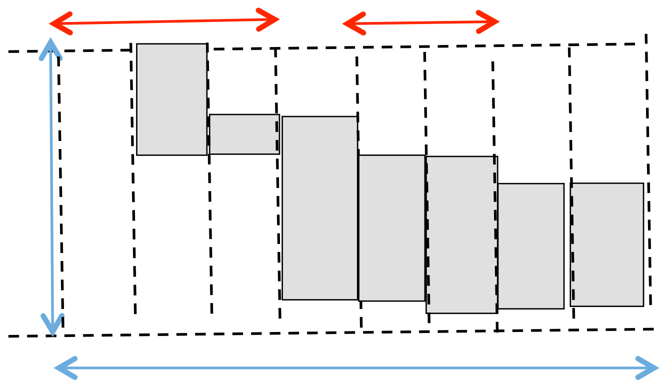
A repacking lemma:

A small fraction of free rectangles can be repacked inside vertical box.

- $\Sigma\, f(i) = \Sigma\, g(i)$.
- If $g(i) \geq f(i)$ => We can repack them.
- Let G be indices with $g(i) \geq f(i)$.



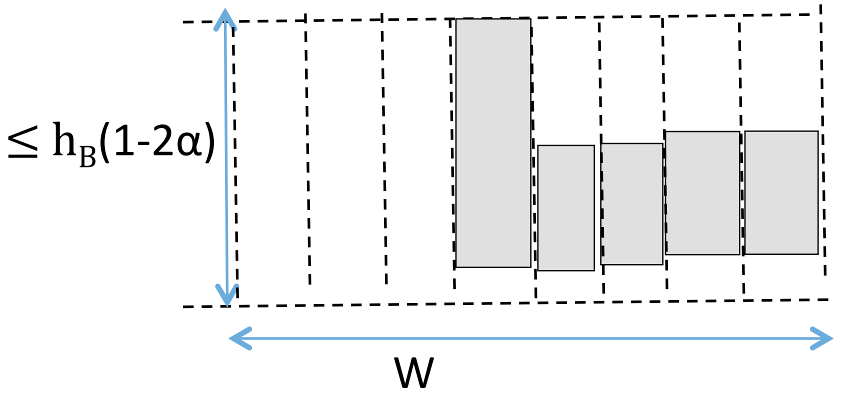
f(i)=height of i'th free rectangle.
g(i)=height of i'th newly free rectangle



A repacking lemma:

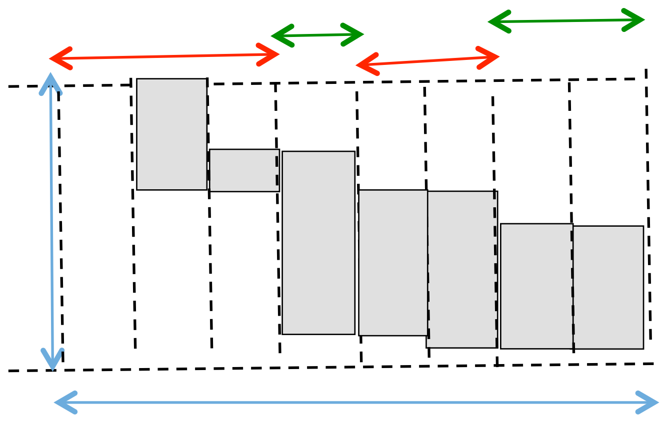
A small fraction of free rectangles can be repacked inside vertical box.

- $\Sigma\, f(i) = \Sigma\, g(i)$.
- If $g(i) \geq f(i)$ => We can repack them.
- Let G be indices with $g(i) \geq f(i)$.
- Let G’ be indices with $g(i) < f(i)$.



$f(i)$=height of i’th free rectangle.

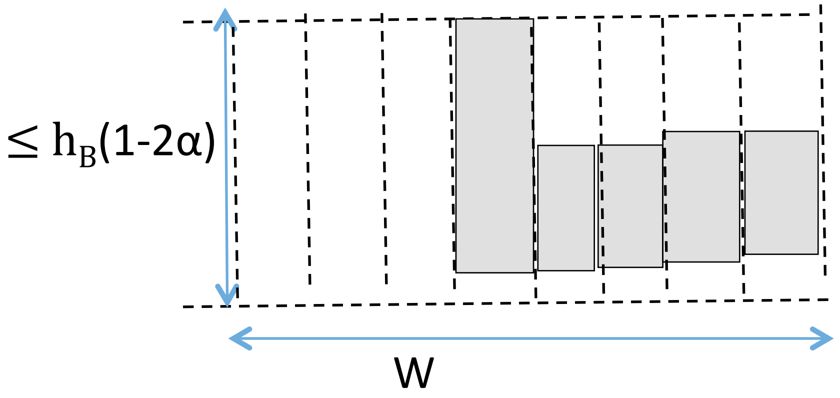
$g(i)$=height of i’th newly free rectangle



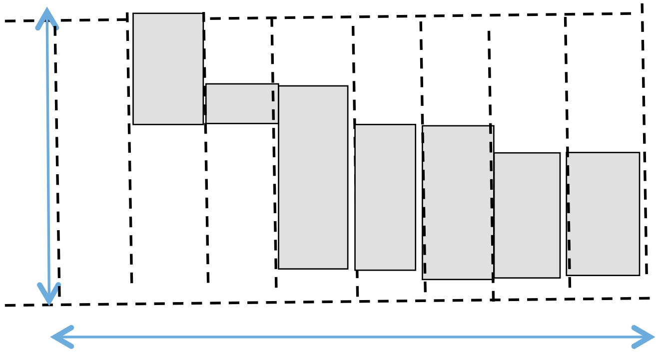
A repacking lemma:

A small fraction of free rectangles can be repacked inside vertical box.

- Σ f(i) = Σ g(i).
- If g(i) $\geq$ f(i) => We can repack them.
- Let G be indices with g(i) $\geq$ f(i) .
- Let G' be indices with g(i) < f(i).
- $$(1\text{-}2\alpha)h_B.|G| \geq \Sigma_{\{i \text{ in } G\}}\, g(i)\text{-}f(i) = \Sigma_{\{i \text{ in } G'\}}\, f(i)\text{-}g(i) \geq (w_B\text{-}|G|).\ \gamma h_B$$
- $|G| \geq w_B\,.\gamma.$

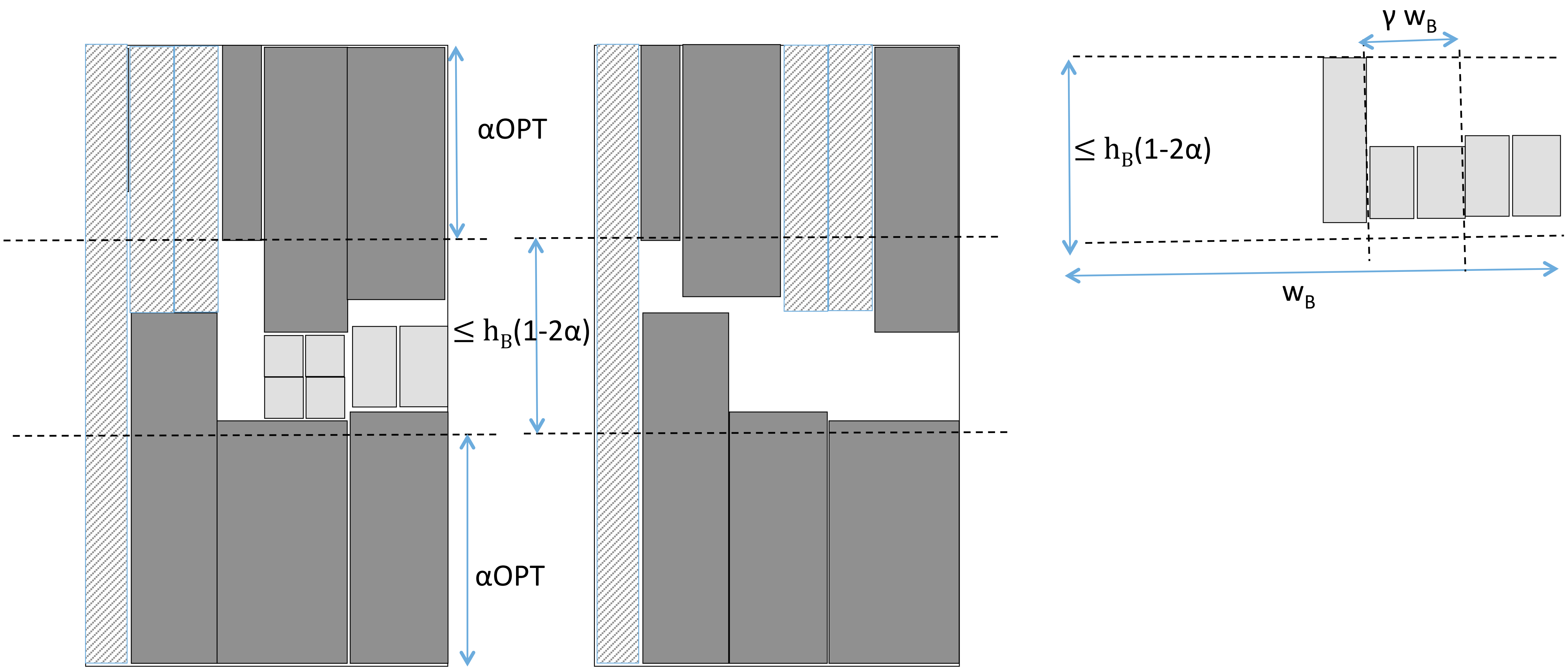


f(i)=height of i'th free rectangle.
g(i)=height of i'th newly free rectangle



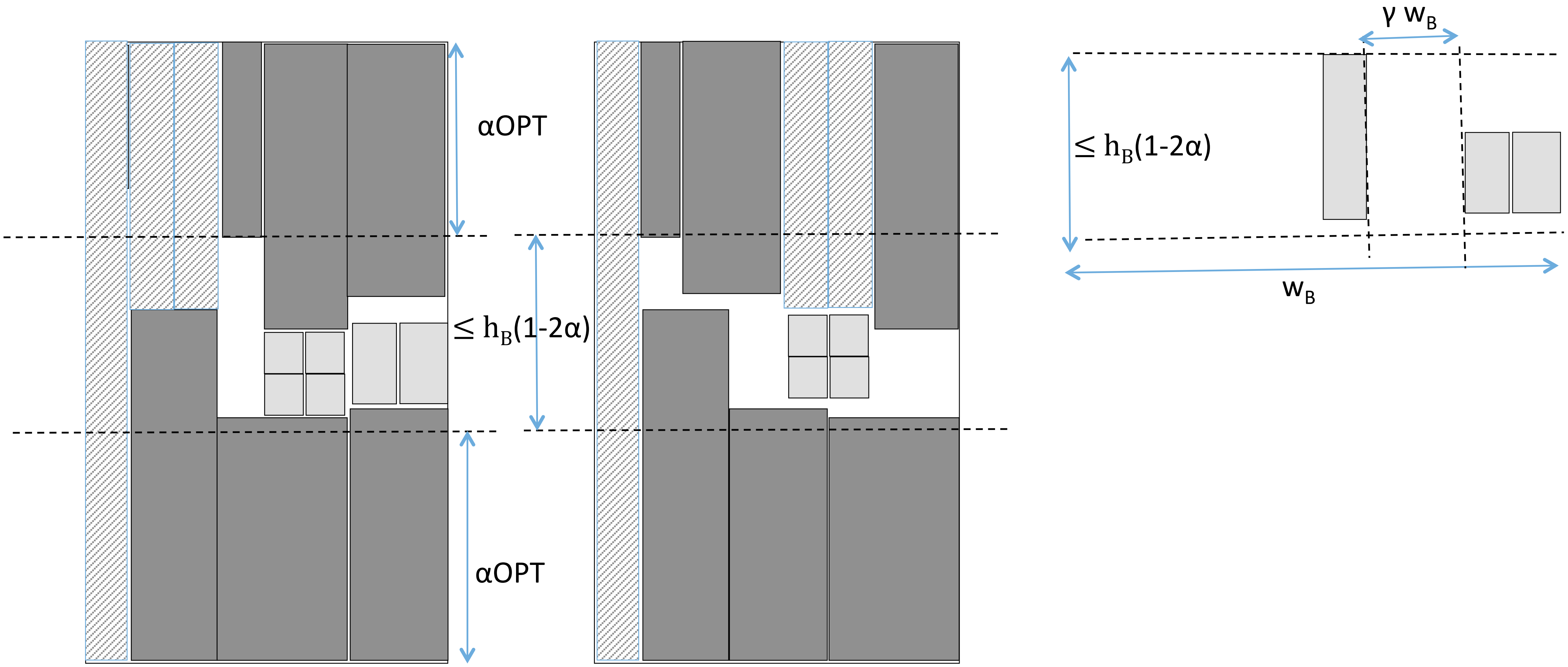
A repacking lemma:

A small fraction of free rectangles can be repacked inside vertical box.



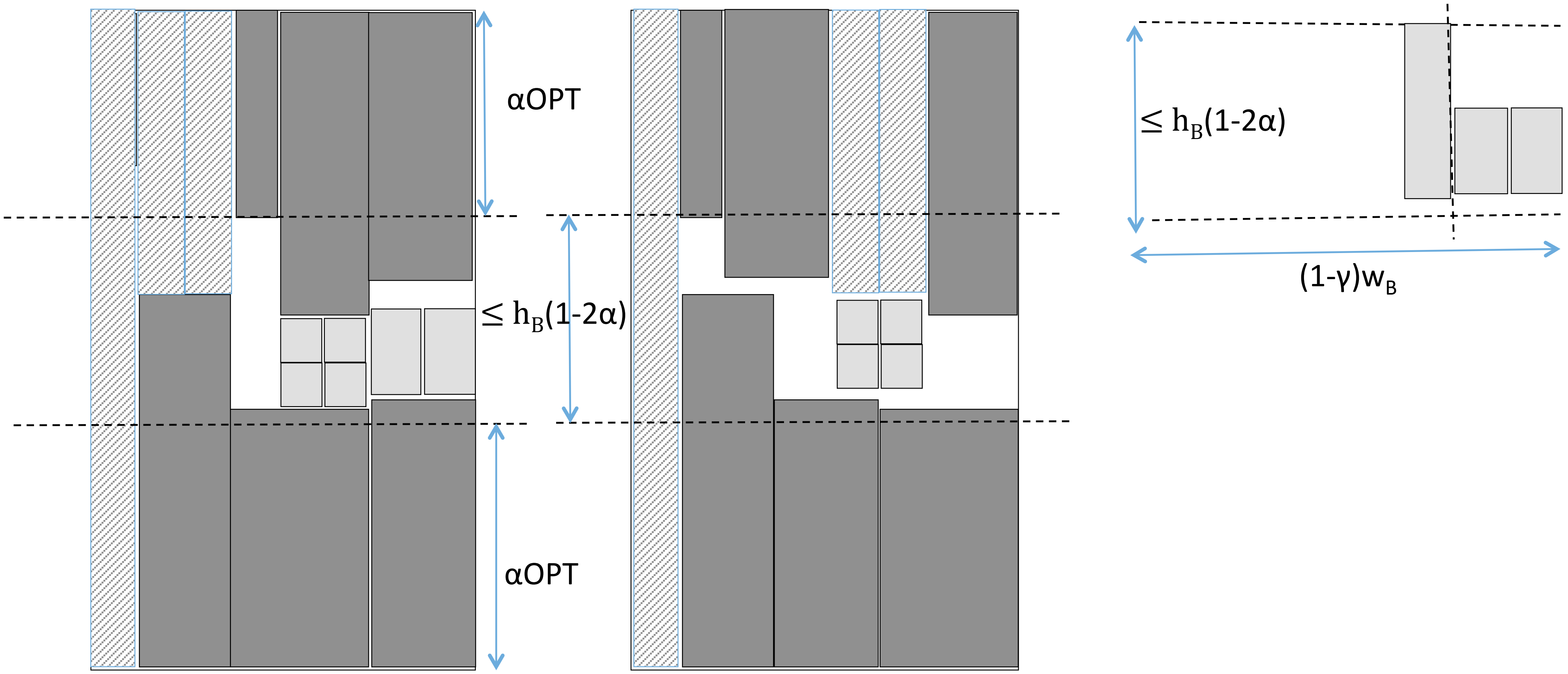
A repacking lemma:

A small fraction of free rectangles can be repacked inside vertical box.

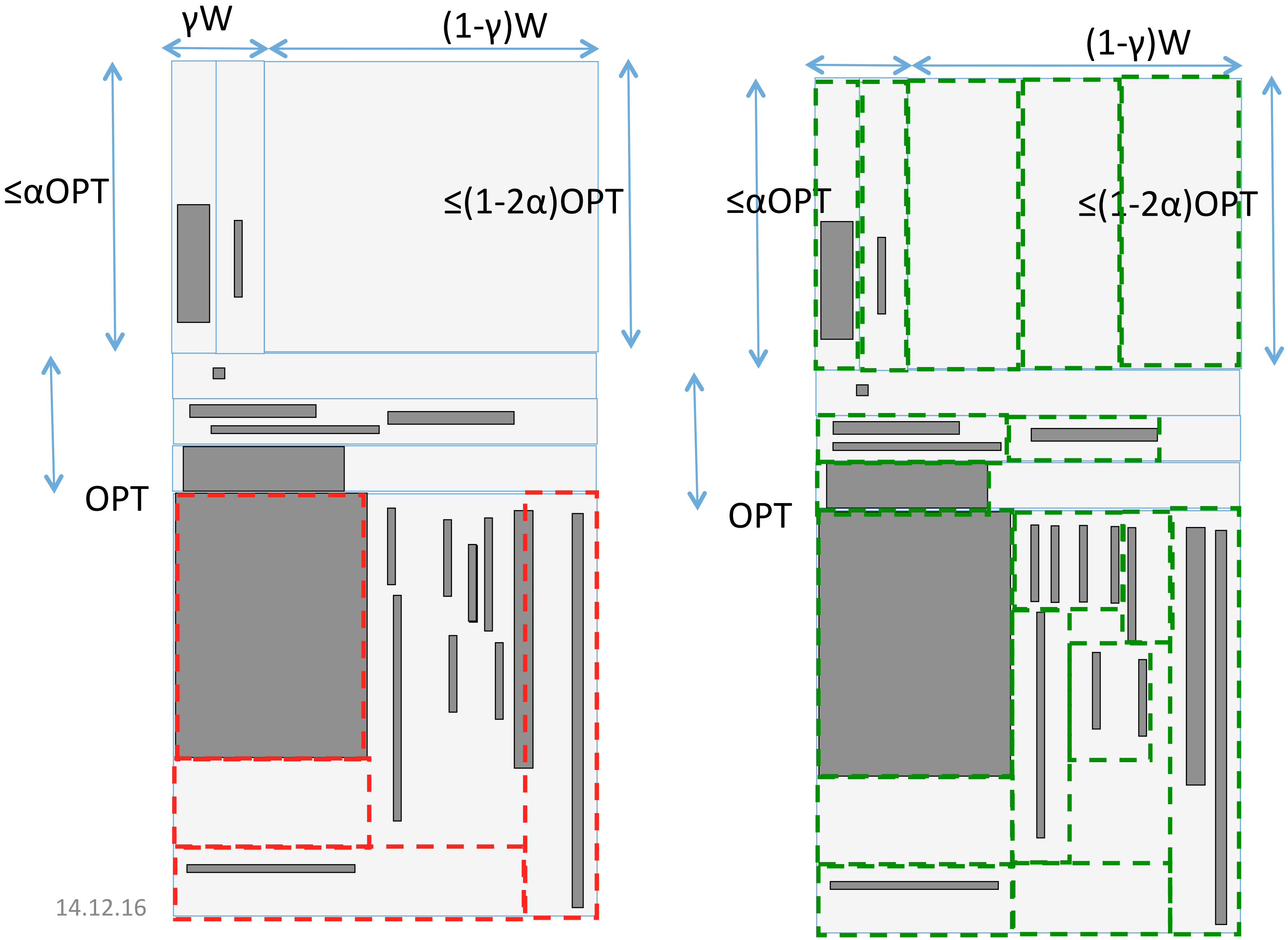


A repacking lemma:

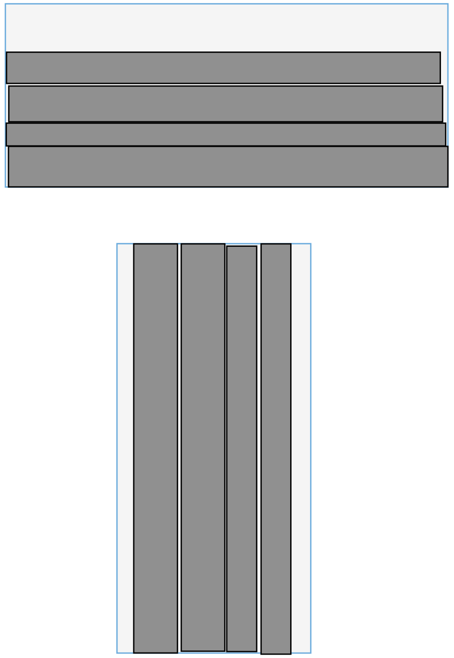
A small fraction of free rectangles can be repacked inside vertical box.



Existence of container-based packing



- For α=1/3, α=(1-2α)
  => 4/3 Approximation.
- Each box can be decomposed into O(1) number of containers.



The algorithm

- Find sizes and positions of containers in the container-based packing of all rectangles in $L \cup T \cup V \cup H$ in strip height $\leq (4/3+\varepsilon)OPT$.
- Pack non-small rectangles using dynamic program for Multiple knapsack.
- Pack small rectangles greedily in the remaining space using Next-Fit-Decreasing-Height

The algorithm

- Find sizes and positions of containers in the container-based packing of all rectangles in L∪T∪V∪H in strip height $\leq (4/3+\varepsilon)\mathrm{OPT}$.
- Pack non-small rectangles using dynamic program for Multiple knapsack.
- Pack small rectangles greedily in the remaining space using Next-Fit-Decreasing-Height

- From Existential packing, all non-small rectangles are packed into O(1) containers.
- Each container has size and position in $\{0, \ldots, W\} \times \{0, \ldots, nh_{max}\}$.
- So we can enumerate all possible such packings in pseudo-polytime.

The algorithm

- Find sizes and positions of containers in the container-based packing of all rectangles in $L \cup T \cup V \cup H$ in strip height $\leq (4/3+\varepsilon)OPT$.
- Pack non-small rectangles using dynamic program for Multiple knapsack.
- Pack small rectangles greedily in the remaining space using Next-Fit-Decreasing-Height

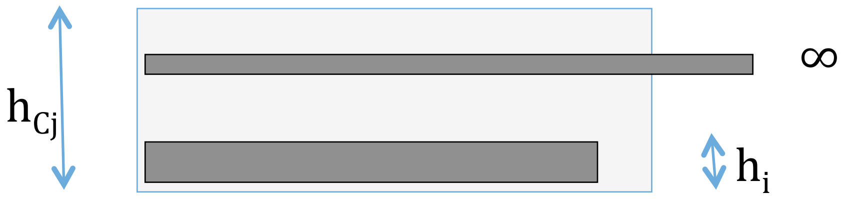
The algorithm

- Find sizes and positions of containers in the container-based packing of all rectangles in L∪T∪V∪H in strip height $\leq (4/3+\varepsilon)OPT$.
- Pack non-small rectangles using dynamic program for Multiple knapsack.
- Pack small rectangles greedily in the remaining space using Next-Fit-Decreasing-Height

- For horizontal (or vertical) container j := $(w_{Cj} \times h_{Cj})$, create knapsack of size h_{Cj} (or w_{Cj}).
- For rectangle R_i, define size w.r.t. knapsack j:

  $= h_i$ if it fits in the container.

  $= \infty$ otherwise.

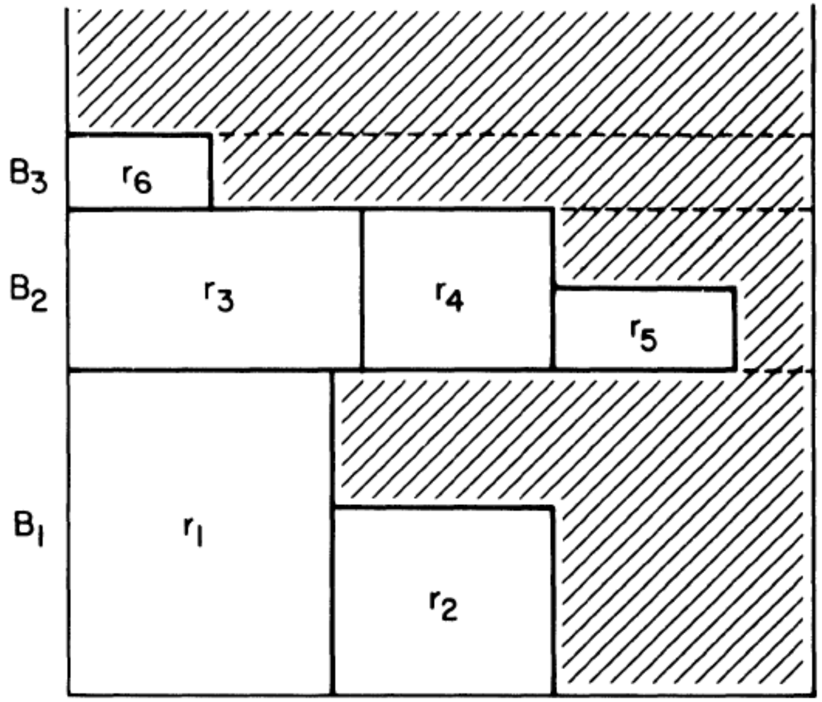


The algorithm

- Find sizes and positions of containers in the container-based packing of all rectangles in L∪T∪V∪H in strip height $\leq (4/3+\varepsilon)\mathrm{OPT}$.
- Pack non-small rectangles using dynamic program for Multiple knapsack.
- Pack small rectangles greedily in the remaining space using Next-Fit-Decreasing-Height

The algorithm

- Find sizes and positions of containers in the container-based packing of all rectangles in L∪T∪V∪H in strip height $\leq (4/3+\varepsilon)\mathrm{OPT}$.
- Pack non-small rectangles using dynamic program for Multiple knapsack.
- Pack small rectangles greedily in the remaining space using Next-Fit-Decreasing-Height



With Rotations!

- N-W Algorithm packed horizontal rectangles using an LP.
  Not clear:
  1. which rectangles are packed using the LP.
  2. which rectangles are small.

With Rotations!

- For container-based packing, we can assign all rectangles using multiple knapsack.
- Pack small rectangles greedily in the remaining space using Next-Fit-Decreasing-Height

- For horizontal (or vertical) container j := (w_{Cj} x h_{Cj}), create knapsack of size h_{Cj} (or w_{Cj}).
- For rectangle R_i, define size w.r.t. horizontal knapsack j:
  $= \min\{h_i , w_i\}$, if it fits both rotated and nonrotated
  $= h_i$, if it fits only rotated
  $= w_i$, if it fits only nonrotated
  $= \infty$ otherwise.
- Extra knapsack (for small rectangles) of size
  = area not occupied by nonsmall rectangles in OPT.
  If a rectangle R_i is small w.r.t. current parameters as rotated or nonrotated, its size= area of R_i.

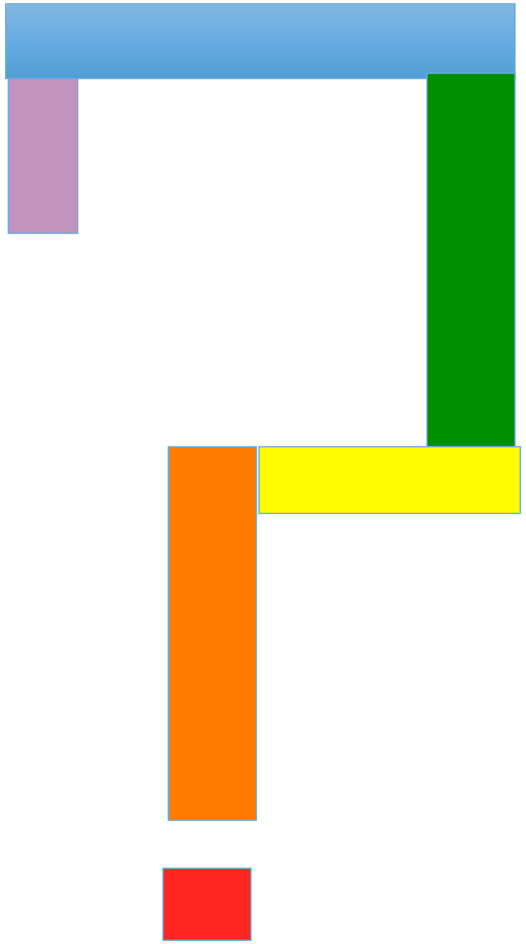
Open Problems

- Tight polynomial-time approximation for strip packing.
- Better Pseudo-polytime hardness/approximation algorithm.

  (Adamaszek et al., No Pseudo-polytime approximation scheme; Arxiv - Oct'16)
- Extension to d-dimensional strip packing.

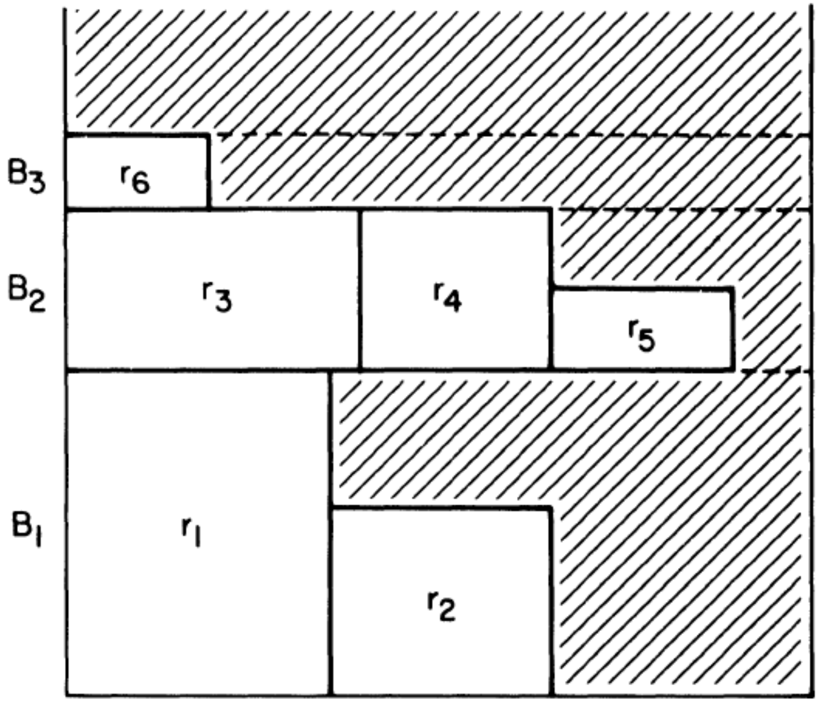
- More related literature and open problems:
  Approximation and Online Algorithms for Multidimensional Bin Packing: A Survey, Christensen-K.-Pokutta-Tetali.

Questions!



Additional Slides

Next Fit Decreasing Height(NFDH)

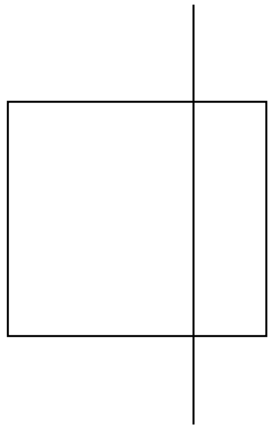


- Considered items in a non-increasing order of height and greedily packs items into shelves.
- Shelf is a row of items having their bases on a line that is either the base of the bin or the line drawn at the top of the highest item packed in the shelf below.
- items are packed left-justified starting from bottom-left corner of the bin, until the next item does not fit. Then the shelf is closed and the next item is used to define a new shelf whose base touches the tallest(left most) item of the previous shelf.
- If the shelf does not fit into the bin, the bin is closed and a new bin is opened. The procedure continues till all the items are packed.

- If we pack small rectangles ($w,h \leq \delta$) using NFDH into B, total $w.h-(w+h).\delta$ area can be packed.

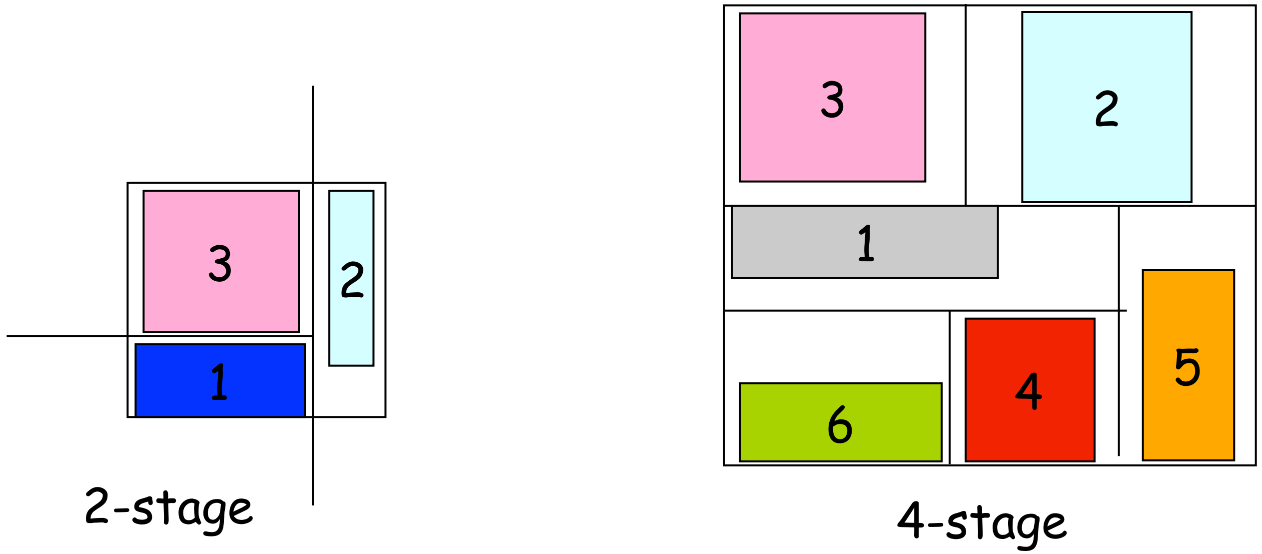
Guillotine Bin Packing

Guillotine Cut: Edge to Edge cut across a bin



Guillotine Bin Packing

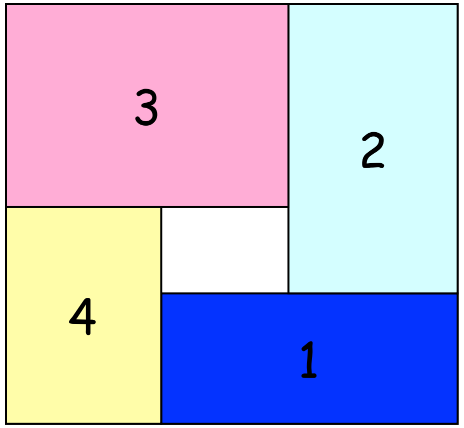
Guillotine Cut: Edge to Edge cut across a bin



k-stage Guillotine Packing [Gilmore, Gomory]

k recursive levels of guillotine cuts to recover all items.

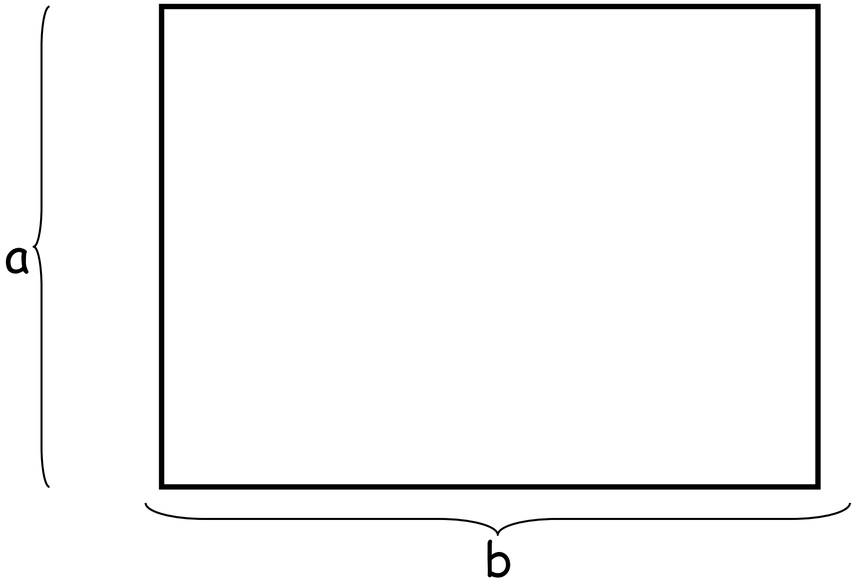
Non-guillotine Packing



Shelf Packing

Given a rectangular region of size a £ b

Goal: Pack squares of length · s

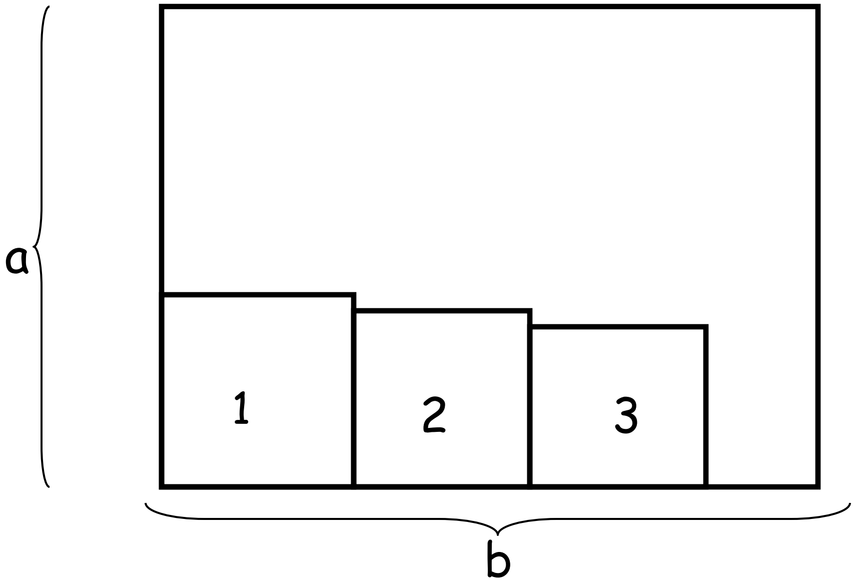


Shelf Packing

Given a rectangular region of size $a \times b$

Goal: Pack squares of length $\leq s$

Algorithm: Decreasing size shelf packing.



Take squares in decreasing size

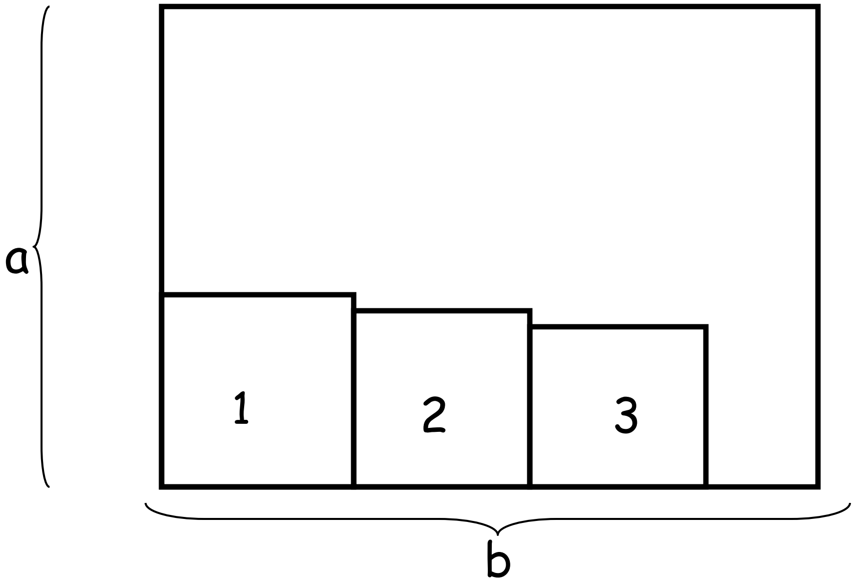
- Place sequentially

Shelf Packing

Given a rectangular region of size a £ b

Goal: Pack squares of length ≤ s

Algorithm: Decreasing size shelf packing.



Take squares in decreasing size

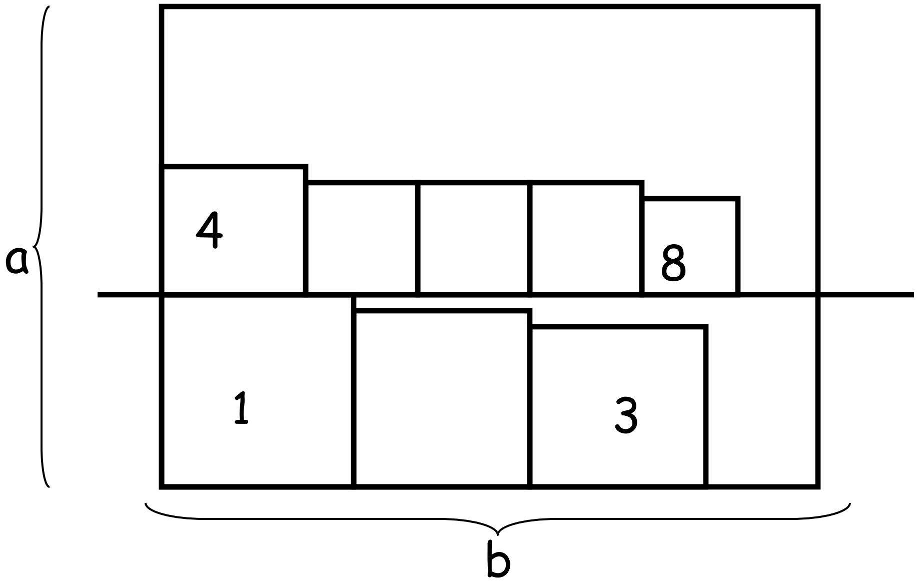
- Place sequentially
- If next does not fit, open a new shelf

Shelf Packing

Given a rectangular region of size a £ b

Goal: Pack squares of length · s

Algorithm: Decreasing size shelf packing.



Take squares in decreasing size

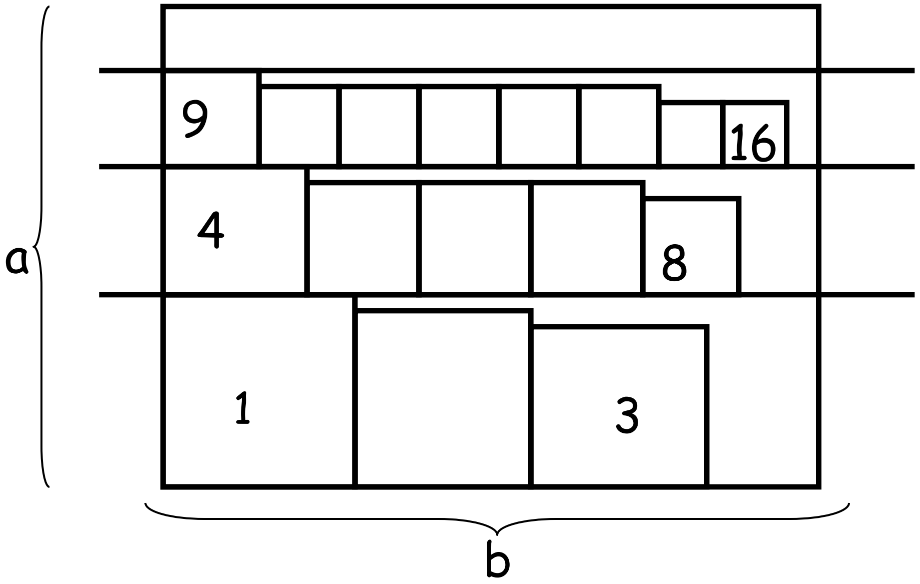
- Place sequentially
- If next does not fit, open a new shelf

Shelf Packing

Given a rectangular region of size a £ b

Goal: Pack squares of length · s

Algorithm: Decreasing size shelf packing.



Take squares in decreasing size

- Place sequentially
- If next does not fit, open a new shelf

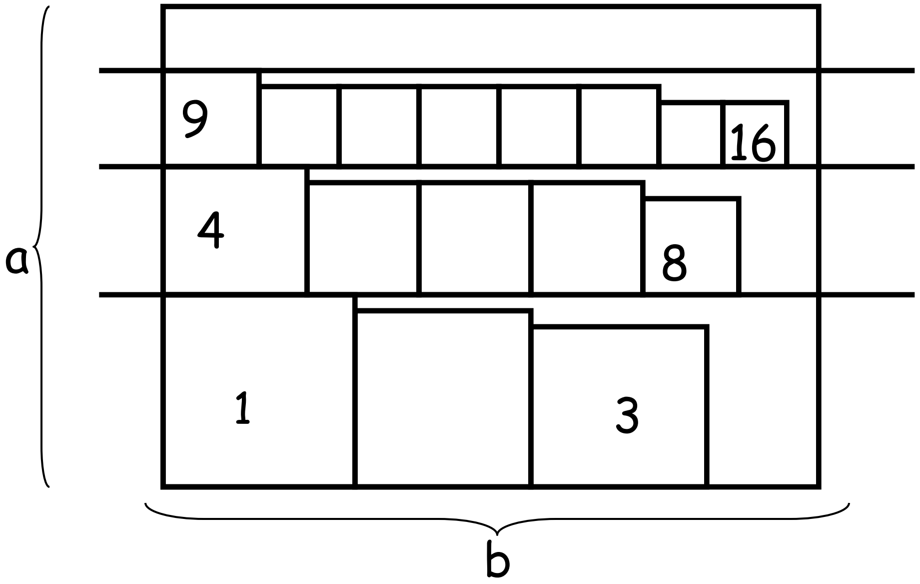
Shelf Packing

Given a rectangular region of size a £ b

Goal: Pack squares of length · s

Algorithm: Decreasing size shelf packing.

Wasted Space · s(a+b)

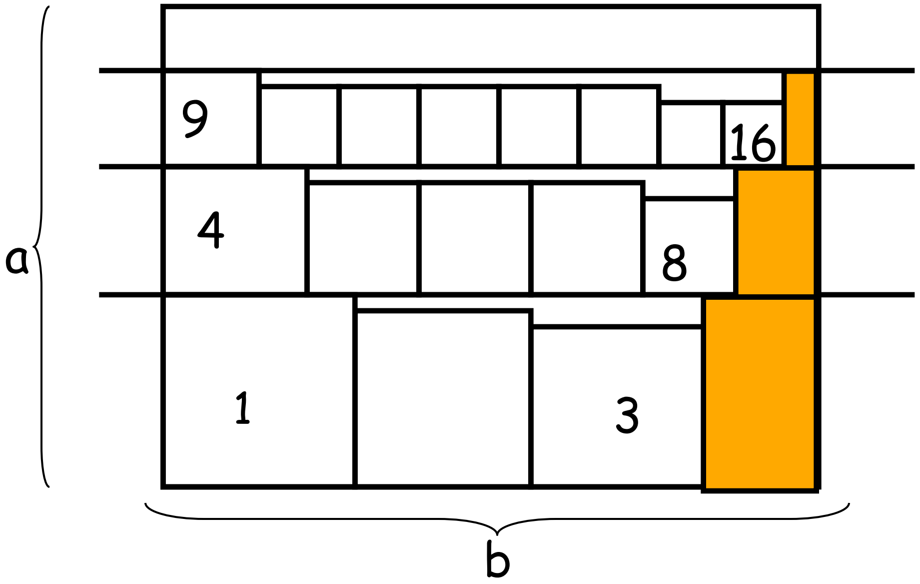


Shelf Packing

Given a rectangular region of size a x b

Goal: Pack squares of length · s

Algorithm: Decreasing size shelf packing.



Wasted Space · s(a+b)

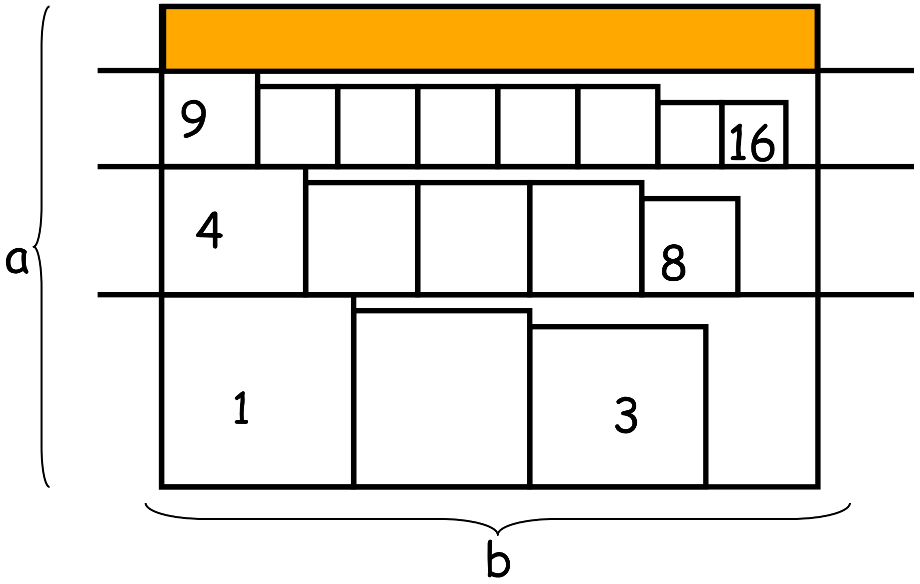
Right side: At most s £ a

Shelf Packing

Given a rectangular region of size a £ b

Goal: Pack squares of length · s

Algorithm: Decreasing size shelf packing.



Wasted Space · s(a+b)

Right side: At most s £ a

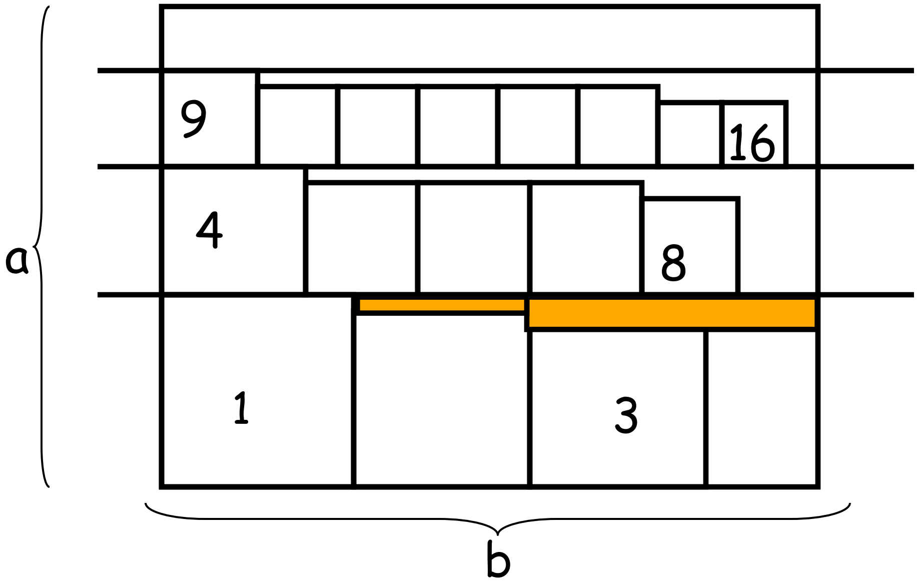
Top · s_{16} b

Shelf Packing

Given a rectangular region of size a £ b

Goal: Pack squares of length · s

Algorithm: Decreasing size shelf packing.



Wasted Space · s(a+b)

Right side: At most s £ a

Top · s_{16} b

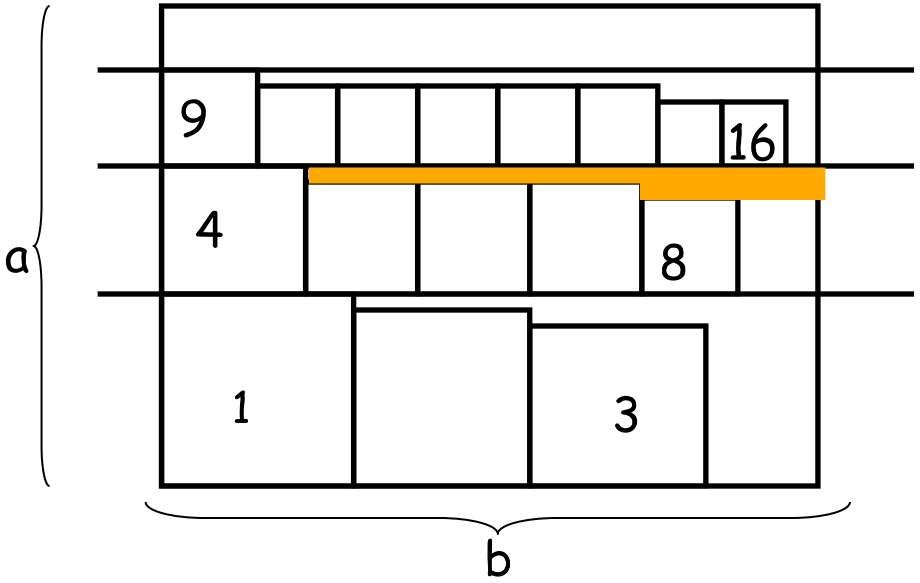
Shelf 1: $(s_1 - s_3)$ b

Shelf Packing

Given a rectangular region of size a £ b

Goal: Pack squares of length · s

Algorithm: Decreasing size shelf packing.



Wasted Space · s(a+b)

Right side: At most s £ a
Top · s_{16} b

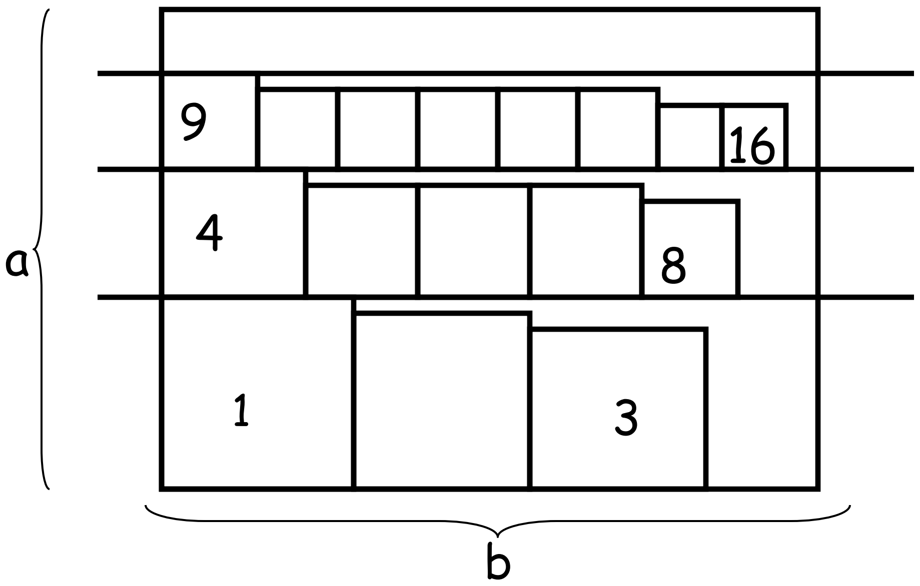
Shelf 1: $(s_1 - s_3)$ b
Shelf 2: $(s_4 - s_8)$ b
…

Shelf Packing

Given a rectangular region of size a £ b

Goal: Pack squares of length · s

Algorithm: Decreasing size shelf packing.



Wasted Space · s(a+b)

Right side: At most s £ a
Top · s_{16} b

Shelf 1: $(s_1 - \cancel{s_3})$ b
Shelf 2: $(\cancel{s_4} - \cancel{s_8})$ b
….
Adding all, at most $(s_1 - s_{16})$ b