Approximating Geometric Knapsack via L-packings

Arindam Khan

IDSIA, Lugano, Switzerland

Joint work with
Waldo Galvez, Fabrizio Grandoni, Salvatore Ingala,
Sandy Heydrich, Andreas Wiese.

Geometric Knapsack: (2-D)

Geometric Knapsack: (2-D)

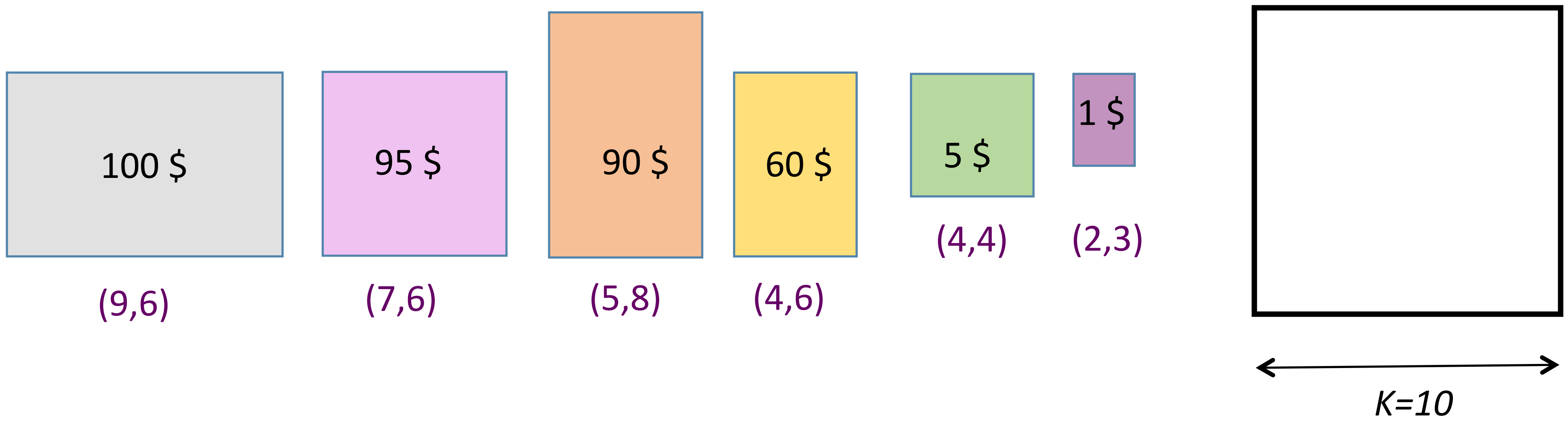
- Input :

  - Rectangles $I := \{R_1, R_2, \ldots, R_n\}$; Each R_i has integral width and height (w_i, h_i) and profit p_i.

  - A Square $K \times K$ knapsack.

Geometric Knapsack: (2-D)

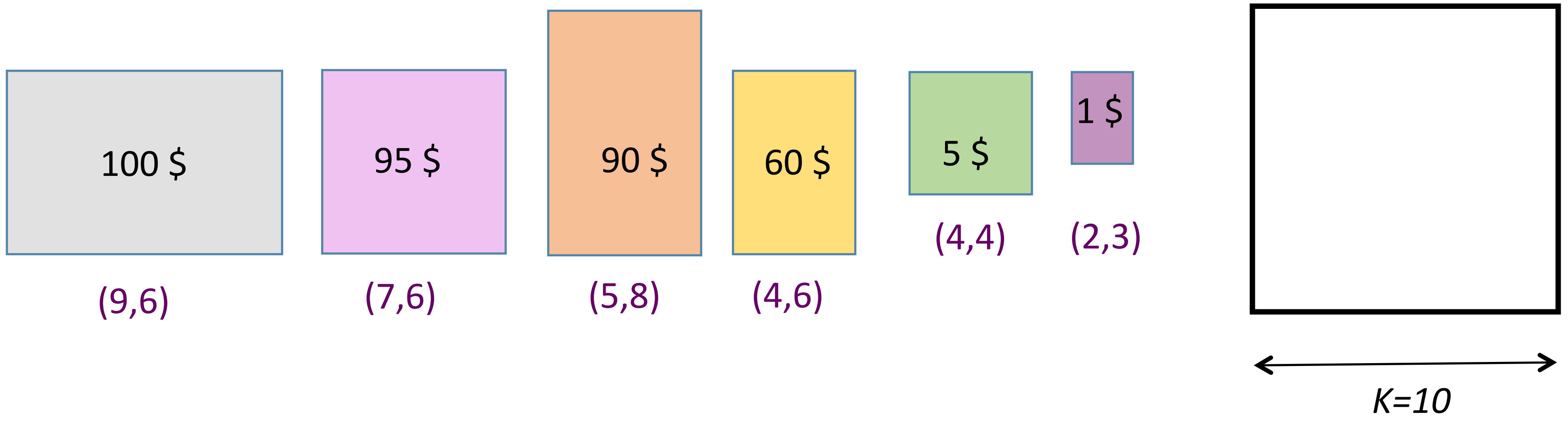
- Input :
  - Rectangles $I := \{R_1, R_2, \ldots, R_n\}$; Each R_i has integral width and height (w_i, h_i) and profit p_i.
  - A Square $K \times K$ knapsack.



Geometric Knapsack: (2-D)

- Input :
  - Rectangles $I := \{R_1, R_2, \ldots, R_n\}$; Each R_i has integral width and height (w_i, h_i) and profit p_i.
  - A Square $K \times K$ knapsack.

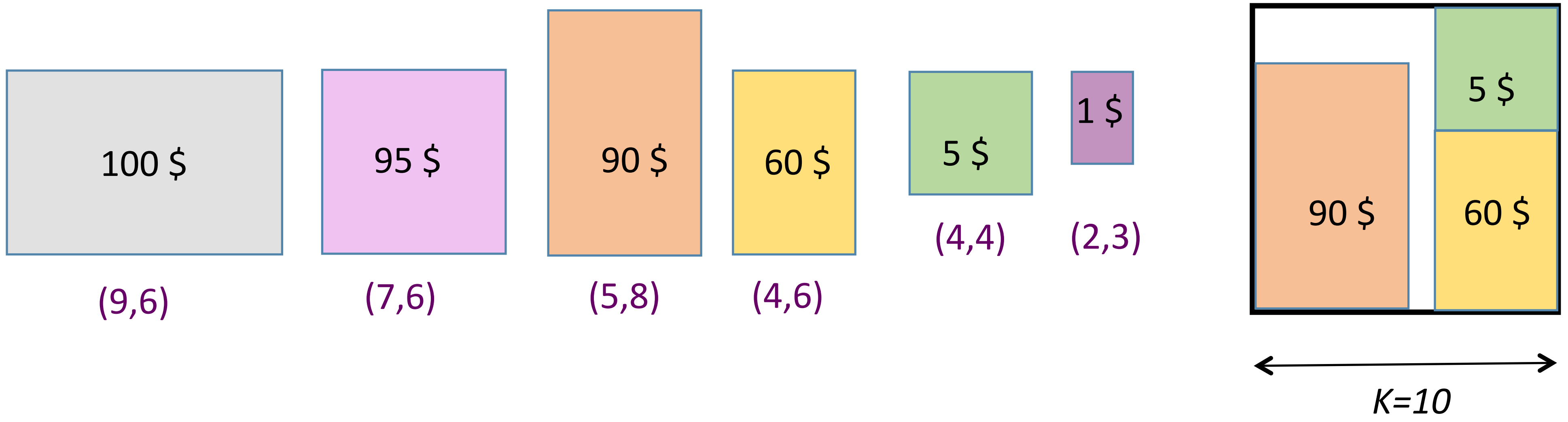
- Goal : Find an axis-parallel non-overlapping packing of a subset of input rectangles into the knapsack that maximizes the total profit.



Geometric Knapsack: (2-D)

- Input :
  - Rectangles $I := \{R_1, R_2, \ldots, R_n\}$; Each R_i has integral width and height (w_i, h_i) and profit p_i.
  - A Square $K \times K$ knapsack.

- Goal : Find an axis-parallel non-overlapping packing of a subset of input rectangles into the knapsack that maximizes the total profit.



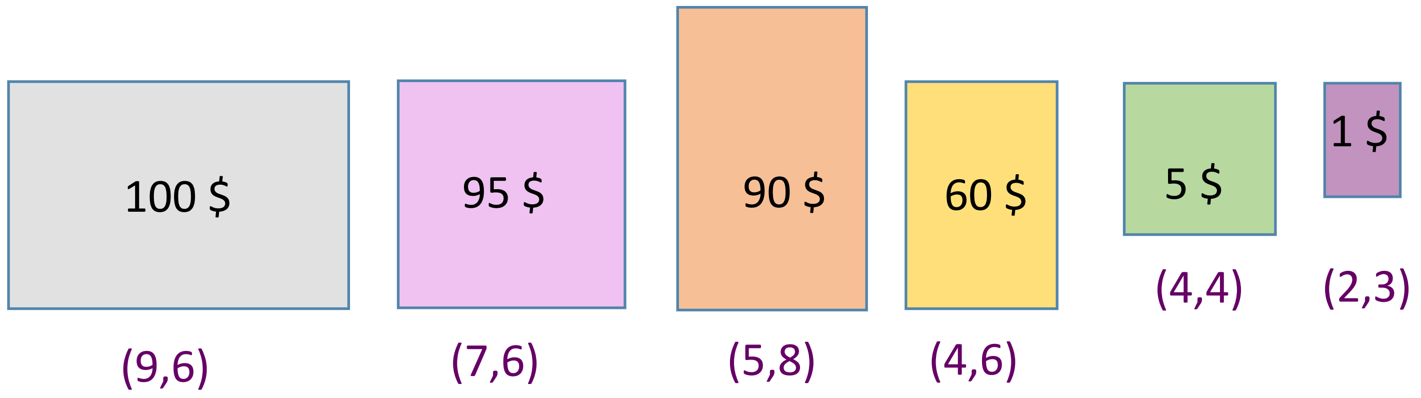
Variant 1: 2DK
No rotations are allowed!

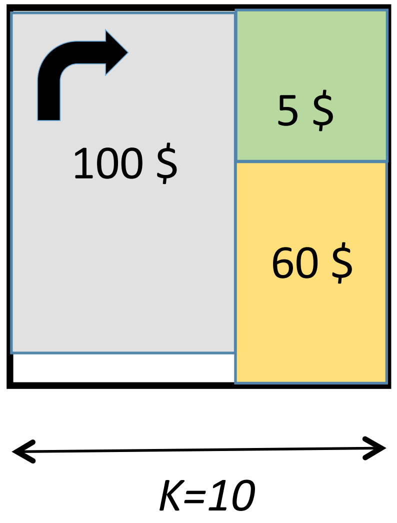
OPT=155

Geometric Knapsack: (2-D)

- Input :
  - Rectangles $I := \{R_1, R_2, \ldots, R_n\}$; Each R_i has integral width and height (w_i, h_i) and profit p_i.
  - A Square $K \times K$ knapsack.

- Goal : Find an axis-parallel non-overlapping packing of a subset of input rectangles into the knapsack that maximizes the total profit.





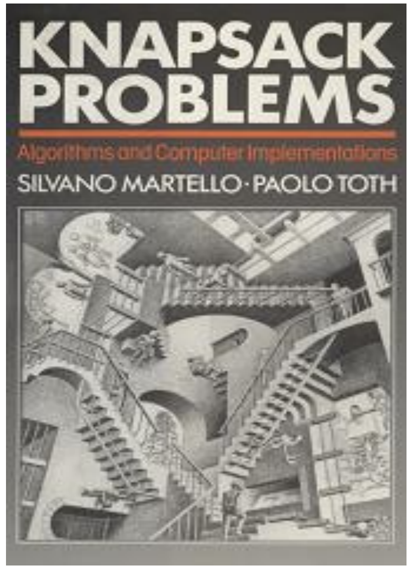
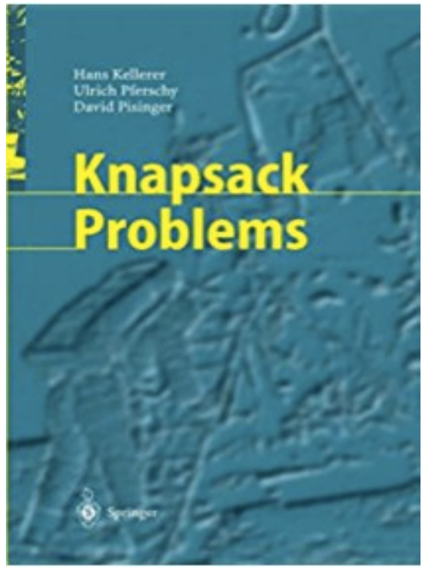
Variant 2: (2DKR)

90 degree rotations are allowed!

OPT=165

Applications:

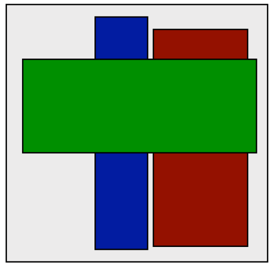
- Generalization of classical knapsack problem.
- Cutting stock: cloth cutting, steel/wood cutting.
- Logistics and Scheduling: memory allocation , truck loading, robotics.
- Ad-placements, VLSI Design.


KNAPSACK PROBLEMS
Algorithms and Computer Implementations
SILVANO MARTELLO · PAOLO TOTH


Related Problems

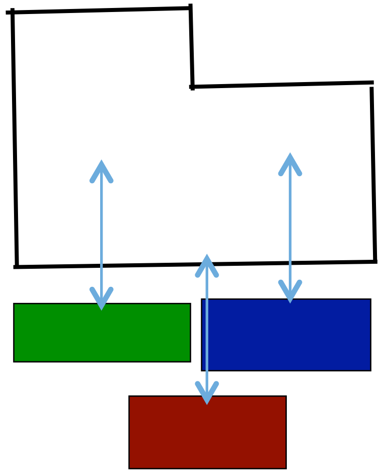
- Independent set of rectangles:

Positions of rectangles are fixed, find max profit subset



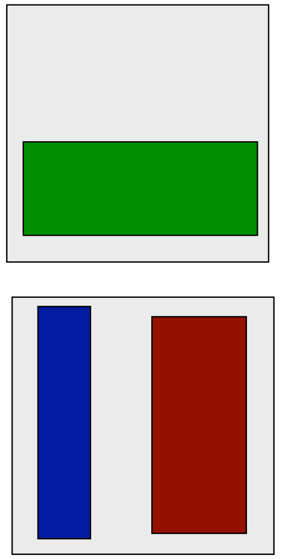
- Unsplittable flow/ Storage allocation:

Horizontal positions of rectangles are fixed, find max profit subset



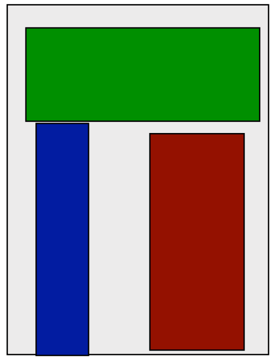
- Two Dimensional Bin Packing:

Pack all items in min # of squares



- Two Dimensional Strip Packing:

Pack all items in min height fixed-width strip



Geometric Knapsack:

- Geometric Knapsack is Strongly NP-hard (even when all items are squares with profit 1), [Leung et al., JPDC 1990].
- No exact algorithm even in pseudo-polynomial time (unless P=NP).
- So, we will consider Approximation Algorithms.
- An algorithm A is α-Approximation -- if OPT(I) ≤ α A(I) for all input instances I.

Geometric Knapsack: Prior works

- Best known approximation: (2+ε) [Jansen-Zhang, SODA'04]
  - for both with and without rotations.
  - even in the cardinality case (when all profits are 1).

- (1+ε)-approximation known if
  - profit of an item is equal to its area. [Bansal et al., ISAAC '09].
  - items are relatively small [Fishkin et al., MFCS '05].
  - items are squares [Jansen-SolisOba, MFCS '07].

Geometric Knapsack: Prior works

- Resource augmentation:
  - if knapsack size is increased from K to (1+ε)K in both [Fishkin et al. MFCS '05] or one dimension [Jansen-SolisOba, MFCS '07],
  - Profit (1-ε)OPT can be obtained in polytime.
- Quasi Polynomial Time Approximation Scheme (QPTAS):
  - Profit (1-ε)OPT can be obtained in quasi-polytime ($O(n^{polylog(n)})$),
  - assuming $K = O(n^{polylog(n)})$ [Adamaszek-Wiese, SODA '15].
- In general, (2+ε)-appx is still best known even in quasi-polytime.

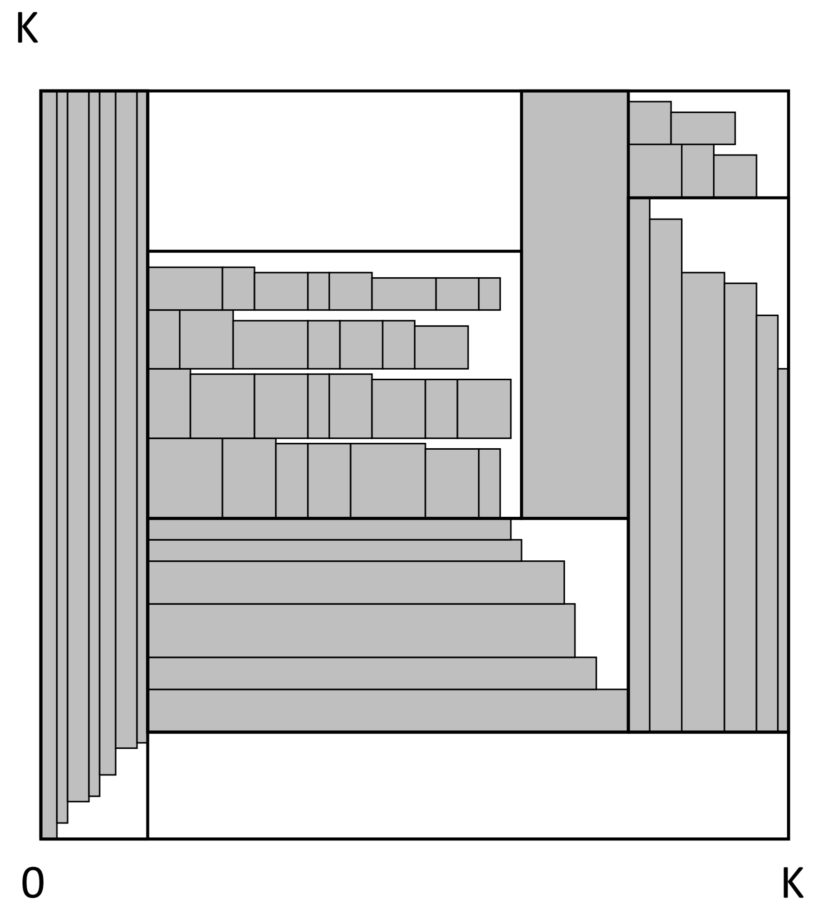
Our Results:

- General case:
- Without rotations: $(17/9+\varepsilon)<1.89$-approximation.
- With rotations: $(1.5+\varepsilon)$-approximation.
- Cardinality case:
- Without rotations: $(558/325+\varepsilon)<1.72$-approximation.
- With rotations: $(4/3+\varepsilon)$-approximation.
- In this talk we present a simpler $(16/9+\varepsilon)<1.78$-approximation for the cardinality case without rotations.

Previous approaches: container-based packing.

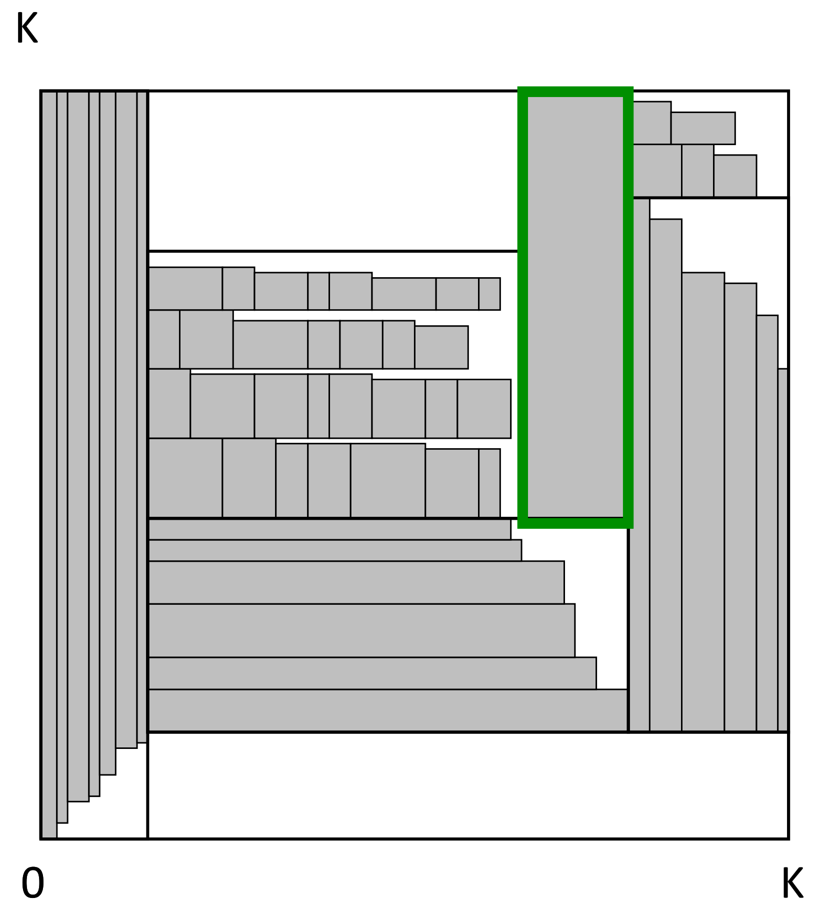
Previous approaches: container-based packing.

- Container is an axis-aligned rectangular region such that



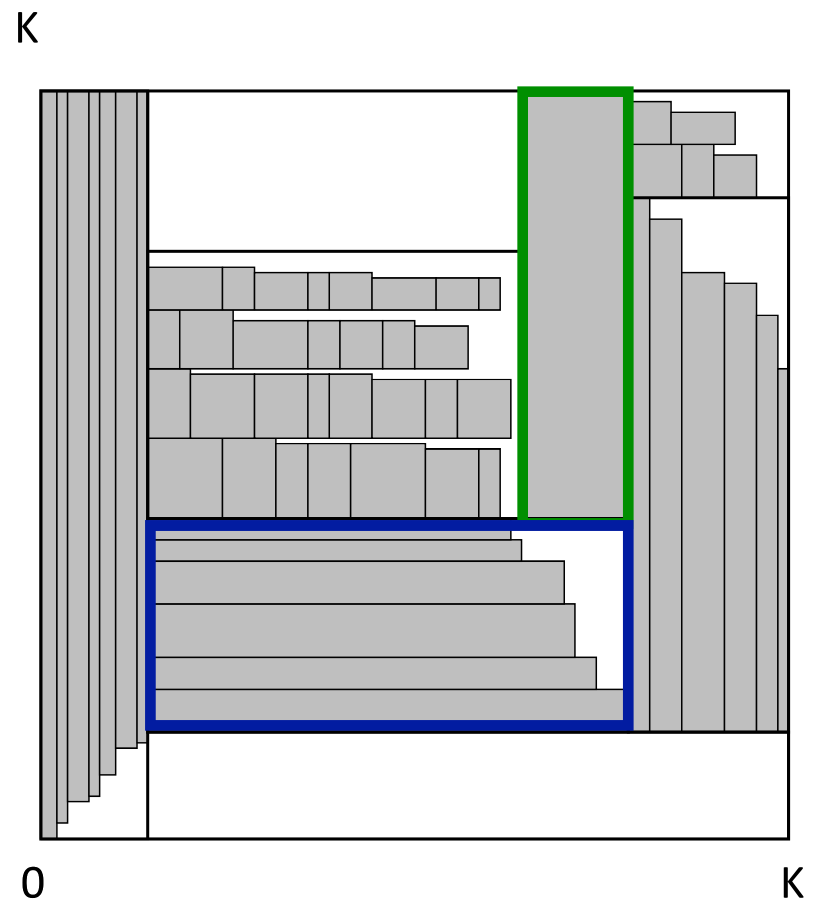
Previous approaches: container-based packing.

- Container is an axis-aligned rectangular region such that
- either it contains one large item.



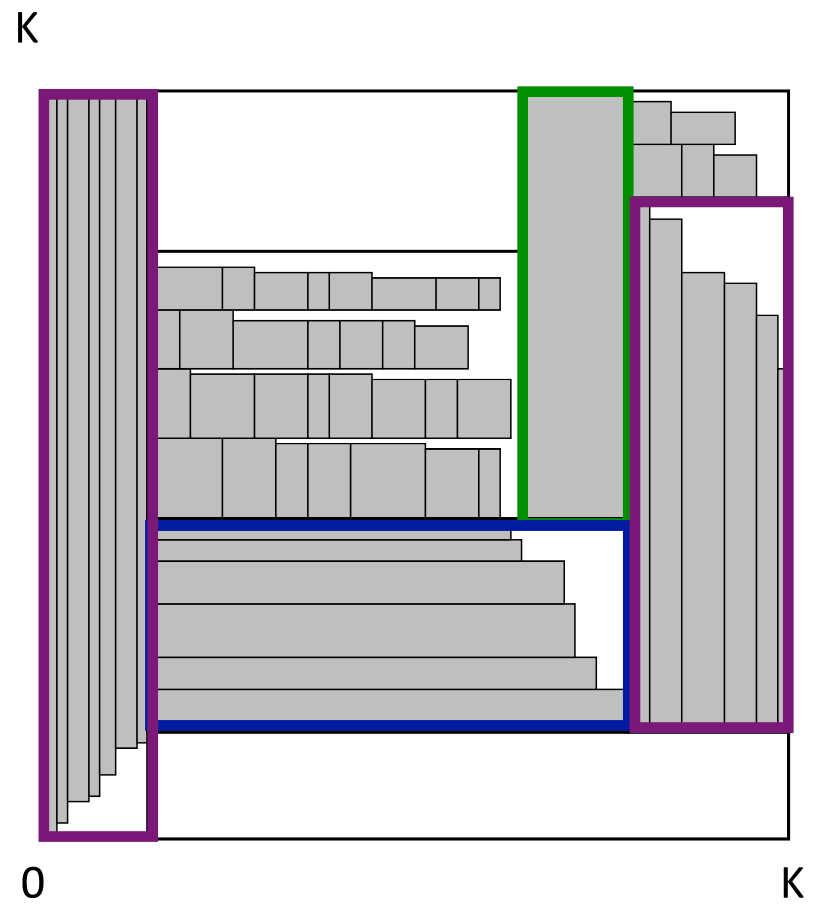
Previous approaches: container-based packing.

- Container is an axis-aligned rectangular region such that
- either it contains one large item.
- or items are packed inside the containers either as a horizontal stack or vertical stack



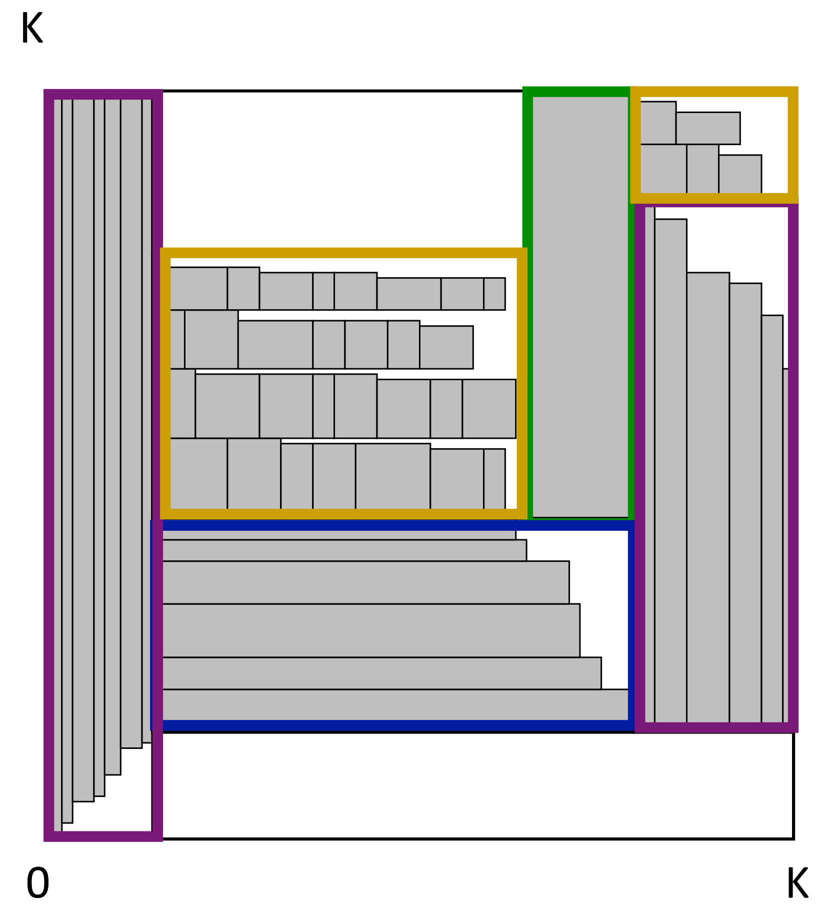
Previous approaches: container-based packing.

- Container is an axis-aligned rectangular region such that
- either it contains one large item.
- or items are packed inside the containers either as a horizontal stack or vertical stack

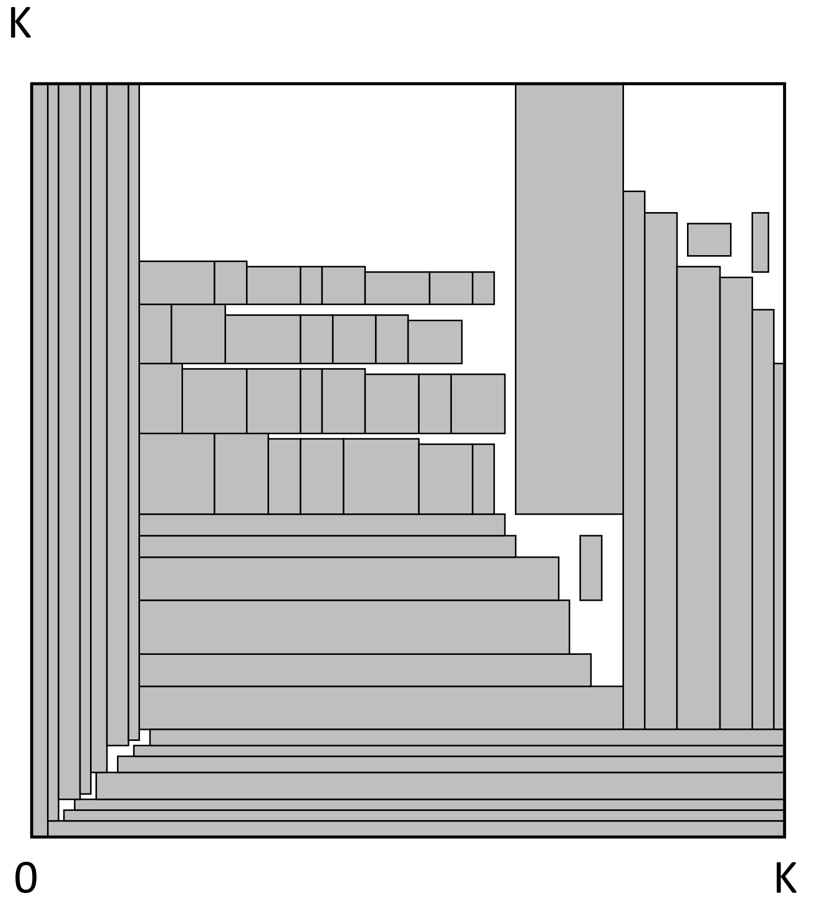


Previous approaches: container-based packing.

- Container is an axis-aligned rectangular region such that
- either it contains one large item.
- or items are packed inside the containers either as a horizontal stack or vertical stack
- or all items inside it are very small in both dimensions.

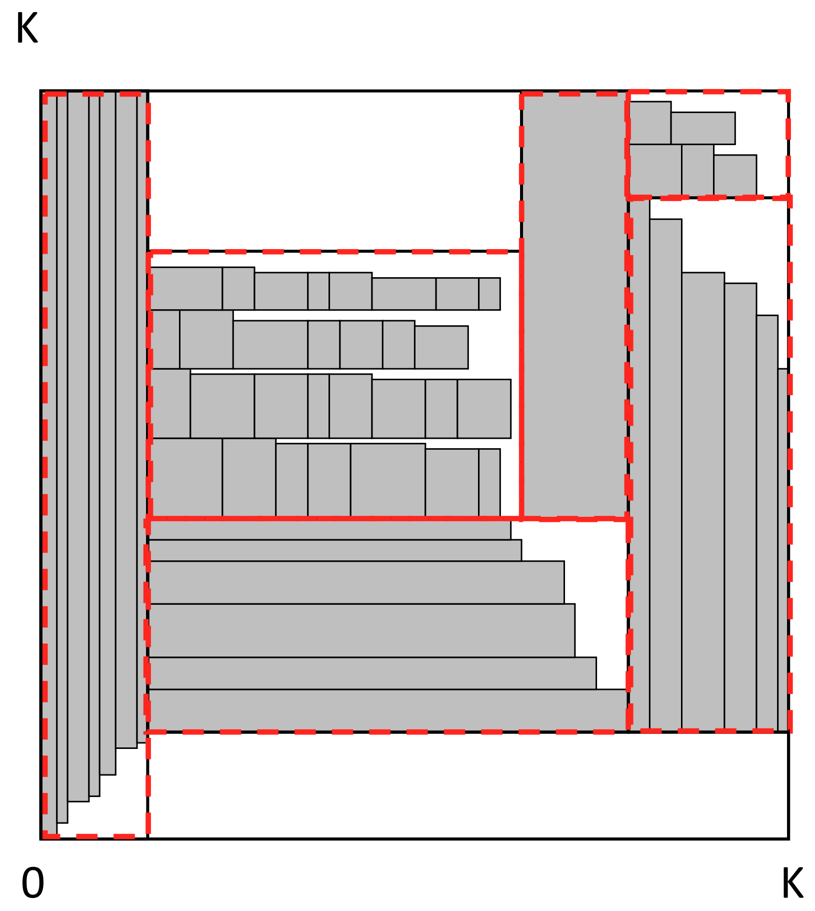


Previous approaches:
α-approximation using container-based packing.



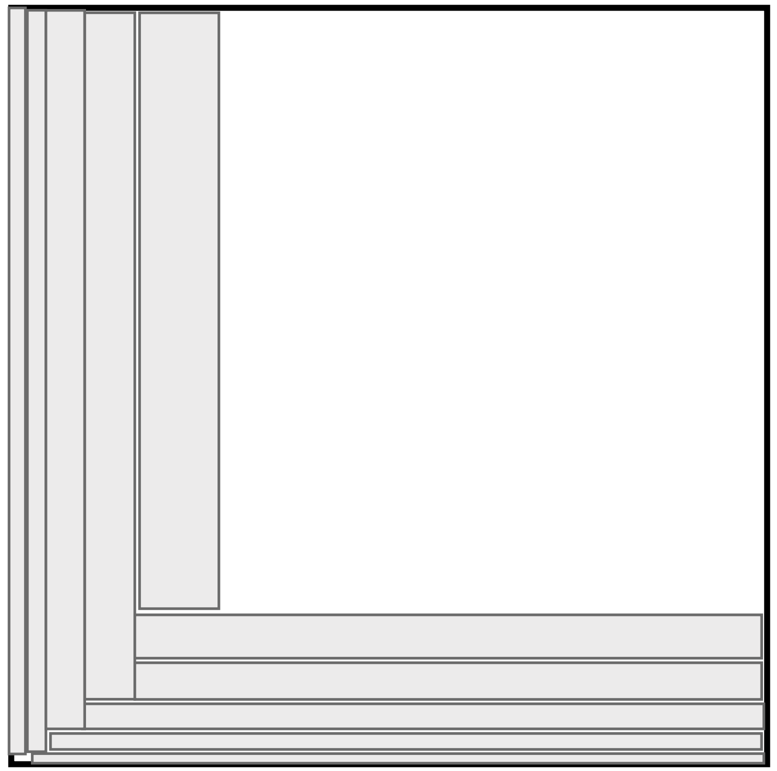
Previous approaches:
α-approximation using container-based packing.

- For any feasible packing, at least α fraction of the profit can be packed into O(1) number of containers.
- The sizes (and thus positions) of C containers can be found in $n^{O(C)}$ time.
- Containers can be packed using a Dynamic Program based PTAS for multiple-knapsack problem.



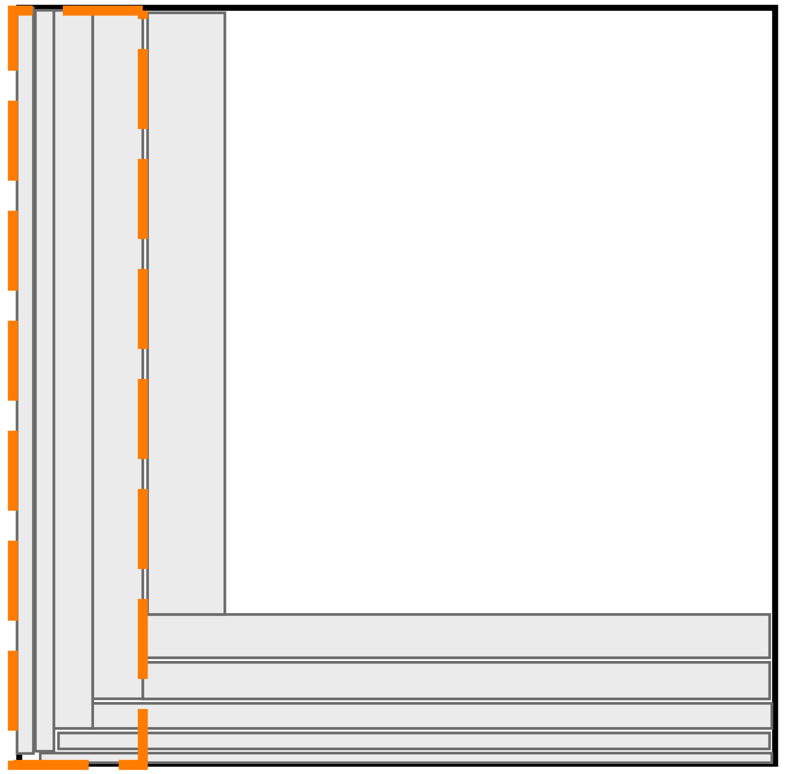
Bottleneck of 2-approximation:

- Consider the case when all items are *long*: have either *width > K/2 or height > K/2.*
- Trivial (2+ε)-approx. by taking either vertical or horizontal items and use 1-D knapsack PTAS.
- Vertical and horizontal items can interact in a very complicated way.
- Not clear how to beat 2-approximation, even in cardinality case, using container-based packing.



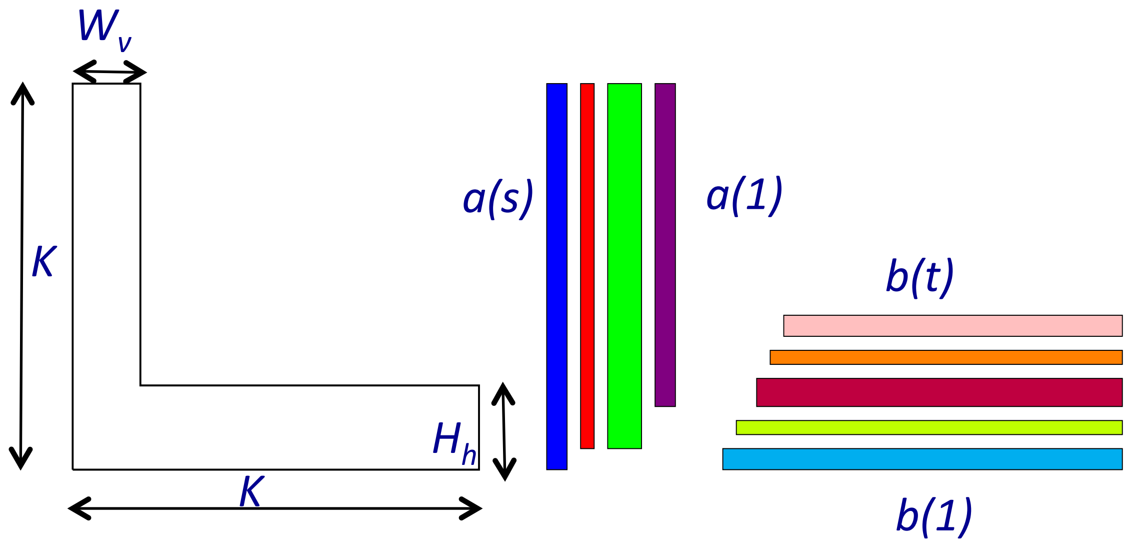
Bottleneck of 2-approximation:

- Consider the case when all items have either *width > K/2 or height > K/2.*
- Trivial (2+ε)-approx. by taking either vertical or horizontal items and use 1-D knapsack PTAS.
- Vertical and horizontal items can interact in a very complicated way.
- Not clear how to beat 2-approximation, even in cardinality case, using container-based packing.



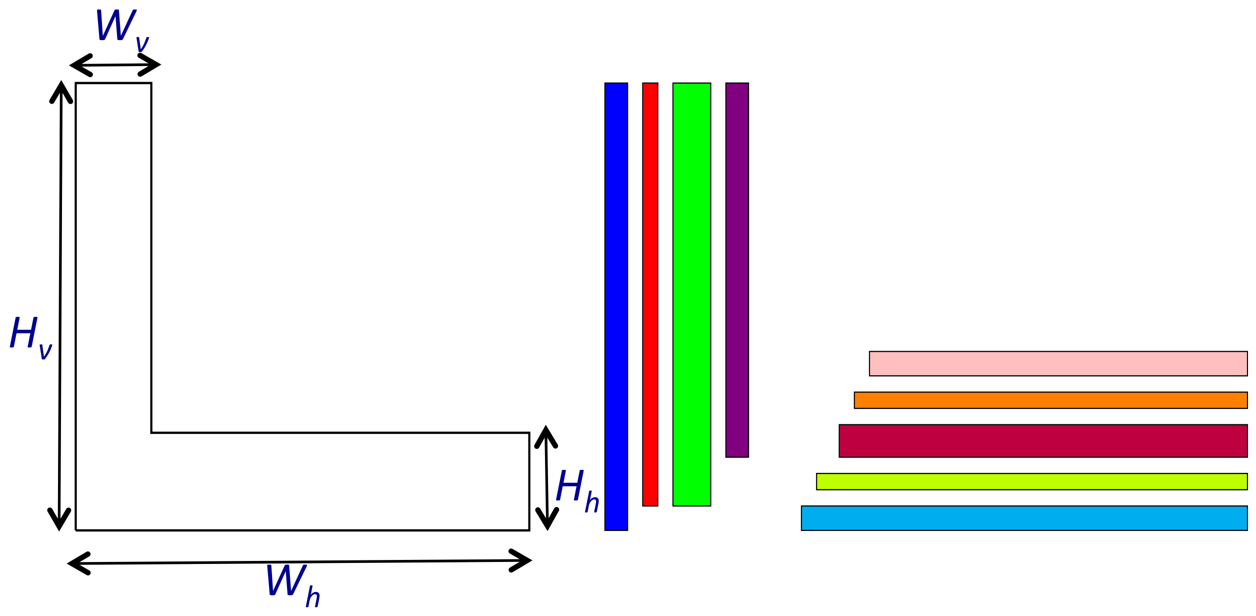
Bottleneck of 2-approximation:

- To handle this complex interaction, we go beyond containers!
- L-packing problem:
  - Given long items (*height or width > K/2)* and *an L-shaped region.*
  - Pack maximum profit subset of items inside the L-region.
- Previous best: (2+ε)-approx.

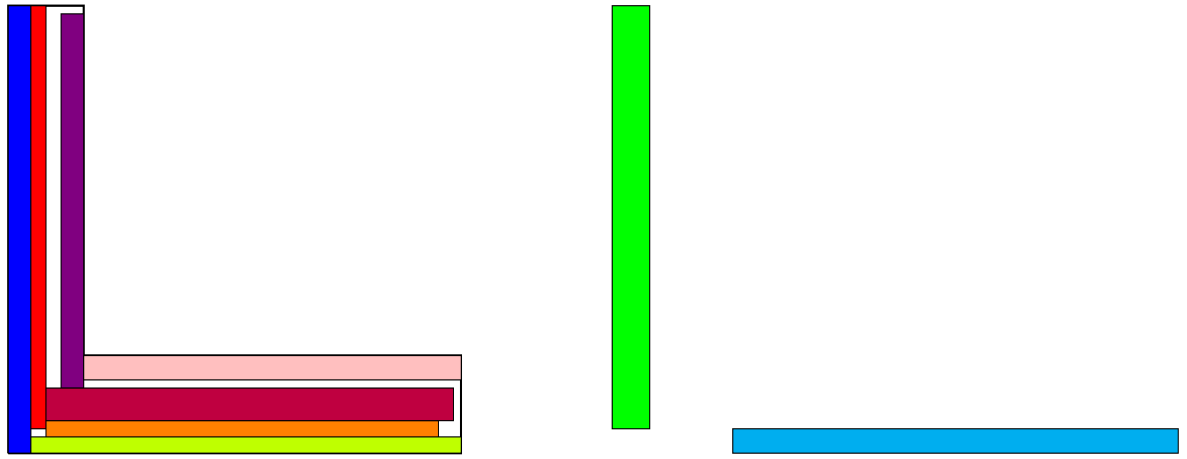


PTAS for L-packing

Pseudo-polytime algorithm for L-packing.

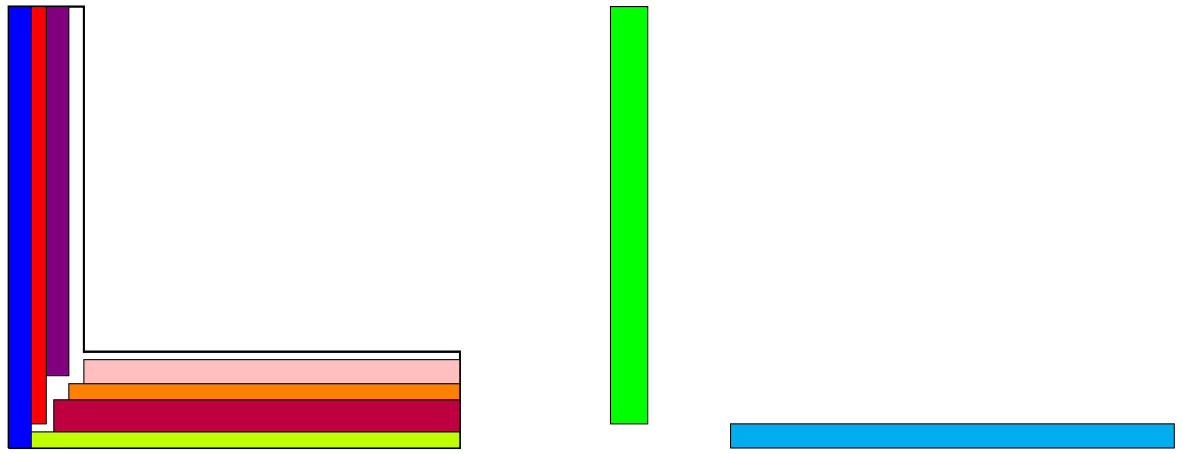


Pseudo-polytime algorithm for L-packing.



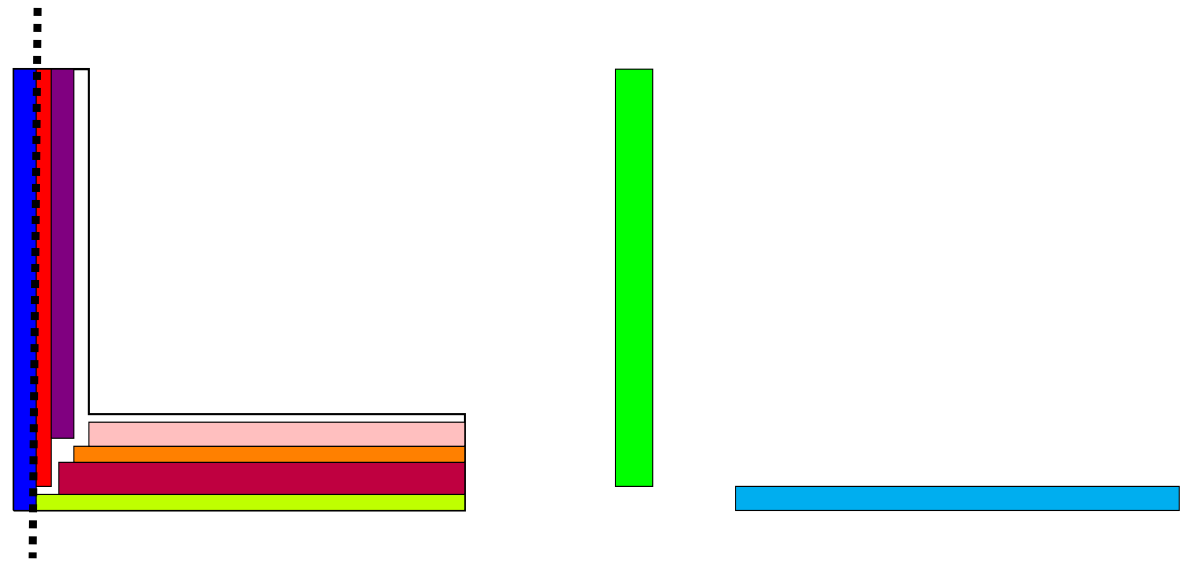
- All horizontal (resp. vertical) items are placed in the L-region according to nonincreasing width (resp. height) and touching right (resp. top) boundary.

Pseudo-polytime algorithm for L-packing.



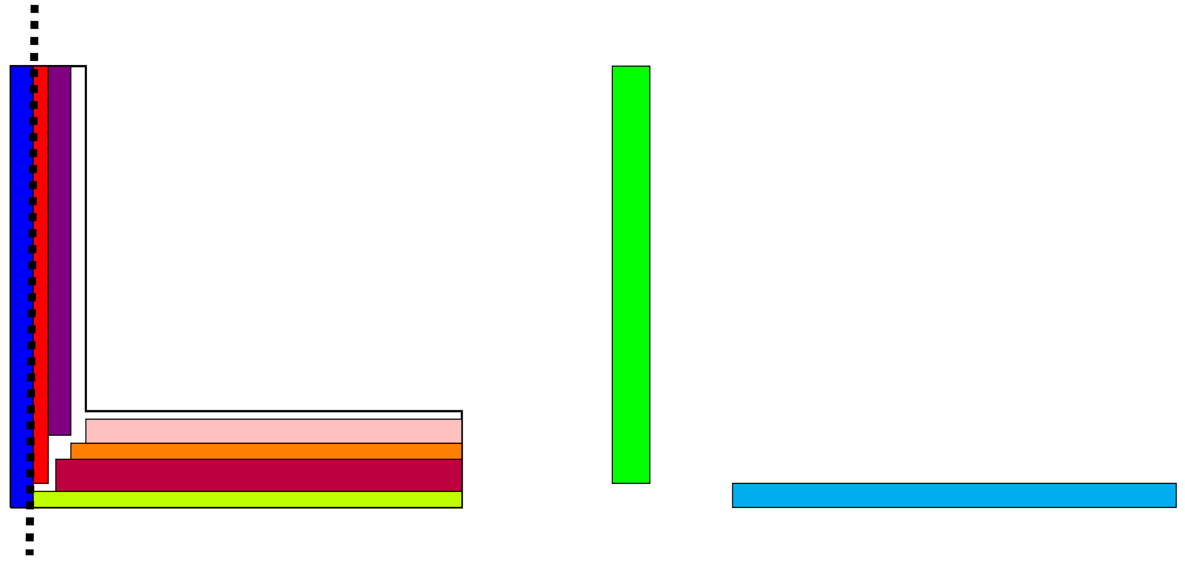
- All horizontal (resp. vertical) items are placed in the L-region according to nonincreasing width (resp. height) and touching right (resp. top) boundary.

Pseudo-polytime algorithm for L-packing.



- All horizontal (resp. vertical) items are placed in the L-region according to nonincreasing width (resp. height) and touching right (resp. top) boundary.
- Either a vertical (or hor.) cut exists that separates the tallest (or widest) item from a smaller L-region.

Pseudo-polytime algorithm for L-packing.

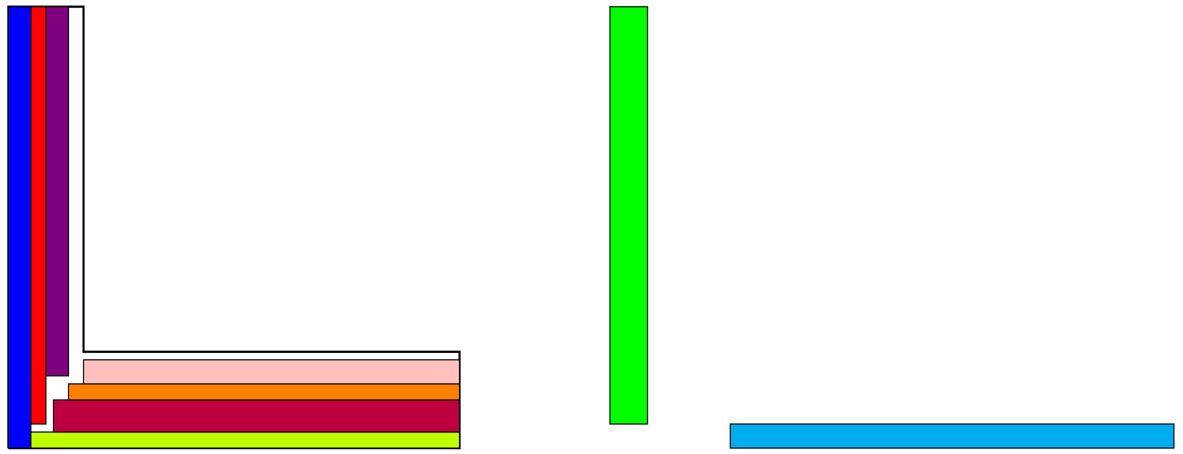


- Dynamic Program gives optimal solution in $(Kn)^{O(1)}$ time.

- All horizontal (resp. vertical) items are placed in the L-region according to nonincreasing width (resp. height) and touching right (resp. top) boundary.
- Either a vertical (or hor.) cut exists that separates the tallest (or widest) item from a smaller L-region.

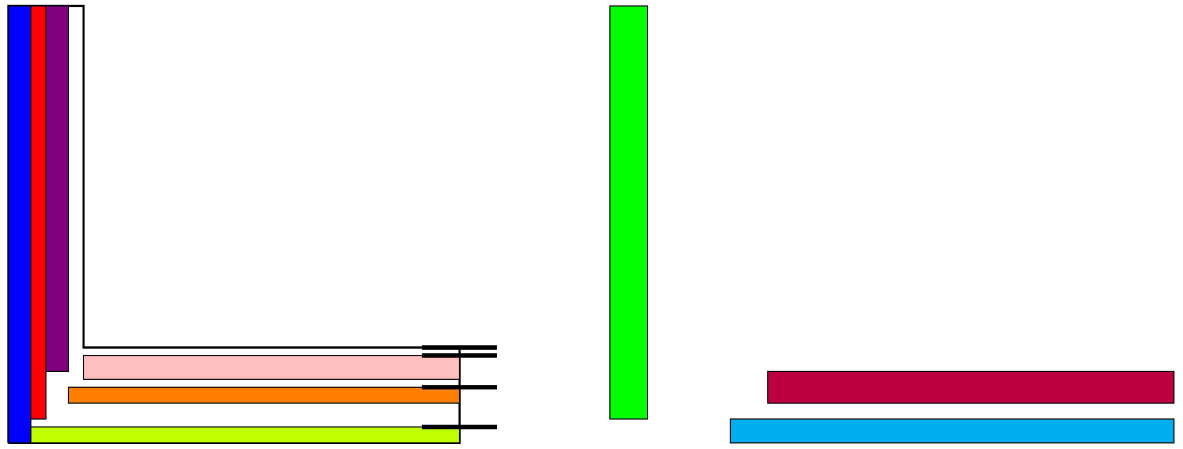
PTAS for L-packing.

- Structural lemma:
  Modify packing of horizontal (resp. vertical) items in L-packing s.t.
  - items of profit $\leq \varepsilon p(OPT)$ is removed,
  - remaining items are shifted down (resp. left) or stays same,
  - the top (resp. right) coordinates of items takes only $n^{O(1)}$ values.



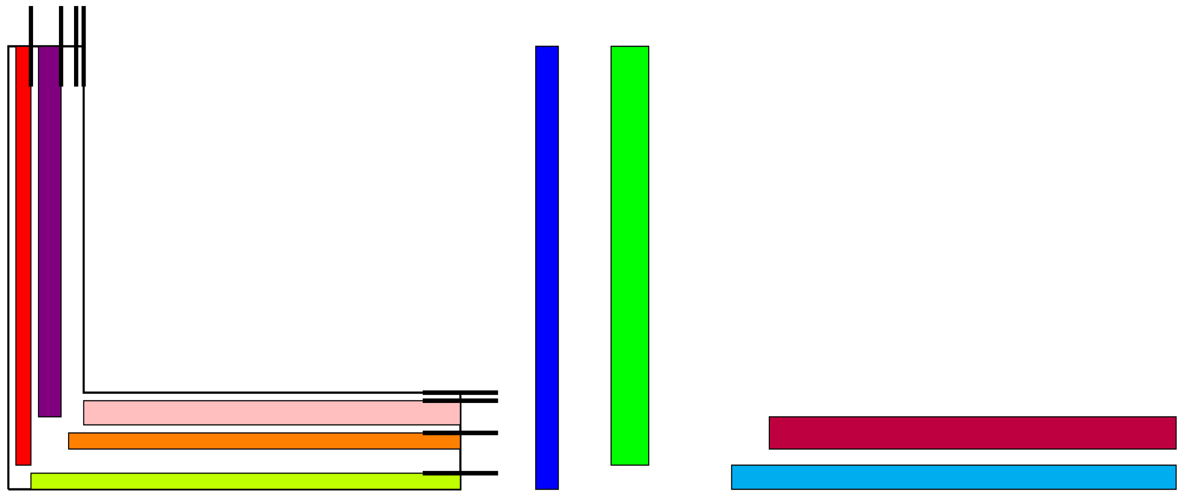
PTAS for L-packing.

- Structural lemma:
  Modify packing of horizontal (resp. vertical) items in L-packing s.t.
  - items of profit ≤εp(OPT) is removed,
  - remaining items are shifted down (resp. left) or stays same,
  - the top (resp. right) coordinates of items takes only $n^{O(1)}$ values.



PTAS for L-packing.

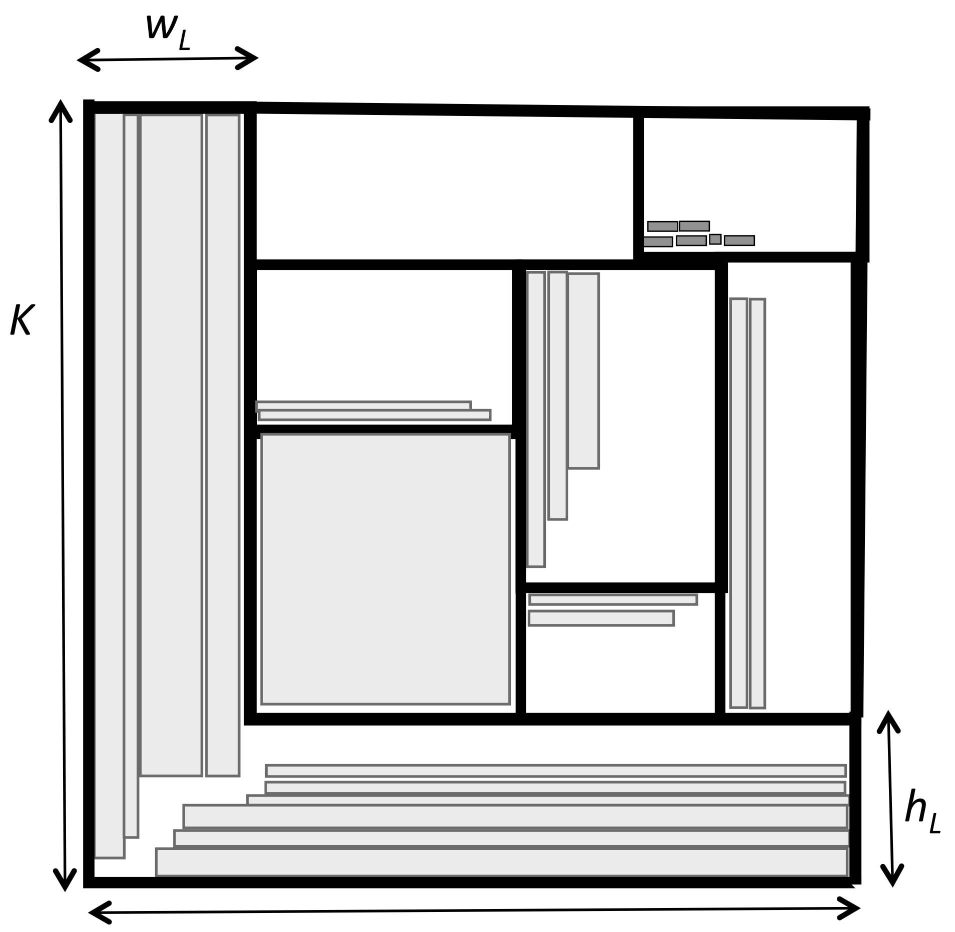
- Structural lemma:
  Modify packing of horizontal (resp. vertical) items in L-packing s.t.
  - items of profit ≤εp(OPT) is removed,
  - remaining items are shifted down (resp. left) or stays same,
  - the top (resp. right) coordinates of items takes only $n^{O(1)}$ values.



Cardinality case without rotations: ≈16/9-approximation

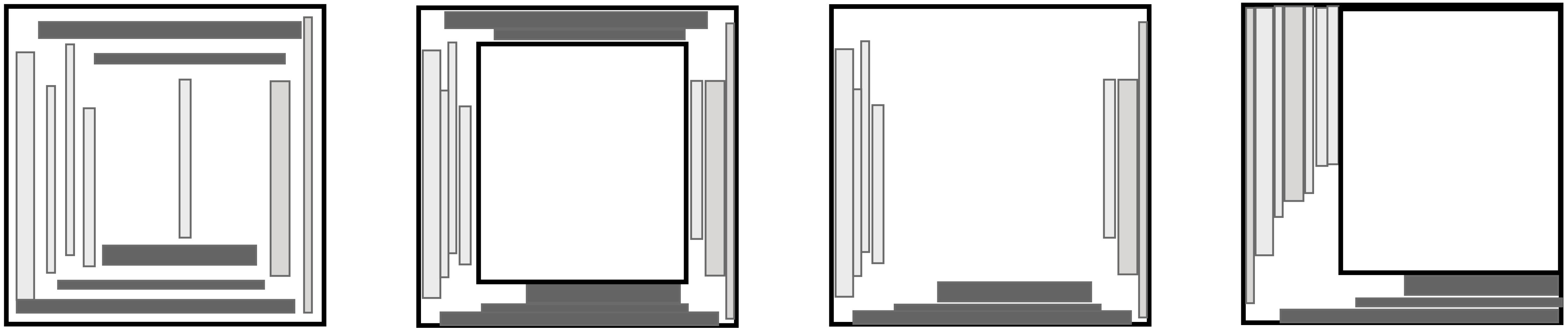
- Long items: longer side > K/2.
- Short items: both sides ≤ K/2.
- Packing 1 : Packing of L-region
  $\approx (\frac{3}{4}\, OPT_{long})$
- Packing 2 : Packing of O(1) containers
  $\approx (\frac{1}{2}\, OPT_{long} + \frac{3}{4}\, OPT_{short})$
- Best packing:

$$(\tfrac{3}{4} OPT_{long})\tfrac{1}{4} + (\tfrac{1}{2} OPT_{long} + \tfrac{3}{4} OPT_{short})\tfrac{3}{4}$$
$$\geq (OPT_{long} + OPT_{short})9/16 \geq \frac{9}{16}\ OPT.$$



"O my Precious!"
L of the ring

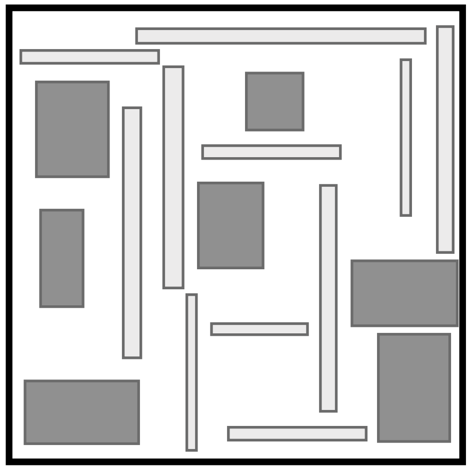
Packing 1: ≈ (¾ OPT_{long}), "L" of the ring!



- Create stacks from rectangles from OPT_{long} to form a ring.
- Remove least profitable stack in the ring.
- Rearrange remaining long items into an L-packing.
- Use PTAS for L-packing to get profit at least ≈ ¾ OPT_{long} .

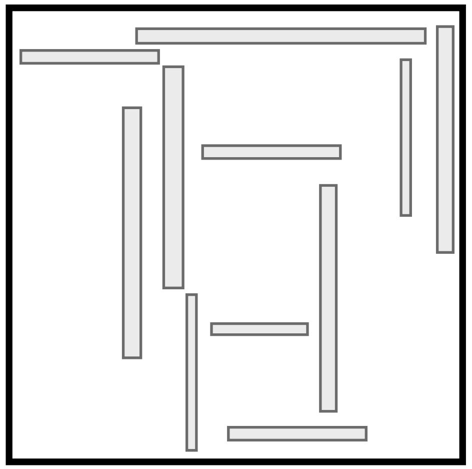
Packing 2 ≈ (½ OPT_{long}+¾ OPT_{short})

- If $OPT < 1/\varepsilon^3$, solve optimally by brute-force.
- So, consider $OPT \geq 1/\varepsilon^3$.
- Define Large items have both sides ≥ εK.
- There are $\leq 1/\varepsilon^2 \leq \varepsilon$ OPT large items.
- We loose small profit by discarding large items.



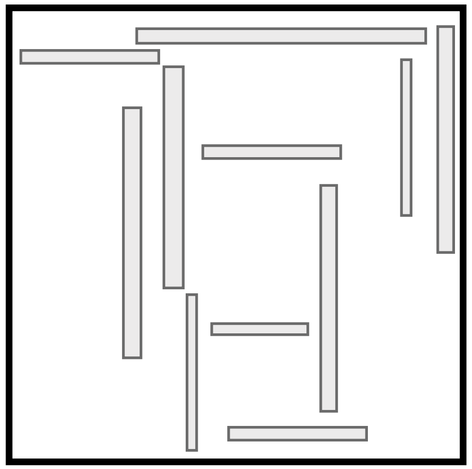
Packing 2 ≈ (½ OPT_{long}+¾ $\text{OPT}_{\text{short}}$)

- If $\text{OPT} < 1/\varepsilon^3$, solve optimally by brute-force.
- So, consider $\text{OPT} \geq 1/\varepsilon^3$.
- Define Large items have both sides ≥ εK.
- There are $\leq 1/\varepsilon^2 \leq \varepsilon$ OPT large items.
- We loose small profit by discarding large items.



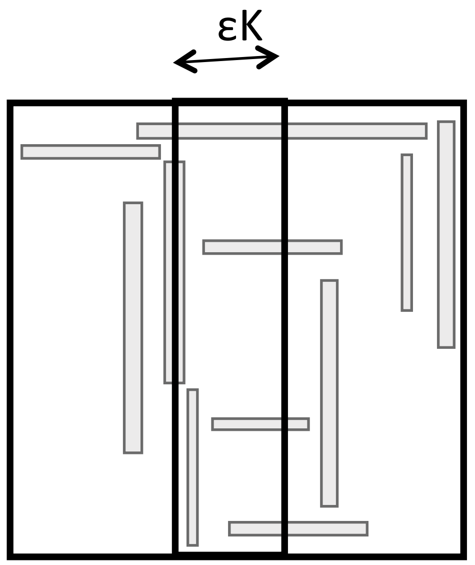
Packing 2 ≈ (½ OPT_{long}+¾ OPT_{short})

- If $OPT < 1/\varepsilon^3$, solve optimally by brute-force.
- So, consider $OPT \geq 1/\varepsilon^3$.
- Define Large items have both sides ≥ εK.
- There are $\leq 1/\varepsilon^2 \leq \varepsilon$ OPT large items.
- We loose small profit by discarding large items.
- So all remaining items have either height or width < εK.



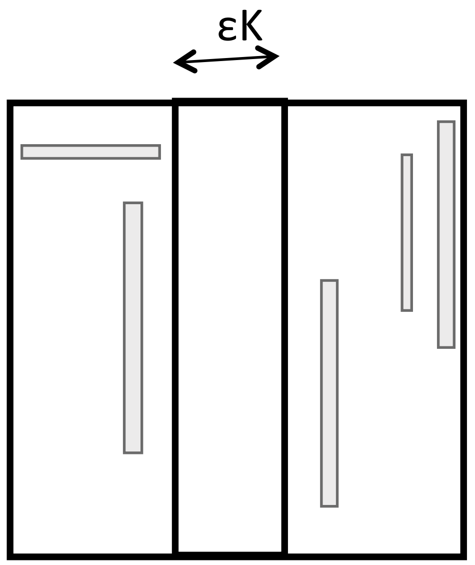
Packing 2 ≈ (½ OPT_{long}+¾ $\text{OPT}_{\text{short}}$)

- Remove all items intersecting a random vertical (or horizontal) strip of width (or height) εK.
- Prob. a horizontal (vertical) long item is removed ≤ ½ . 1 + ½ . O(ε).
- Prob. a horizontal (vertical) short item is removed ≤ ½ . ½ + ½ . O(ε).
- Remaining items ≈ (½ OPT_{long}+¾ $\text{OPT}_{\text{short}}$).



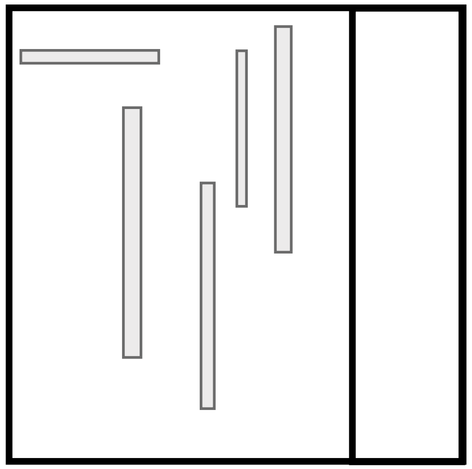
Packing 2 ≈ (½ OPT_{long}+¾ $\text{OPT}_{\text{short}}$)

- Remove all items intersecting a random vertical (or horizontal) strip of width (or height) εK.
- Prob. a horizontal (vertical) long item is removed ≤ ½ . 1 + ½ . O(ε).
- Prob. a horizontal (vertical) short item is removed ≤ ½ . ½ + ½ . O(ε).
- Remaining items ≈ (½ OPT_{long}+¾ $\text{OPT}_{\text{short}}$).
- We can pack remaining items into O(1) containers using resource-augmentation.



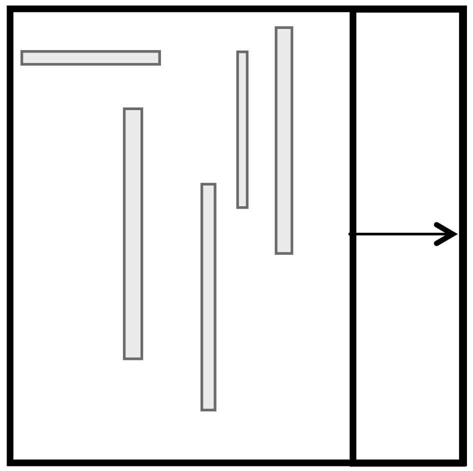
Packing 2 ≈ (½ OPT_{long}+¾ OPT_{short})

- Remove all items intersecting a random vertical (or horizontal) strip of width (or height) εK.
- Prob. a horizontal (vertical) long item is removed ≤ ½ . 1 + ½ . O(ε).
- Prob. a horizontal (vertical) short item is removed ≤ ½ . ½ + ½ . O(ε).
- Remaining items ≈ (½ OPT_{long}+¾ OPT_{short}).
- We can pack remaining items into O(1) containers using resource-augmentation.



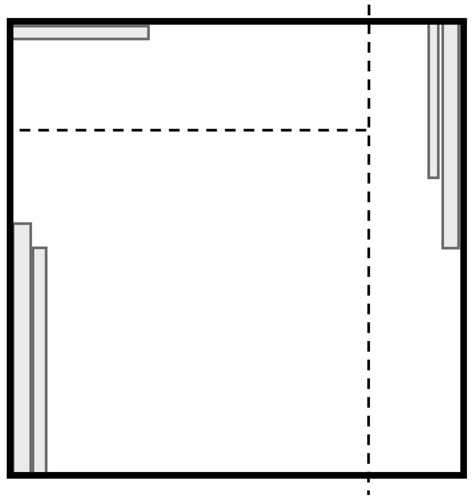
Packing 2 ≈ (½ OPT_{long}+¾ OPT_{short})

- Remove all items intersecting a random vertical (or horizontal) strip of width (or height) εK.
- Prob. a horizontal (vertical) long item is removed ≤ ½ . 1 + ½ . O(ε).
- Prob. a horizontal (vertical) short item is removed ≤ ½ . ½ + ½ . O(ε).
- Remaining items ≈ (½ OPT_{long}+¾ OPT_{short}).
- We can pack remaining items into O(1) containers using resource-augmentation.



Packing 2 ≈ (½ $\mathrm{OPT}_{\mathrm{long}}$+¾ $\mathrm{OPT}_{\mathrm{short}}$)

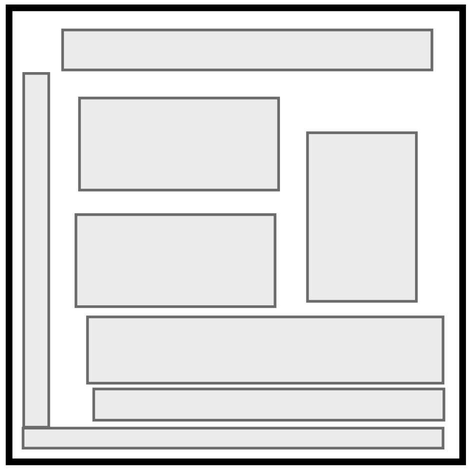
- Remove all items intersecting a random vertical (or horizontal) strip of width (or height) εK.
- Prob. a horizontal (vertical) long item is removed ≤ ½ . 1 + ½ . O(ε).
- Prob. a horizontal (vertical) short item is removed ≤ ½ . ½ + ½ . O(ε).
- Remaining items ≈ (½ $\mathrm{OPT}_{\mathrm{long}}$+¾ $\mathrm{OPT}_{\mathrm{short}}$).
- We can pack remaining items into O(1) containers using resource-augmentation.



Cardinality case with Rotations

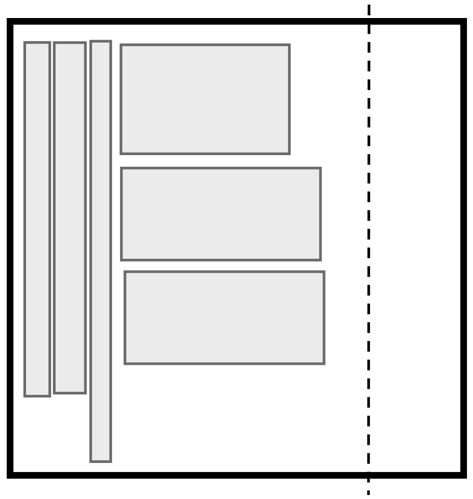
With rotations: a simple 3/2-approximation.

- Resource Contraction Lemma: If rectangles *M* can be packed in *KxK* bin and $|M| \geq 1/\varepsilon^3$, then at least $2|M|/3$ rectangles can be packed into $K \times (1-O(\varepsilon))K$ bin.



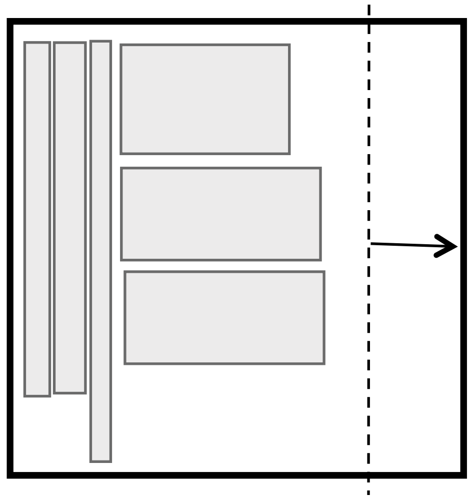
With rotations: a simple 3/2-approximation.

- Resource Contraction Lemma:
  If rectangles M can be packed in KxK bin and $|M| \geq 1/\varepsilon^3$, then at least $2|M|/3$ rectangles can be packed into Kx$(1-O(\varepsilon))K$ bin.



With rotations: a simple 3/2-approximation.

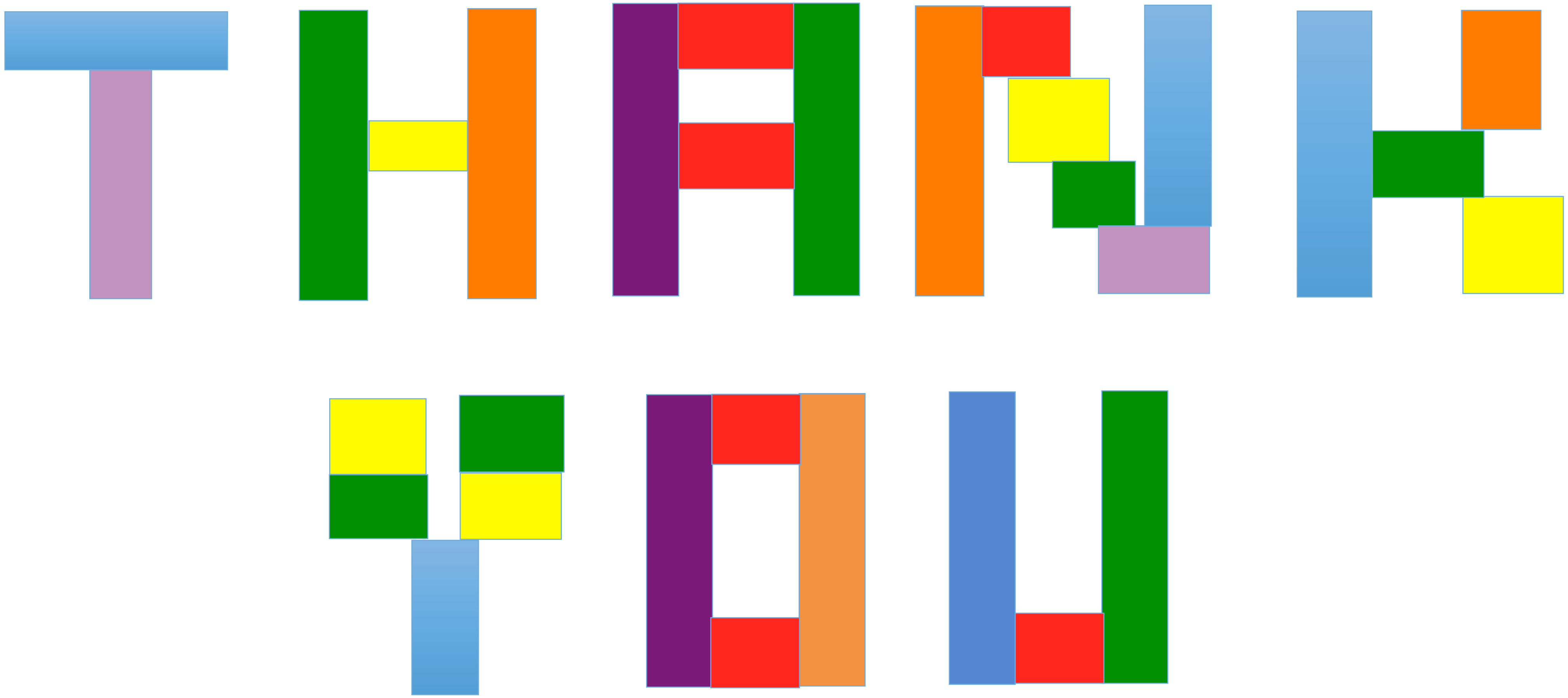
- Resource Contraction Lemma:
  If rectangles M can be packed in KxK bin and $|M|\geq 1/\varepsilon^3$, then at least $2|M|/3$ rectangles can be packed into Kx$(1\text{-}O(\varepsilon))K$ bin.
- Using resource augmentation, this shows existence of a container packing that gives 3/2-approximation.



Open Problems.

- Find a PTAS! Even in the cardinality case.
- Understand natural generalizations of L-packing.
  - Is there PTAS for ring instance?
  - Is there PTAS for L-packing with rotations?
  - Is there PTAS for O(1) instances of L-packing?

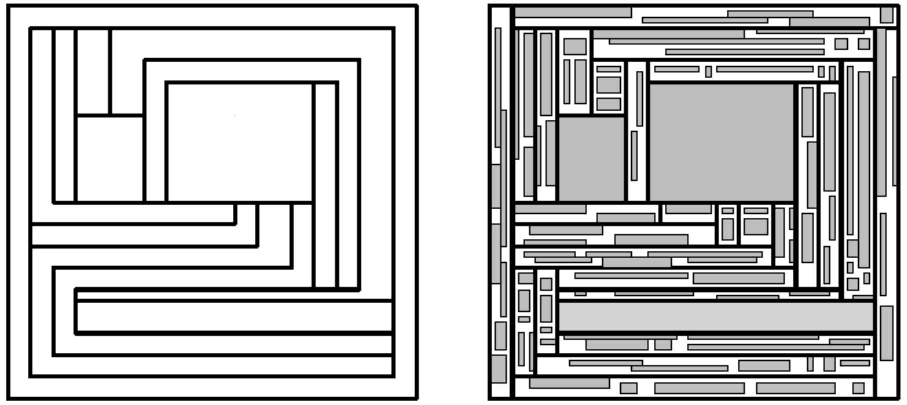
- More related literature and open problems:
  Approximation and Online Algorithms for Multidimensional Bin Packing: A Survey, Christensen-Khan-Pokutta-Tetali, Computer Science Review'17.


THANK
YOU

Additional Slides

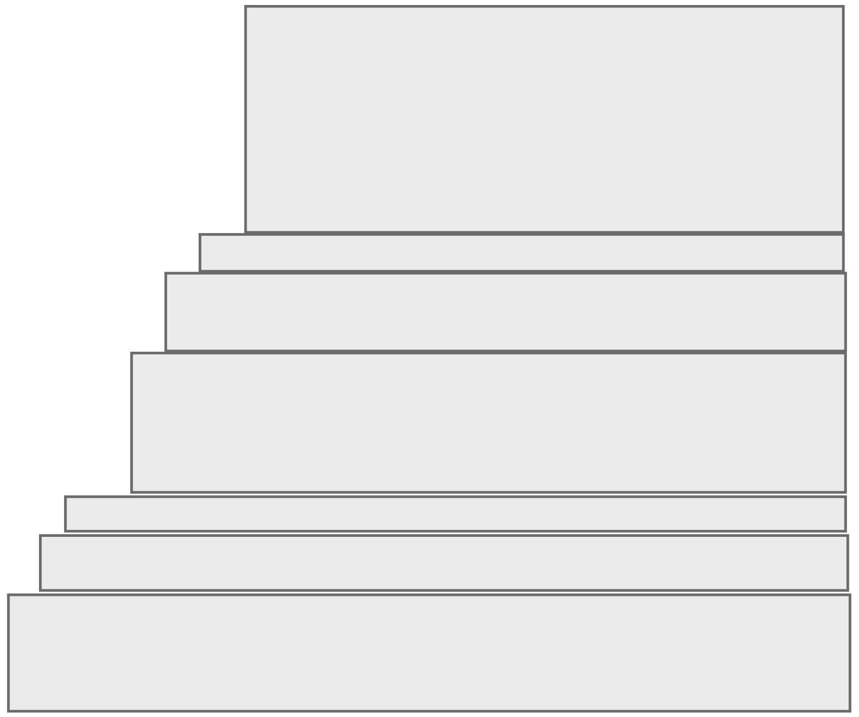
Extension to the weighted case.

- Few items can contribute to the majority of the profit.
- We can no more discard large items.
- Involved use of corridor-partitioning. [Adamaszek,Wiese; SODA'15, FOCS'13]
  - Any feasible packing can be partitioned into O(1) corridors (rectilinear polygons) defined by O(1) number of line segments and intersecting only rectangles of profit $\leq \varepsilon p(OPT)$.
  - A large fraction of the profit can be retained by containers constructed from corridors.



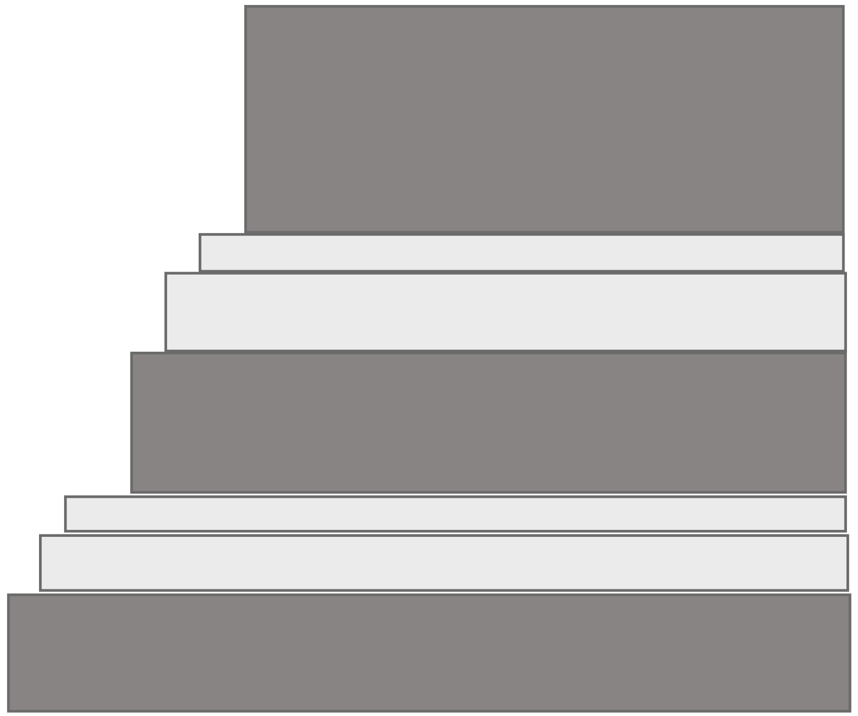
PTAS for L-packing.

- Consider horizontal items H.



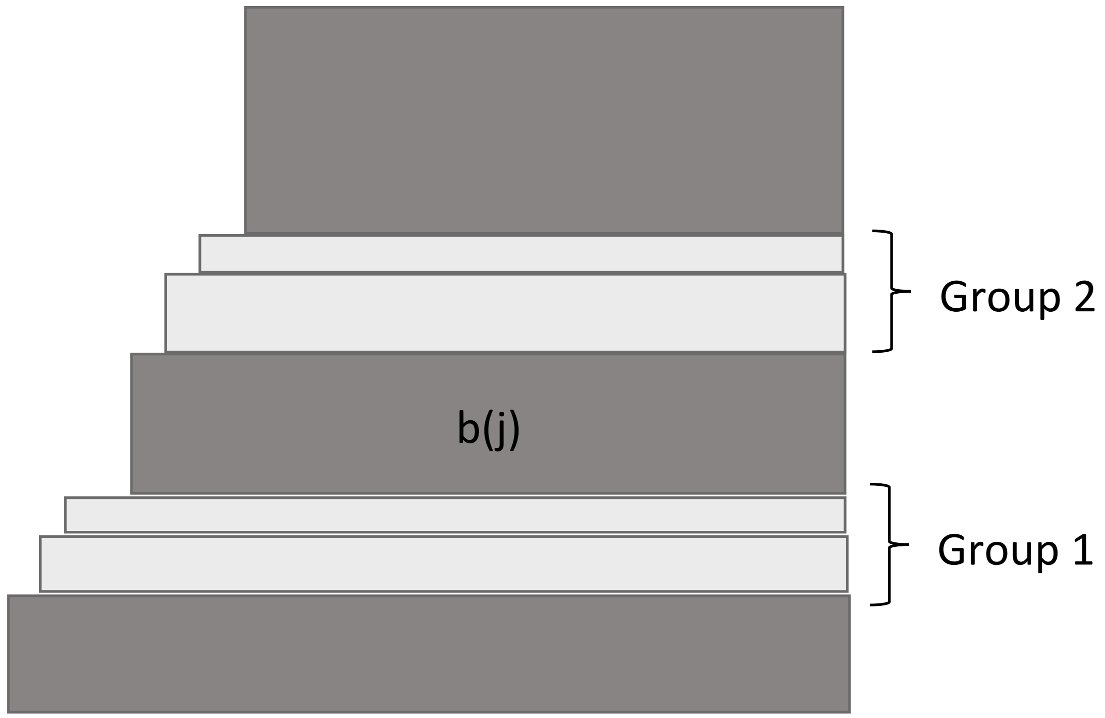
PTAS for L-packing.

- Consider horizontal items H.
- Create G, a growing subsequence of items where heights increase.



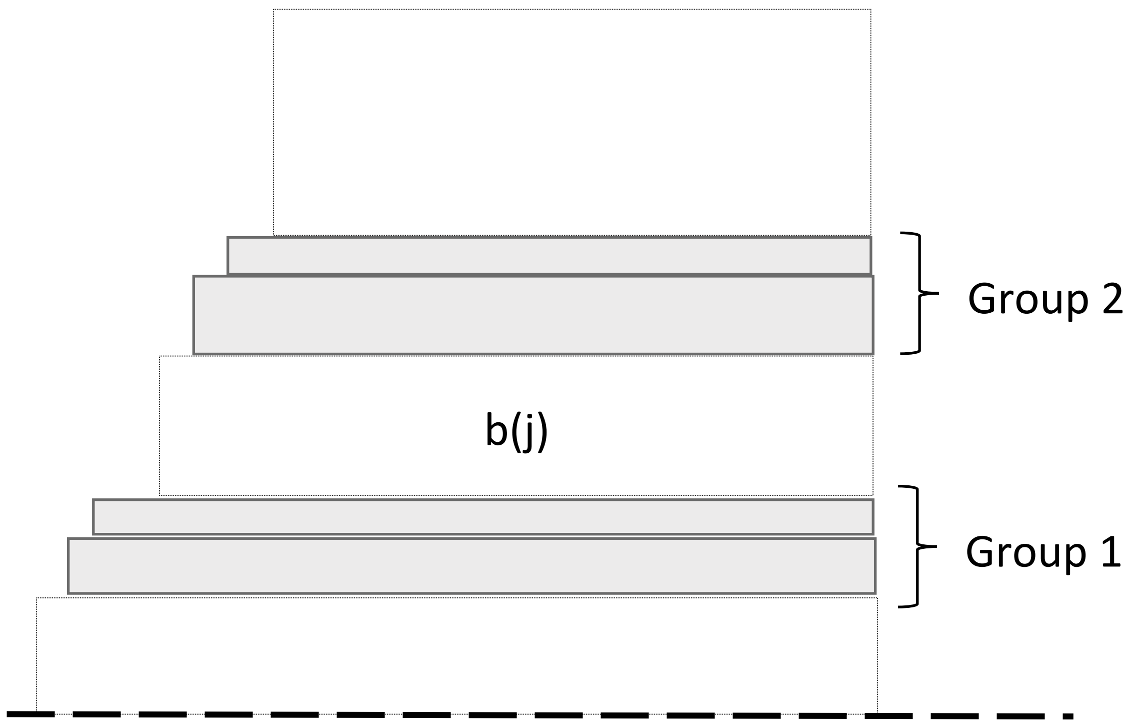
PTAS for L-packing.

- Consider horizontal items H.
- Create G, a growing subsequence of items where heights increase.
- If p(G)≤εp(OPT),
  - remove G.
  - This creates several groups.
  - shift items within each group.



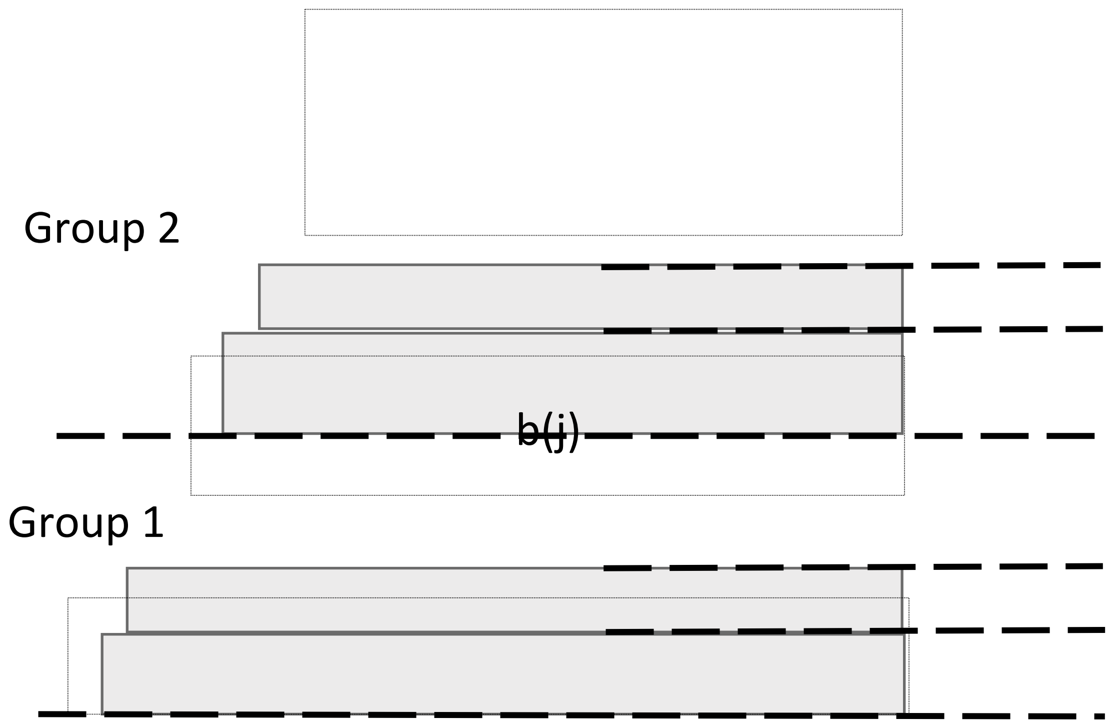
PTAS for L-packing.

- Consider horizontal items H.
- Create G, a growing subsequence of items where heights increase.
- If p(G)≤εp(OPT),
  - remove G.
  - This creates several groups.
  - shift items within each group.



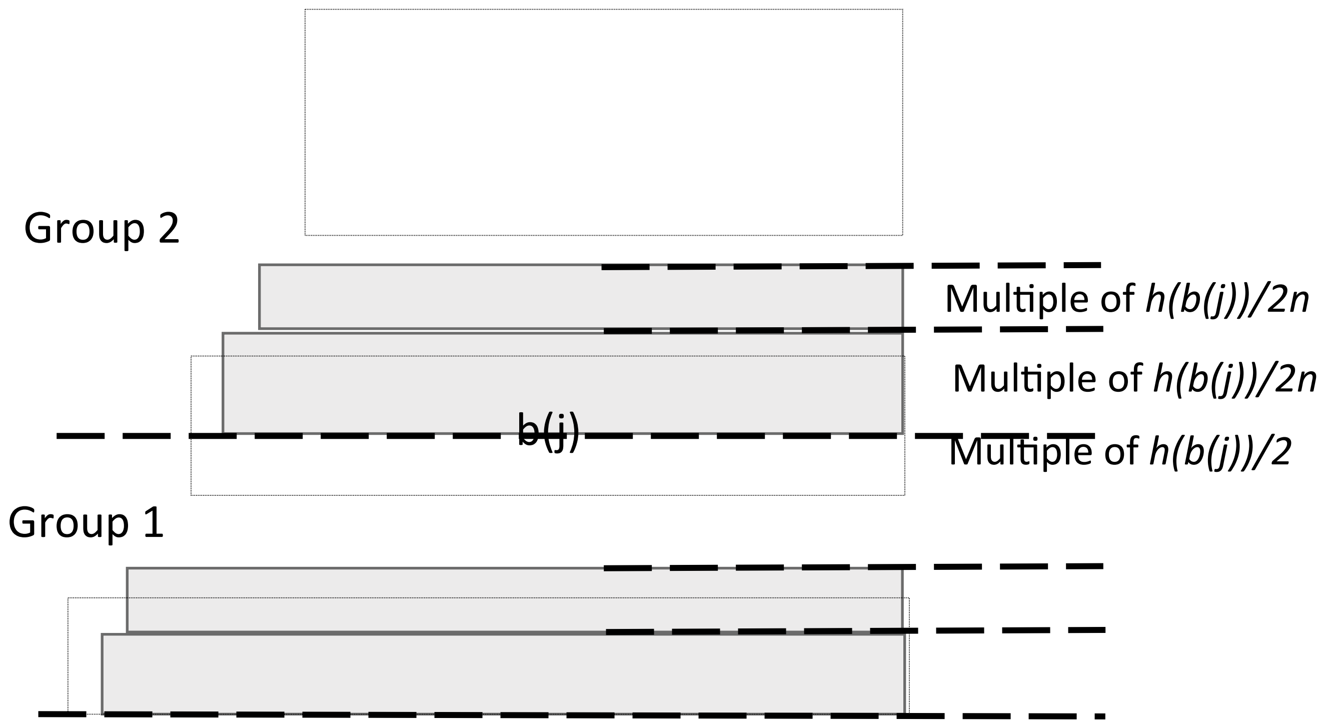
PTAS for L-packing.

- Consider horizontal items H.
- Create G, a growing subsequence of items where heights increase.
- If p(G)≤εp(OPT),
  - remove G.
  - This creates several groups.
  - shift items within each group.



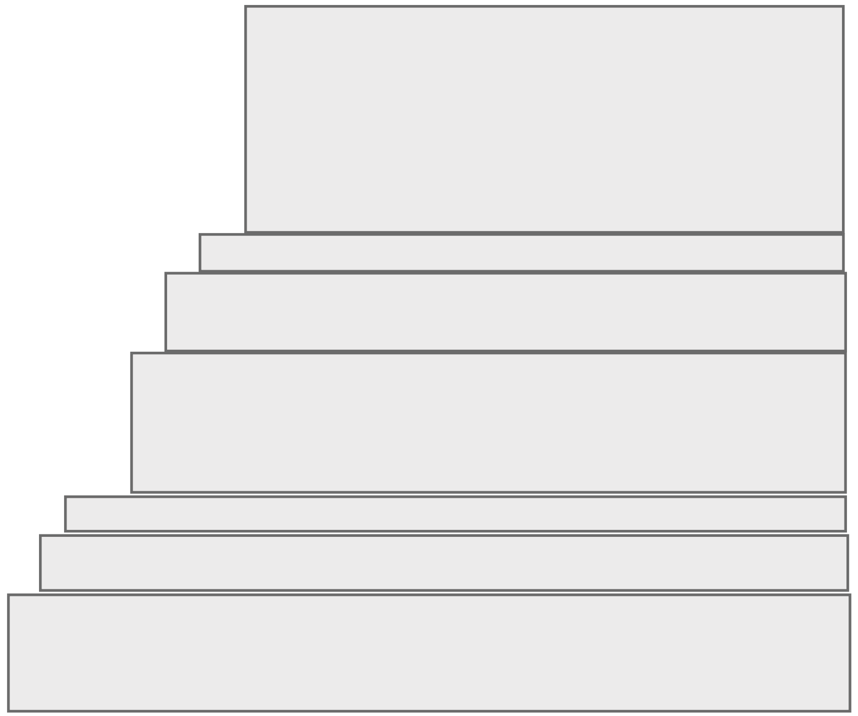
PTAS for L-packing.

- Consider horizontal items H.
- Create G, a growing subsequence of items where heights increase.
- If p(G)≤εp(OPT),
  - remove G.
  - This creates several groups.
  - shift items within each group.

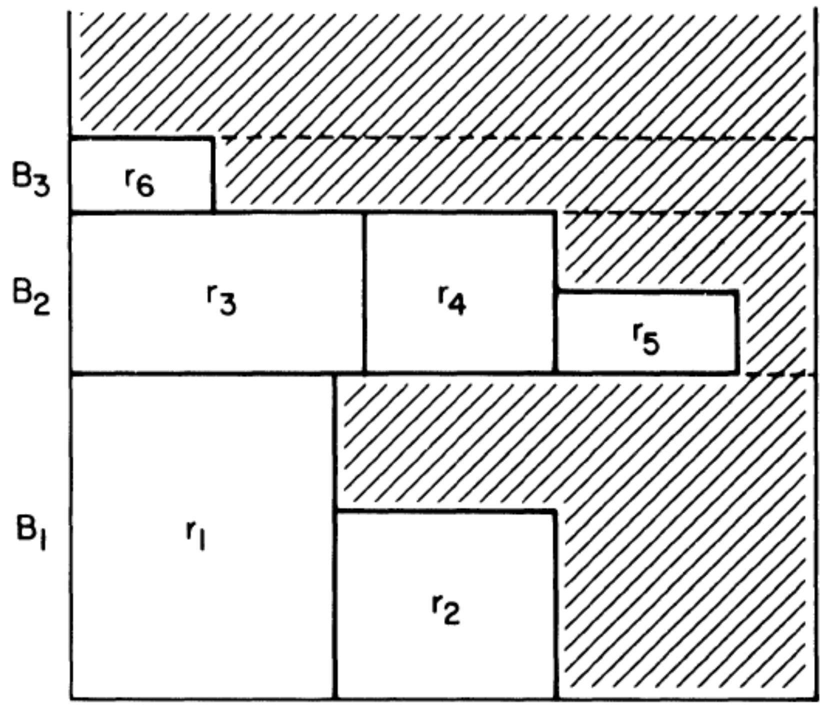


PTAS for L-packing.

- Consider horizontal items H.
- Create G, a growing subsequence of items where heights increase.
- If p(G)≤εp(OPT),
  - remove G.
  - This creates several groups.
  - shift items within each group.
- Otherwise if p(G)>εp(OPT), use recursion within the groups.
  − much involved!



Next Fit Decreasing Height(NFDH)



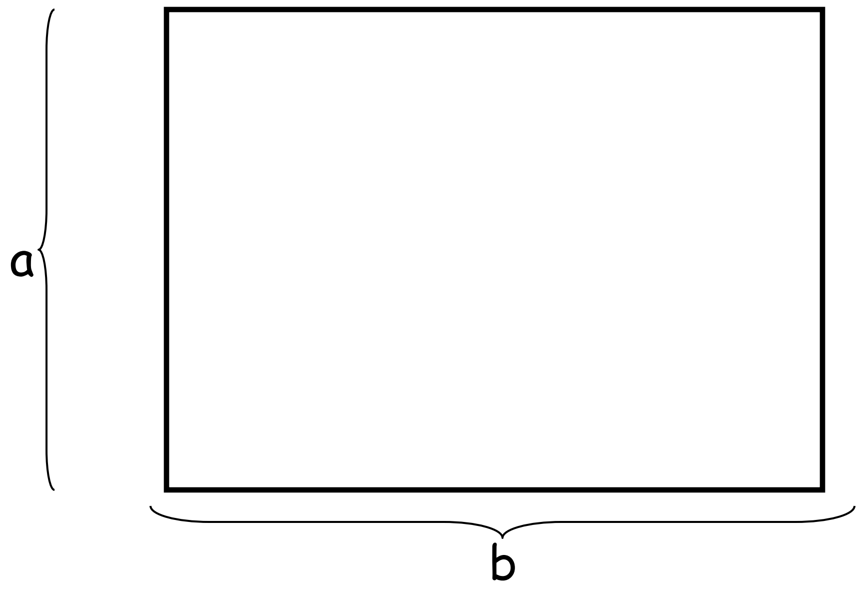
- Considered items in a non-increasing order of height and greedily packs items into shelves.
- Shelf is a row of items having their bases on a line that is either the base of the bin or the line drawn at the top of the highest item packed in the shelf below.
- items are packed left-justified starting 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.

Shelf Packing

Given a rectangular region of size a £ b

Goal: Pack squares of length · s

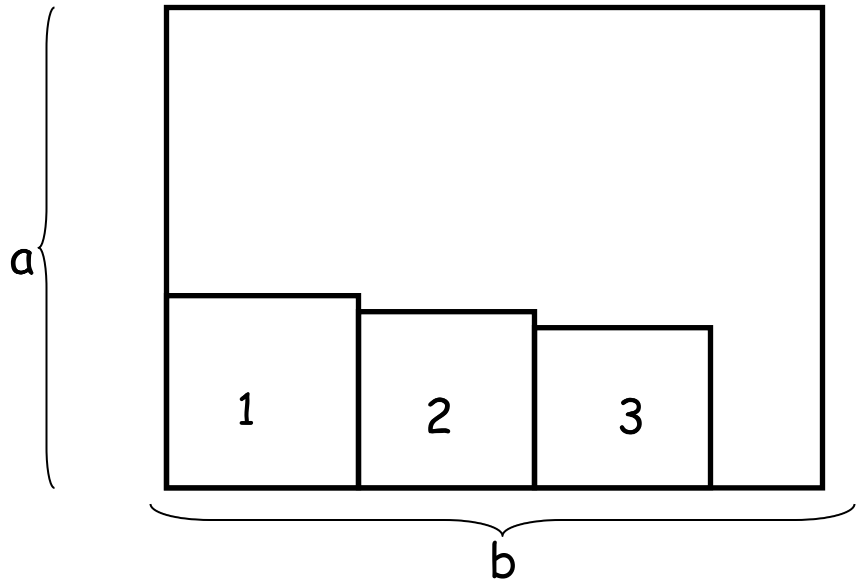


Shelf Packing

Given a rectangular region of size a x b

Goal: Pack squares of length ≤ s

Algorithm: Decreasing size shelf packing.



Take squares in decreasing size

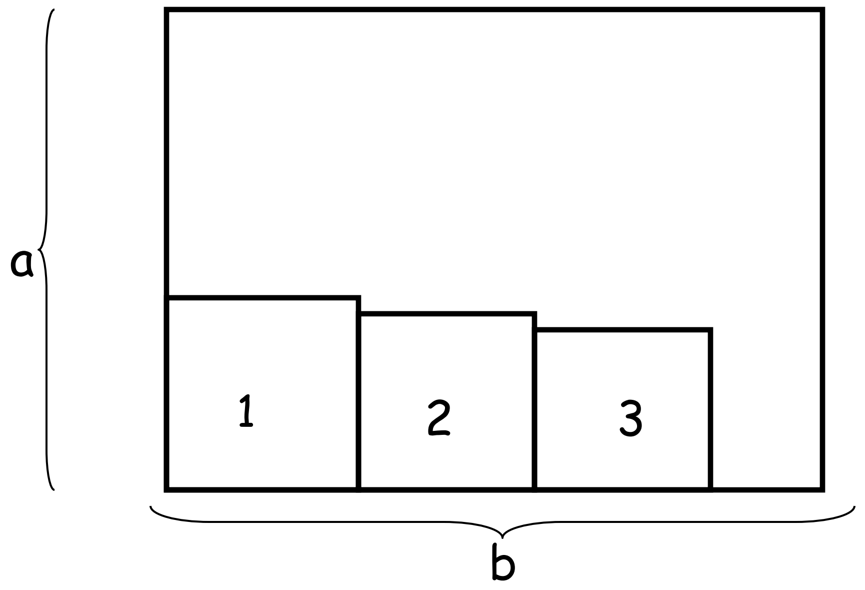
- Place sequentially

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

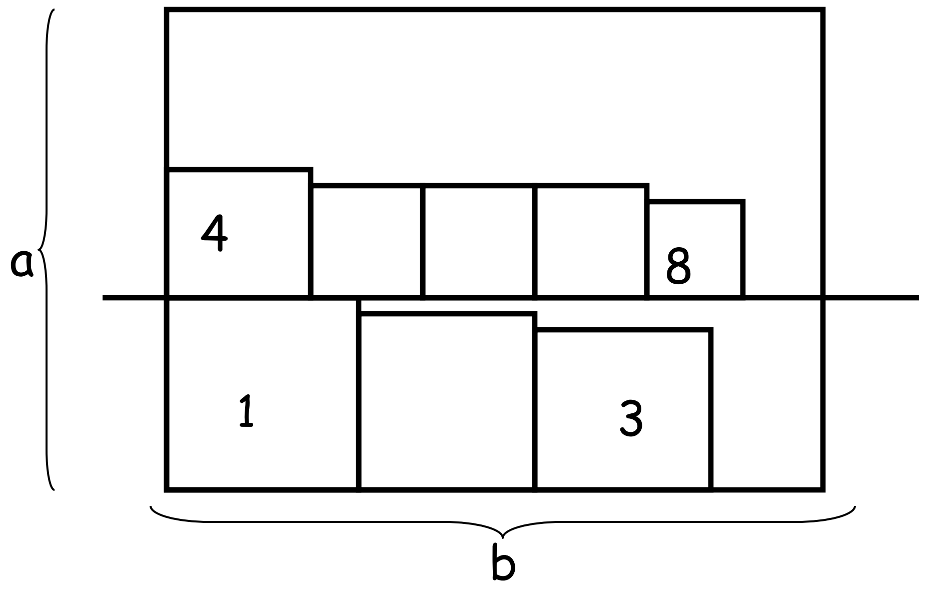
- Place sequentially
- If next does not fit, open a new shelf

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

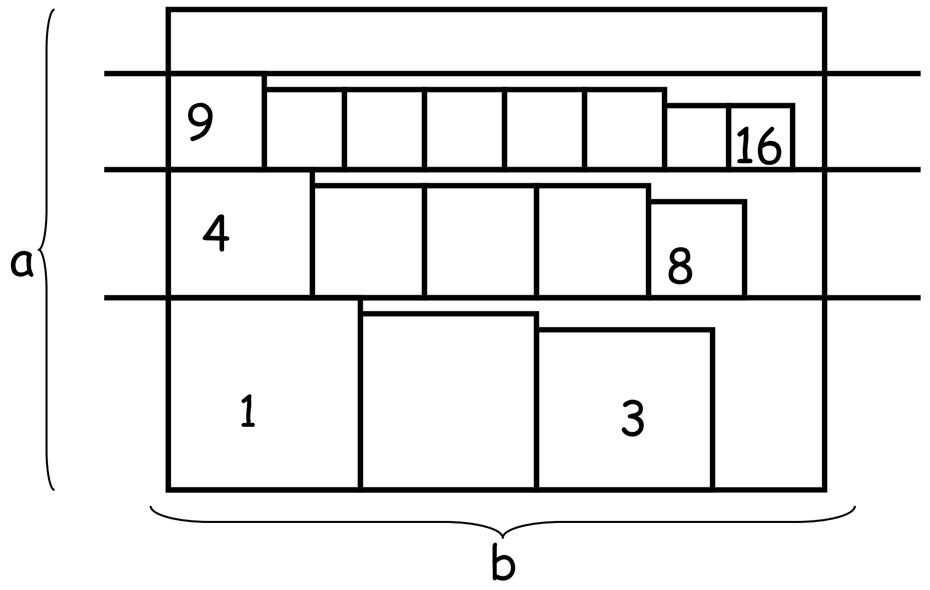
- Place sequentially
- If next does not fit, open a new shelf

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

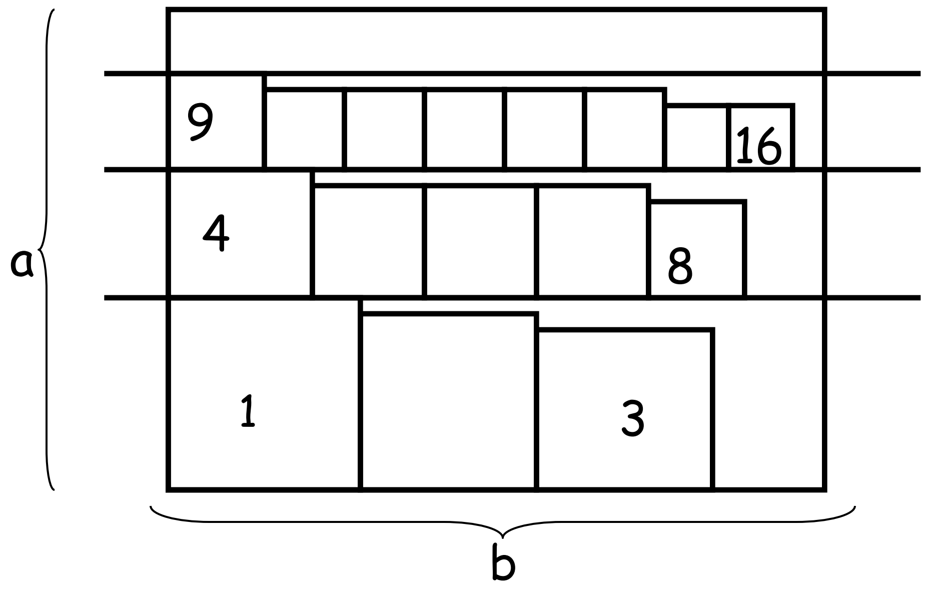
- Place sequentially
- If next does not fit, open a new shelf

Shelf Packing

Given a rectangular region of size a £ b

Goal: Pack squares of length · s

Algorithm: Decreasing size shelf packing.



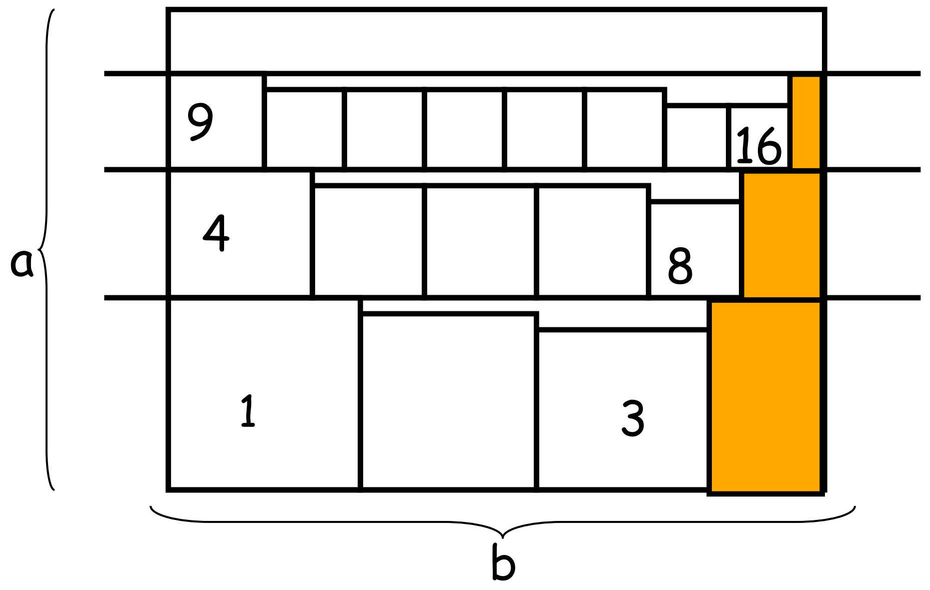
Wasted Space · s(a+b)

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)

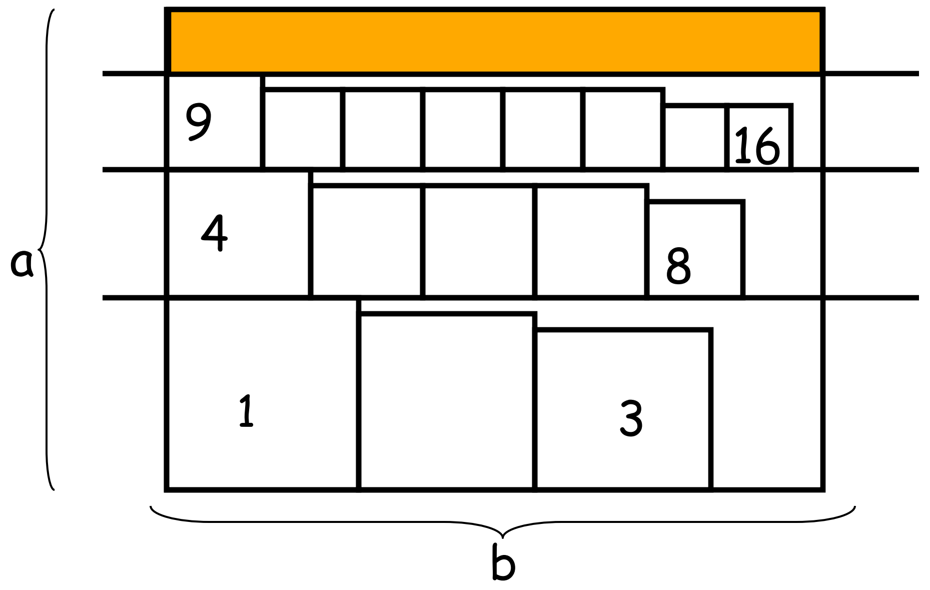
Right side: At most s £ a

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

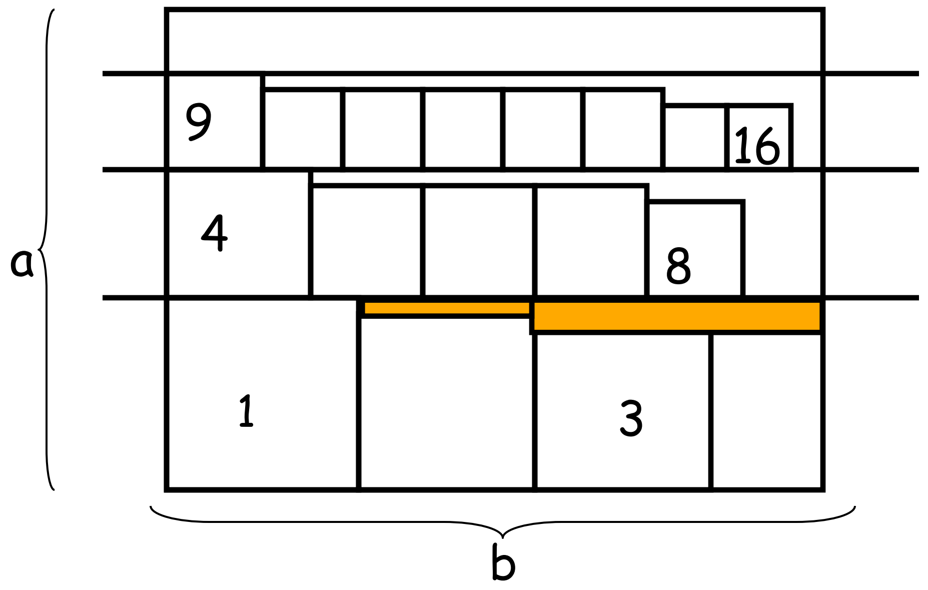
Top · s_{16} b

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

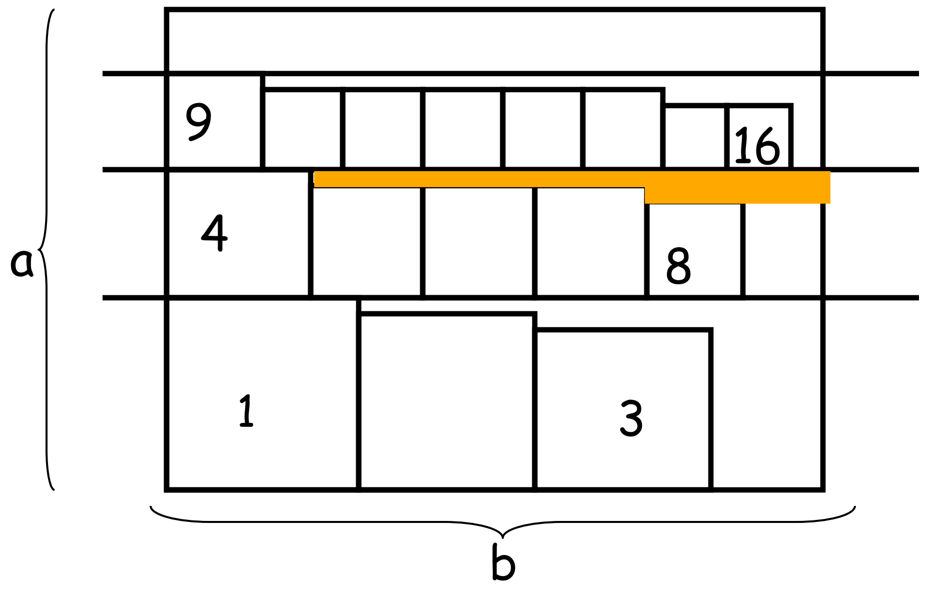
Shelf 1: (s_1 −s_3) b

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

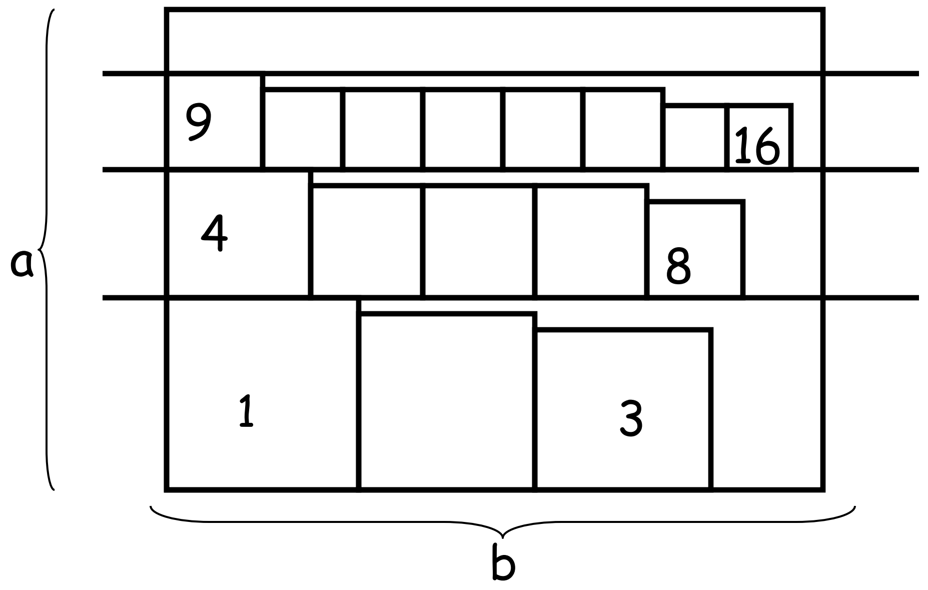
…

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