# Harmonic Algorithms for Packing d-dimensional Cuboids Into Bins

**Eklavya Sharma**

Student at Indian Institute of Science

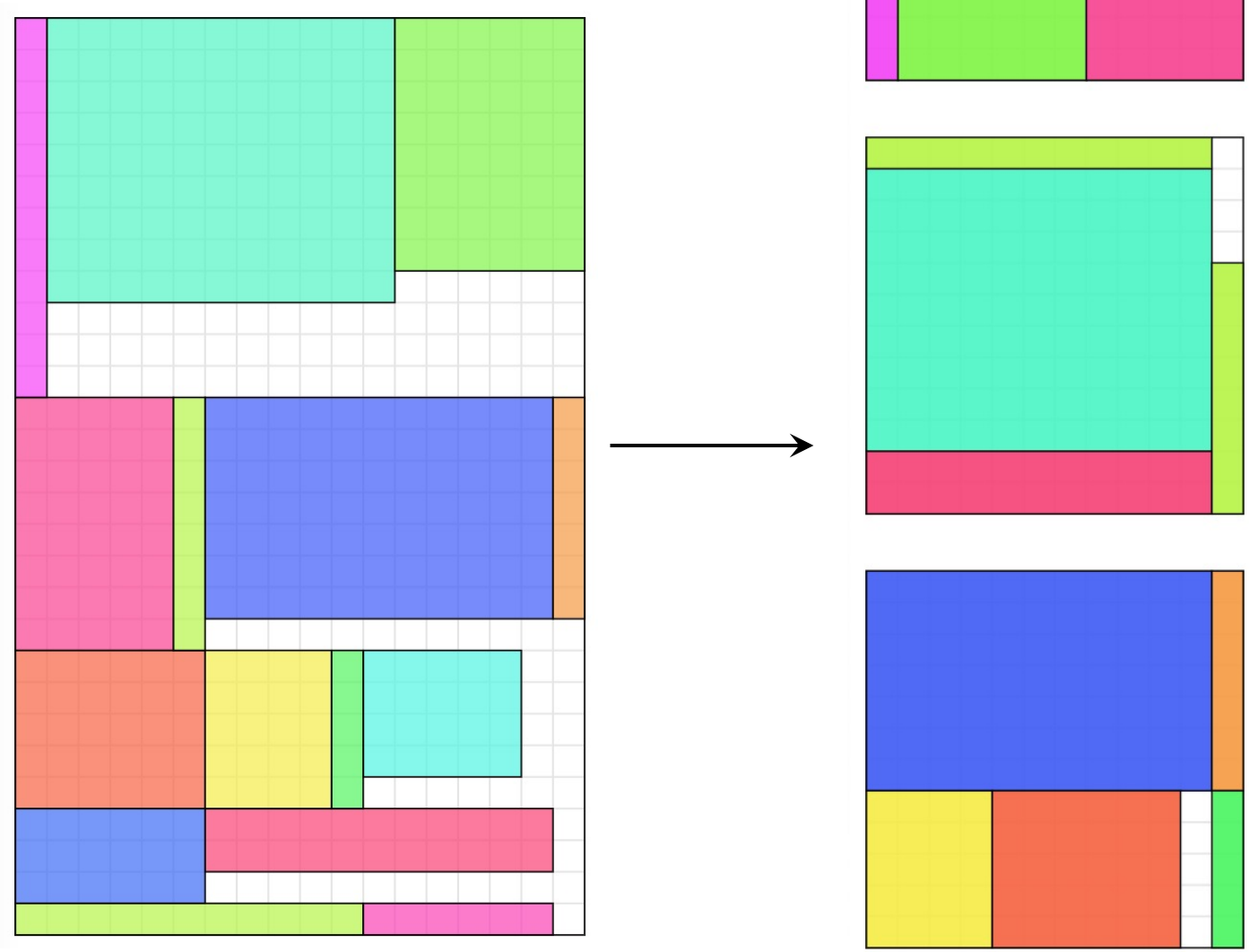
Advised by **Prof. Arindam Khan**

arXiv:2011.10963

# Geometric Bin Packing

# 2BP

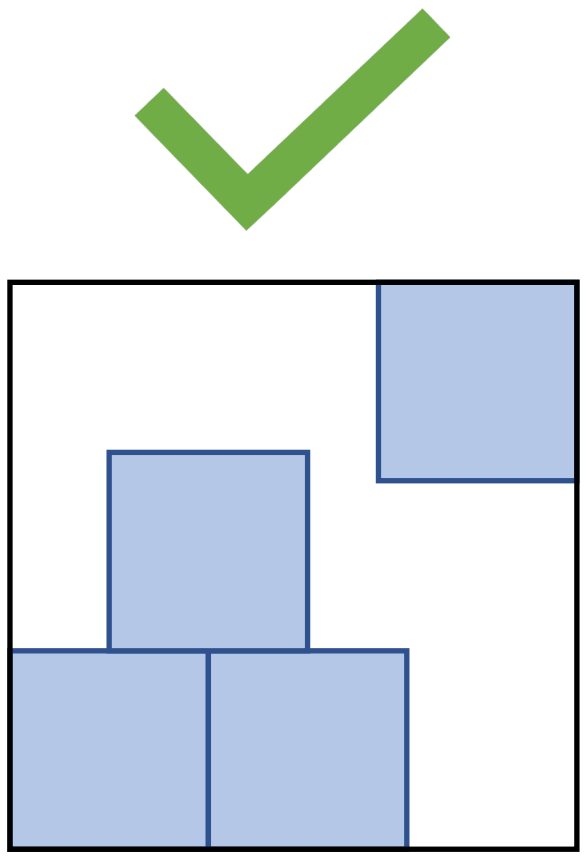
(a.k.a. rectangle packing)

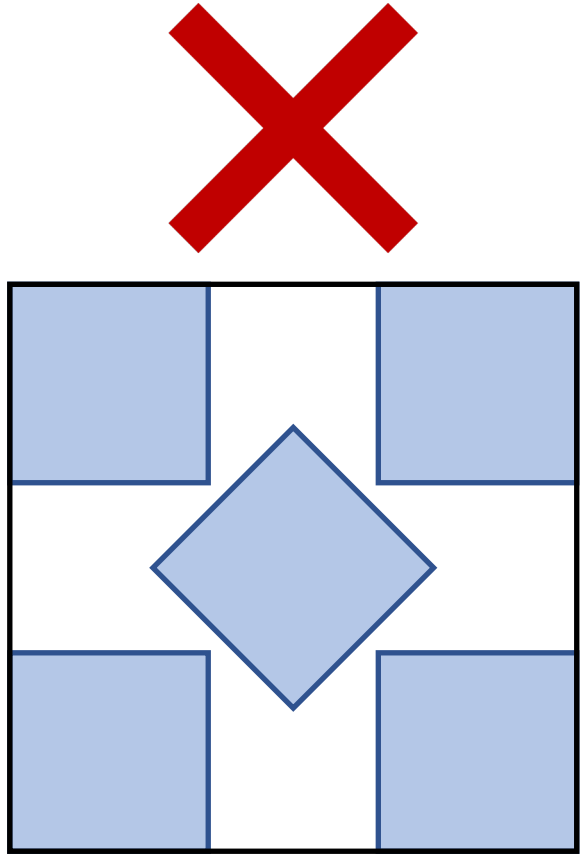


We are given a set $I$ of $n$ rectangular items and an infinite supply of identical rectangular bins.

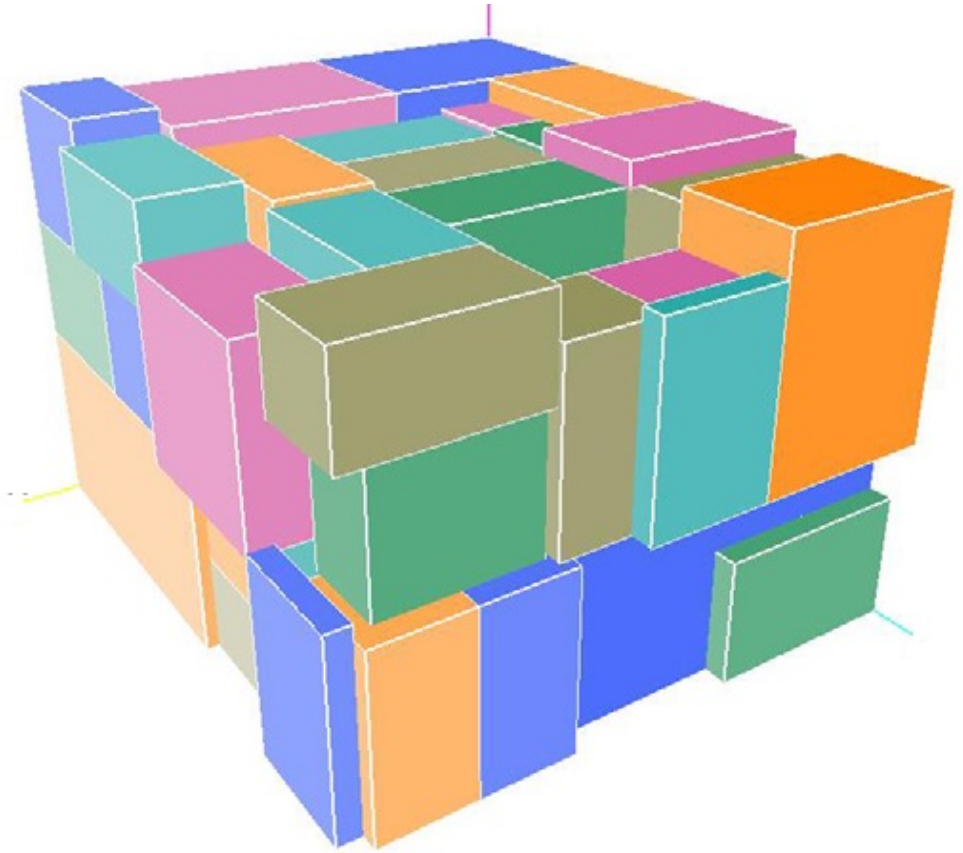
We must pack the items into the minimum number of bins.

# Axis-aligned packing





# 3BP

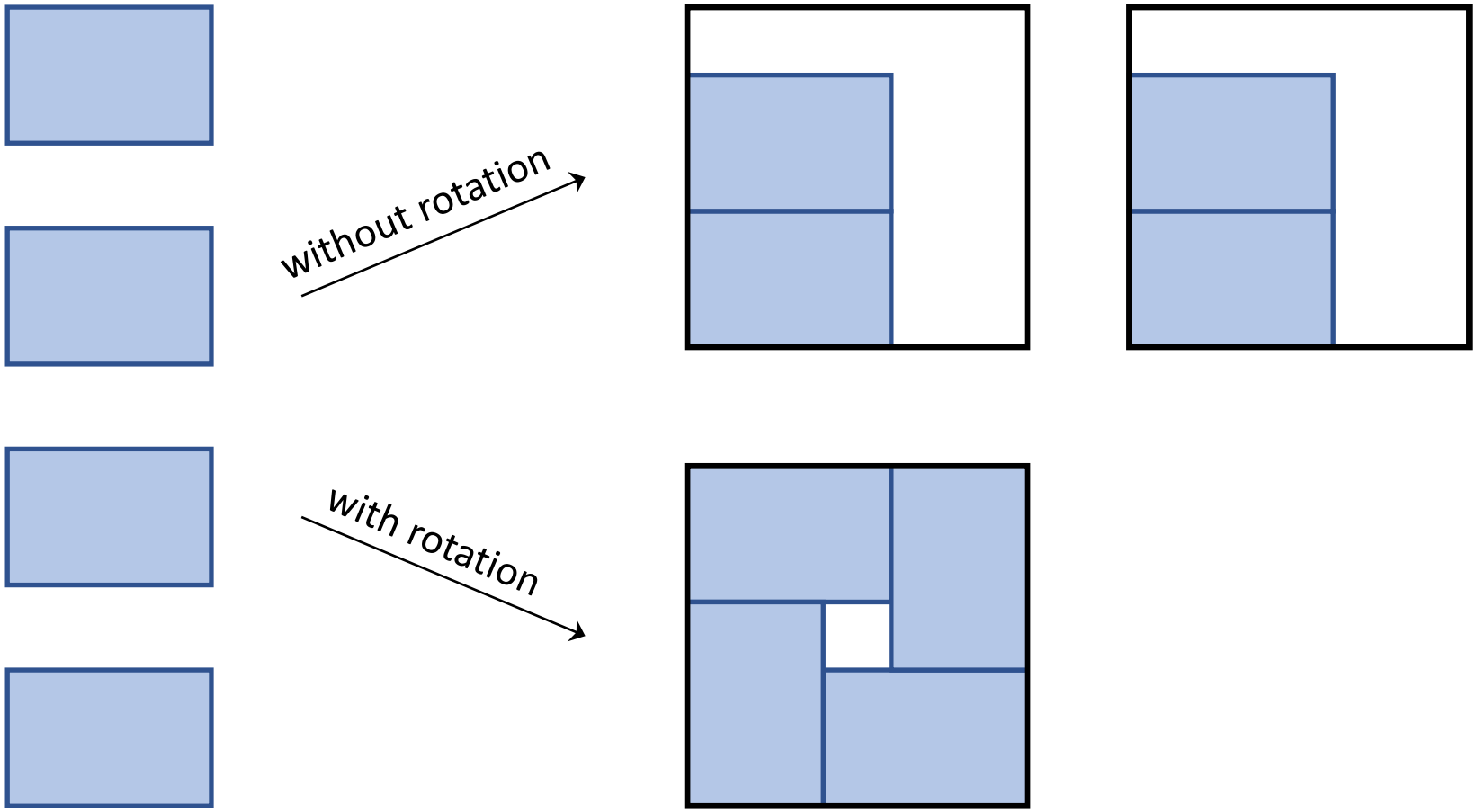


(Pic source: Y. Wu, W. Li, M. Goh, R. Souza. Three-dimensional bin packing problem with variable bin height)

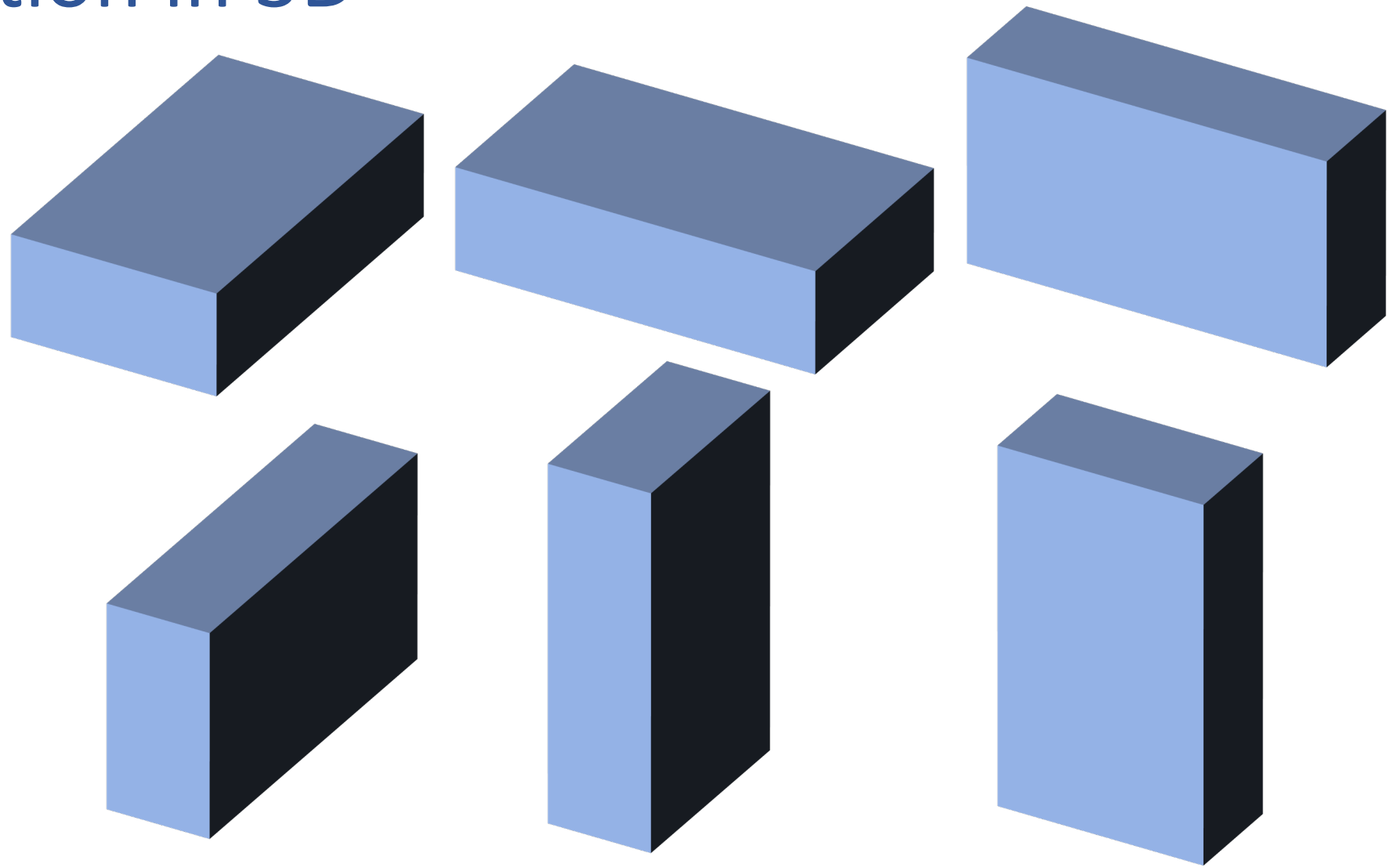
# $d$BP

- We can extend this problem to $d$ dimensions, for any constant $d \geq 1$.
- In 1BP, items and bins are line segments. This is the classical bin packing problem.
- Hard to visualize for $d > 3$. We will focus on $d \leq 3$.
- $d$BP generalizes $(d-1)$BP.

# Rotation by 90°?

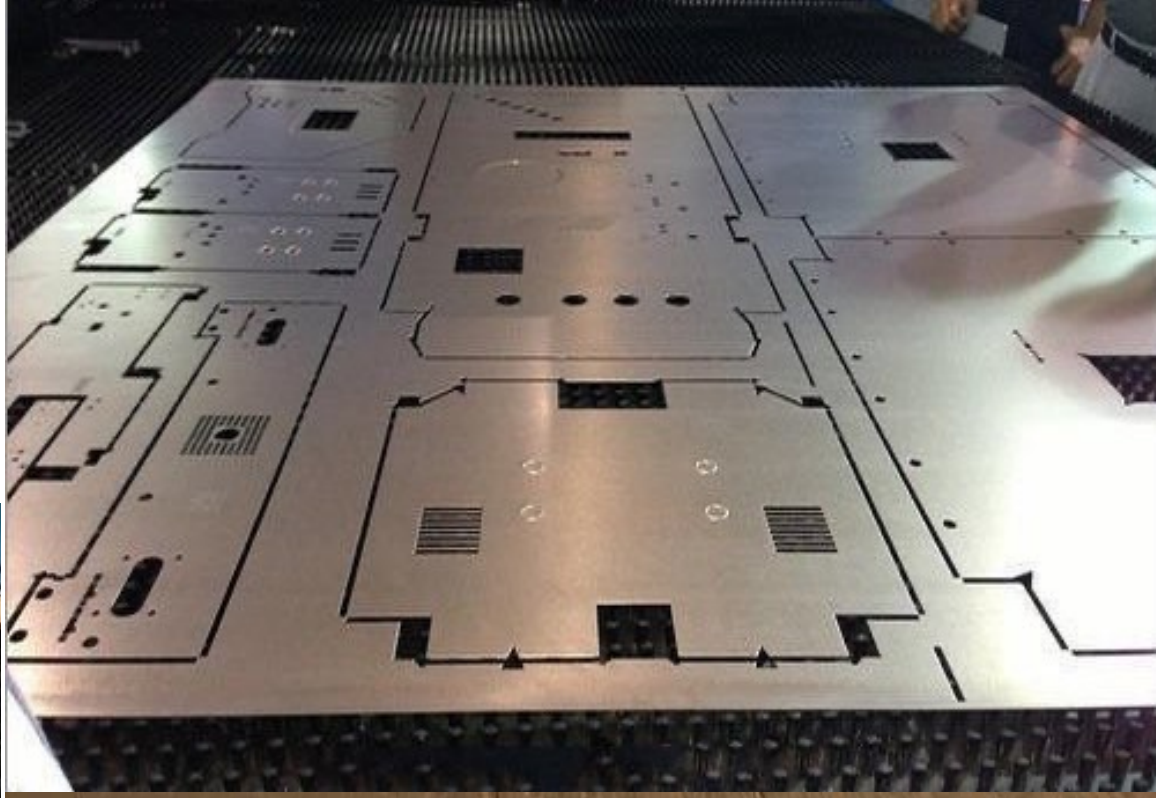


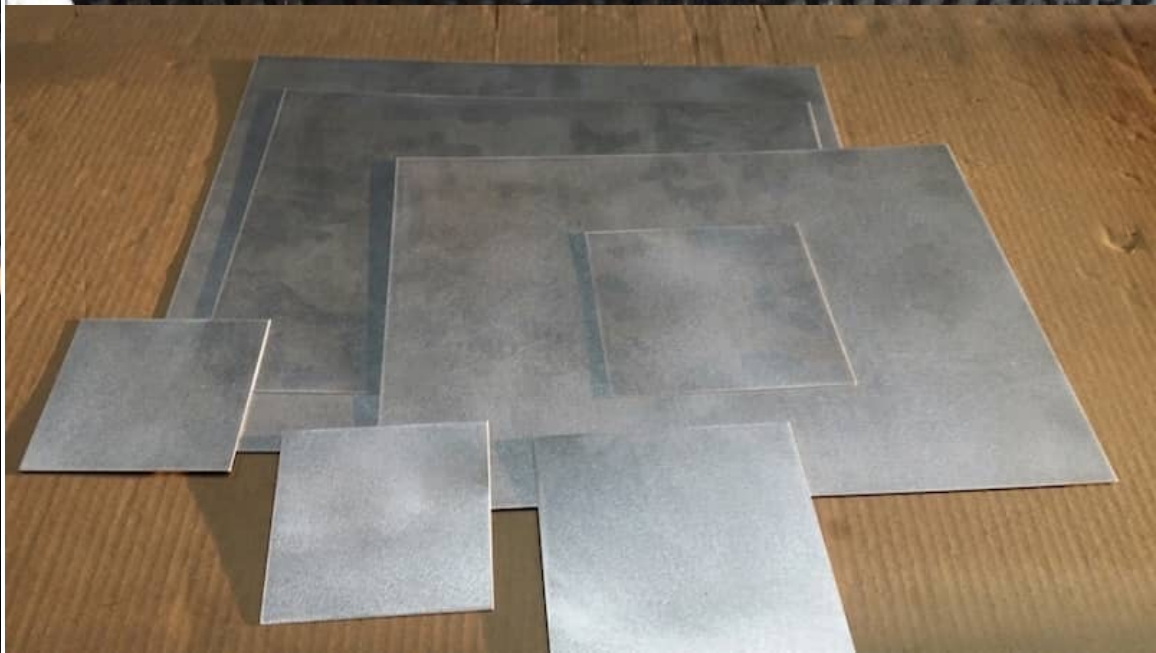
# Rotation in 3D



# 2BP application: cutting metal, wood, cloth, paper

Here rotations are usually allowed



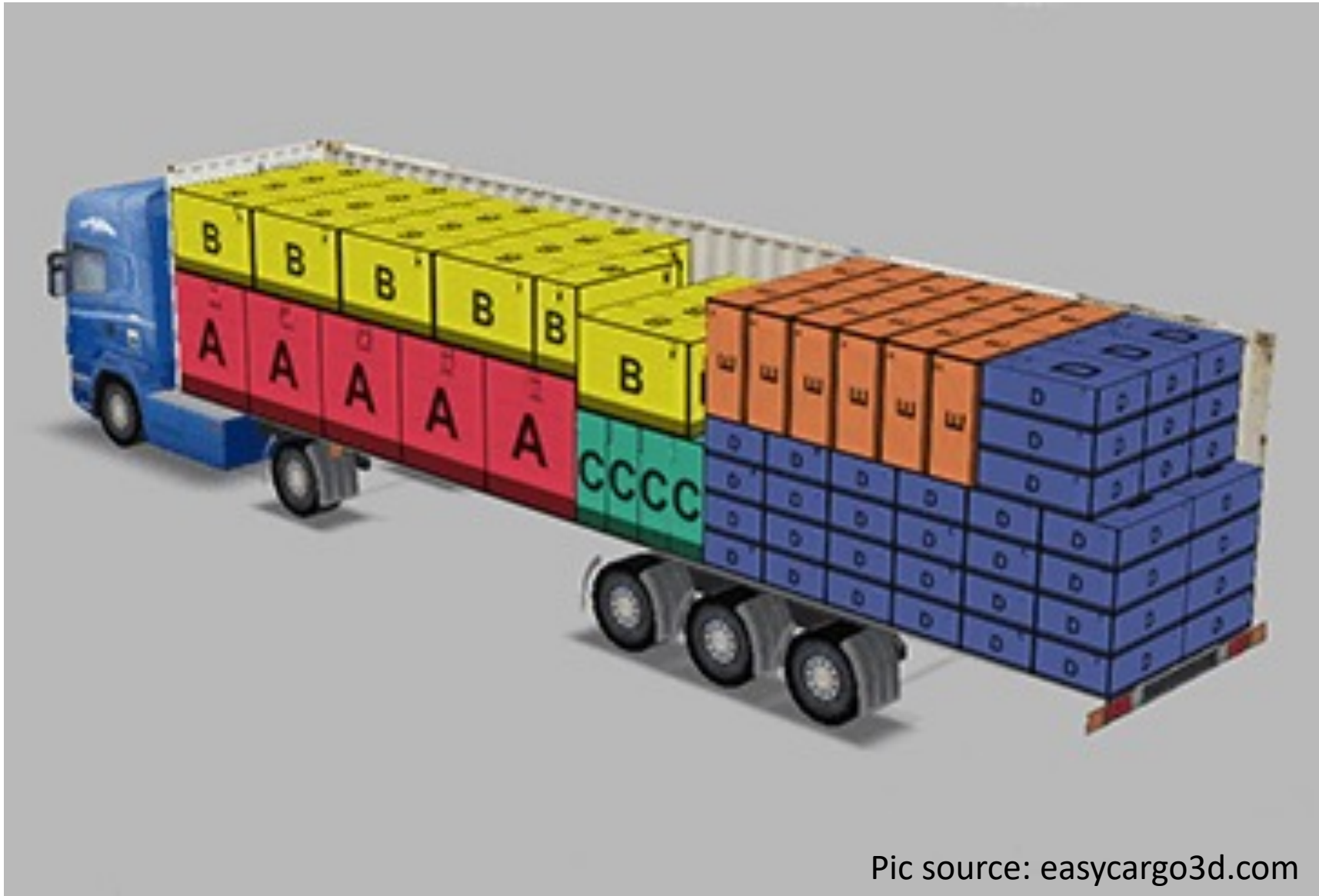


# 2BP application: placing advertisements

Here rotations are forbidden

# 3BP application: transporting boxes

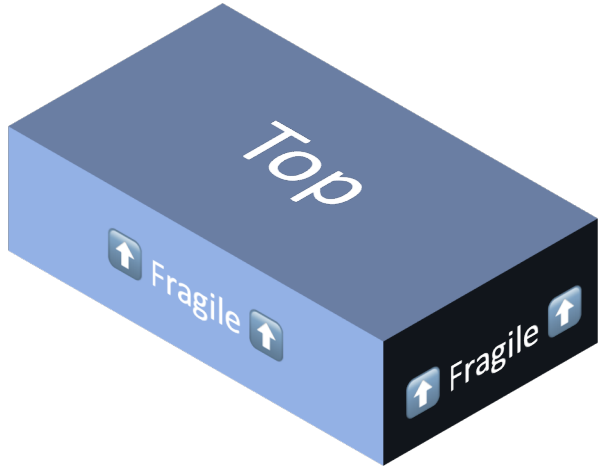
Rotations allowed, but sometimes there are orientation constraints.



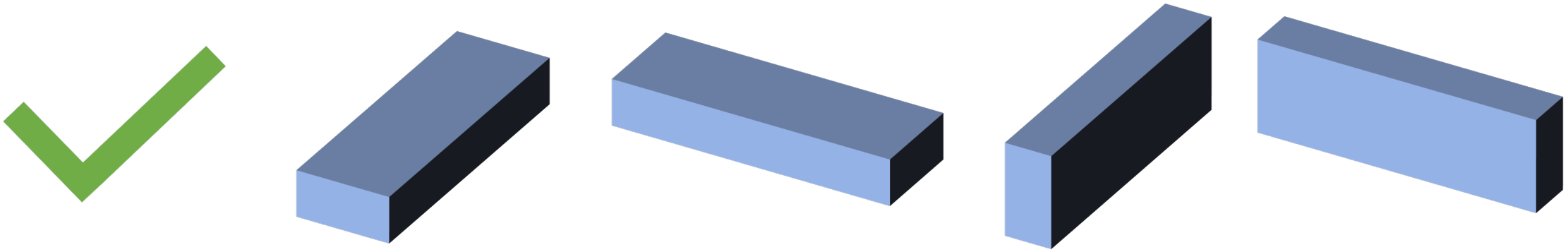

Pic source: easycargo3d.com

# Orientation constraints: this side up



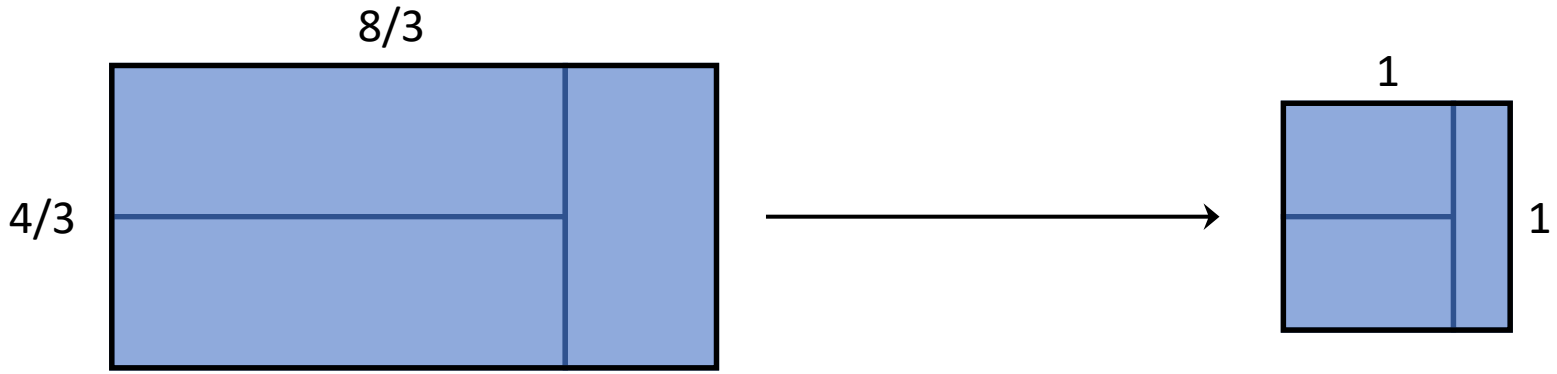


# Orientation constraints: not on smallest face



# Scaling

- No rotation: assume wlog that bin is unit square/cube.
- With rotations: cannot assume this.



# 1BP is NP-hard

- NP-Complete to decide if two bins are sufficient.
- (Standard reduction from partition problem.)
- So, we cannot expect optimal polynomial-time algorithms.

# Approximation Algorithms

Bin packing algorithm $A$ is $\alpha$-approx iff for all inputs $I$,

$$A(I) \leq \alpha \, \mathrm{opt}(I).$$

- $A(I)$ is the number of bins used by algorithm $A$ to pack $I$.
- $\mathrm{opt}(I)$ is the minimum number of bins required to pack $I$.
- Goal: find algorithms with small approximation ratio $\alpha$.

# Hardness of Approximation for 1BP

- NP-Complete to decide if two bins are sufficient.
- No algorithm is better than 3/2-approx.
- There exist 3/2-approx $O(n \log n)$ algorithms: First Fit Decreasing.
- But we can still do better…

# Asymptotic Approximation

- There may exist an $\mathrm{opt}(I) + 1$ algorithm for 1BP.
- In practice, number of bins can be large.
- We will look at guarantees of this form:

$$\forall I, \qquad A(I) \leq \alpha\ \mathrm{opt}(I) + \beta$$

- Here $\alpha$ and $\beta$ are constants. We want $\alpha$ to be small.
- $\alpha$ is called the asymptotic approximation ratio (AAR) of $A$.

Any questions so far?

# Prior work and our results

# Algorithms for 1BP

- Next-Fit: $2\ \mathrm{opt}(I)$
- First-Fit: $\lfloor 1.7\ \mathrm{opt}(I) \rfloor$ [Dósa, Sgall, STACS'13]
- Harmonic: $T_\infty \approx 1.69103$ [Lee, Lee, JACM'85]
- First-Fit Decreasing: $\frac{11}{9}\mathrm{opt}(I) + \frac{6}{9}$ [Dósa, ESCAPE'07]

# 1BP: APTAS and beyond

- Lueker-Vega: $(1+\varepsilon)\,\mathrm{opt}(I)+1$ [Combinatorica'81]
  - Trade-off: running time increases as we decrease $\varepsilon$.
  - This is an Asymptotic Polynomial-Time Approximation Scheme (APTAS)
- Karmarkar-Karp: $\mathrm{opt}(I)+O(\log^2 \mathrm{opt}(I))$ [SFCS'82]
- Hoberg-Rothvoß: $\mathrm{opt}(I)+O(\log \mathrm{opt}(I))$ [SODA'17]
- If we only care about AAR, then 1BP is solved.

# Algorithms for 2BP

| Algorithm for 2BP | Asymptotic Approx Ratio | Rotation? |
|---|---|---|
| NFDH [CGJT, SIAM J. Comput. 1980] | 4 | Yes |
| FFDH [CGJ, CGJT + Caprara] | $187/90 \approx 2.0\bar{7}8$ | No |
| HDH [Caprara, MOR'08] | $T_\infty \approx 1.691$ | No |
| BCS [FOCS'06] | $1 + \ln(T_\infty) + \varepsilon \approx 1.525 + \varepsilon$ | Yes* |
| Jansen, Prädel [SODA'13] | $1.5 + \varepsilon$ | Yes* |
| **Bansal, Khan [SODA'14]** | $\mathbf{1 + \ln(1.5) + \varepsilon \approx 1.405 + \varepsilon}$ | **Yes*** |

** assumes square bin*

# Algorithms for $d$BP ($d \geq 3$)

| Algorithm for 3BP | Asymp Appx Ratio | Rotation? |
|---|---|---|
| Miyazawa, Wakabayashi, [COR'09] | $\approx 4.89$ | Yes |
| Epstein, van Stee, [TALG'06] | $4.5$ | Yes* |
| HDH [Caprara, MOR'08] | $T_\infty^2 \approx 2.860$ | No |

** assumes bins have square base*

| Algorithm for dBP | Asymp Appx Ratio | Rotation? |
|---|---|---|
| HDH [Caprara, MOR'08] | $T_\infty^{d-1}$ | No |
| BCKS [MOR'06] (squares only) | $1 + \varepsilon$ | – |

# Shortcomings of Existing Algorithms

- Some of the algorithms don’t allow rotation.
- Assume square bin.
- All items have the same orientation constraints.

Our work not only addresses these shortcomings but also improves the AAR for rotational 3BP.

# Hardness of Approximation for 2BP

- [BCKS, MOR'06] NP-hard to get an APTAS (unlike 1BP).
- [Chlebík, Chlebíková, JoDA'09] NP-hard to get AAR better than
  - $1 + 1/3792 \approx 1.000264$ without rotation.
  - $1 + 1/2196 \approx 1.000455$ with rotation.
- These results are the best known even for $d$BP for $d \geq 2$.
- There's a large gap between lower and upper bounds. For $d \geq 3$, the best upper bound is $\approx 1.691^{d-1}$.

# Our Results

- We give two algorithms for rotational $d$BP.
- **`fullh`**: simple algorithm, AAR $T_{\infty}^{d}$, runs in $O(n \log n)$ time.
- **`HGaP`**: has AAR $T_{\infty}^{d-1}(1+\varepsilon)$ and runs in $O(n^{(1/\varepsilon)^{O(1/\varepsilon)}})$ time.
- HGaP's AAR matches that of HDH, closing the gap between rotational and non-rotational $d$BP for $d \geq 3$.

# Comparison with prior work for 3BP

| Prior Algorithm for 3BP | Asymp Appx Ratio | Rotation? |
|---|---|---|
| Miyazawa, Wakabayashi [COR'09] | $\approx 4.89$ | Yes |
| Epstein, van Stee [TALG'06] | $4.5$ | Yes* |
| HDH [Caprara, MOR'08] | $T_\infty^2 \approx 2.860$ | No |

* *assumes bins have square base*

| Our Algorithm for 3BP | Asymp Appx Ratio | Rotation? |
|---|---|---|
| fullh | $T_\infty^3 \approx 4.836$ | Yes |
| HGaP | $(1+\varepsilon)T_\infty^2 \approx 2.860(1+\varepsilon)$ | Yes |

Our algorithms work for arbitrary orientation constraints and bin sizes.

Any questions so far?

# Multiple-Choice Bin Packing

(abbr. MCBP)

# Multiple-Choice Bin Packing (MCBP)

- fullh and HGaP can be extended to the $d$MCBP problem, which generalizes rotational $d$BP.
- $d$MCBP offers a perspective that is useful for analyzing rotational $d$BP.
- Helps us handle different orientation constraints and non-square bins.

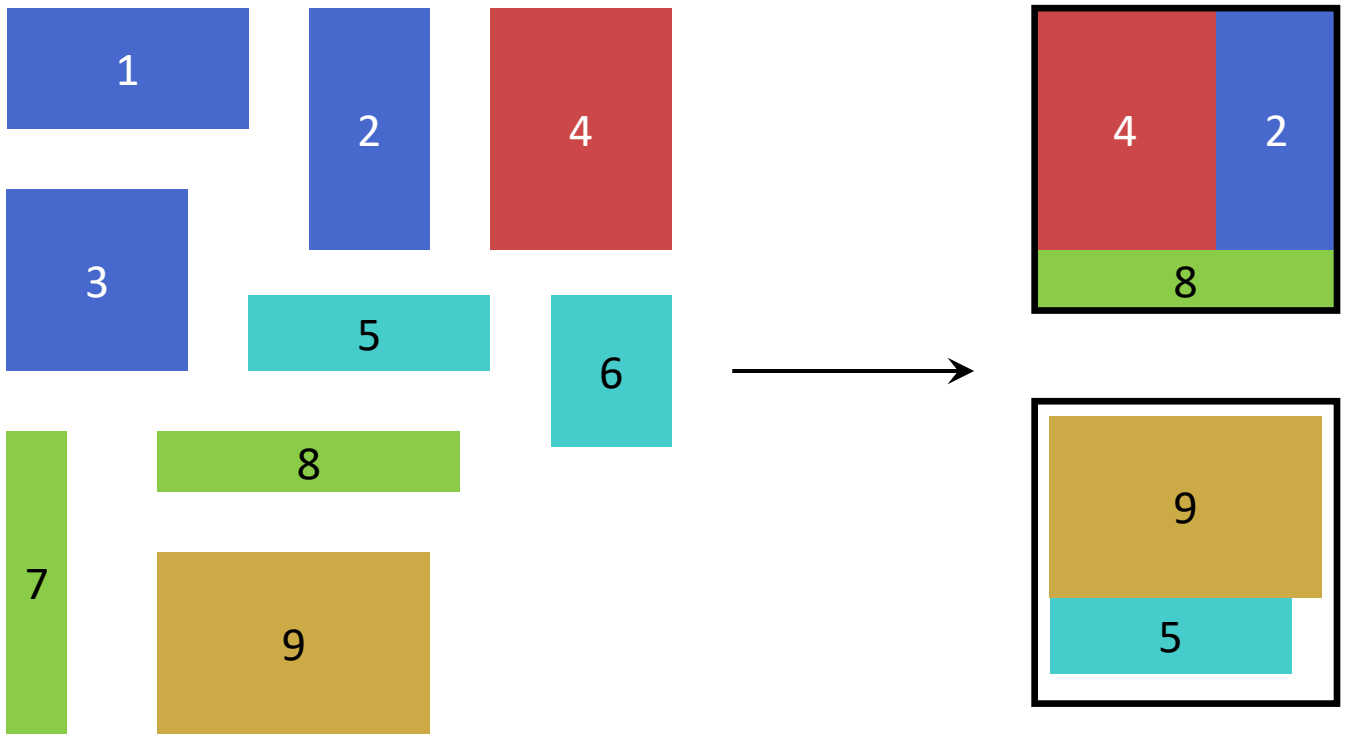
# Multiple-Choice Bin Packing ($d$MCBP)

Input: set of $d$D items of different colors.

Choose one item of each color and pack them into bins without rotating.

A subset containing a single item of each color is called an *assortment*.

Let $\Psi(I)$ be the set of all assortments of $I$.




$$\mathrm{opt}_{\mathrm{MCBP}}(I) = \min_{K \in \Psi(I)} \mathrm{opt}_{\mathrm{BP}}(K)$$

# Example: packing luggage for travel

⋮

choose one

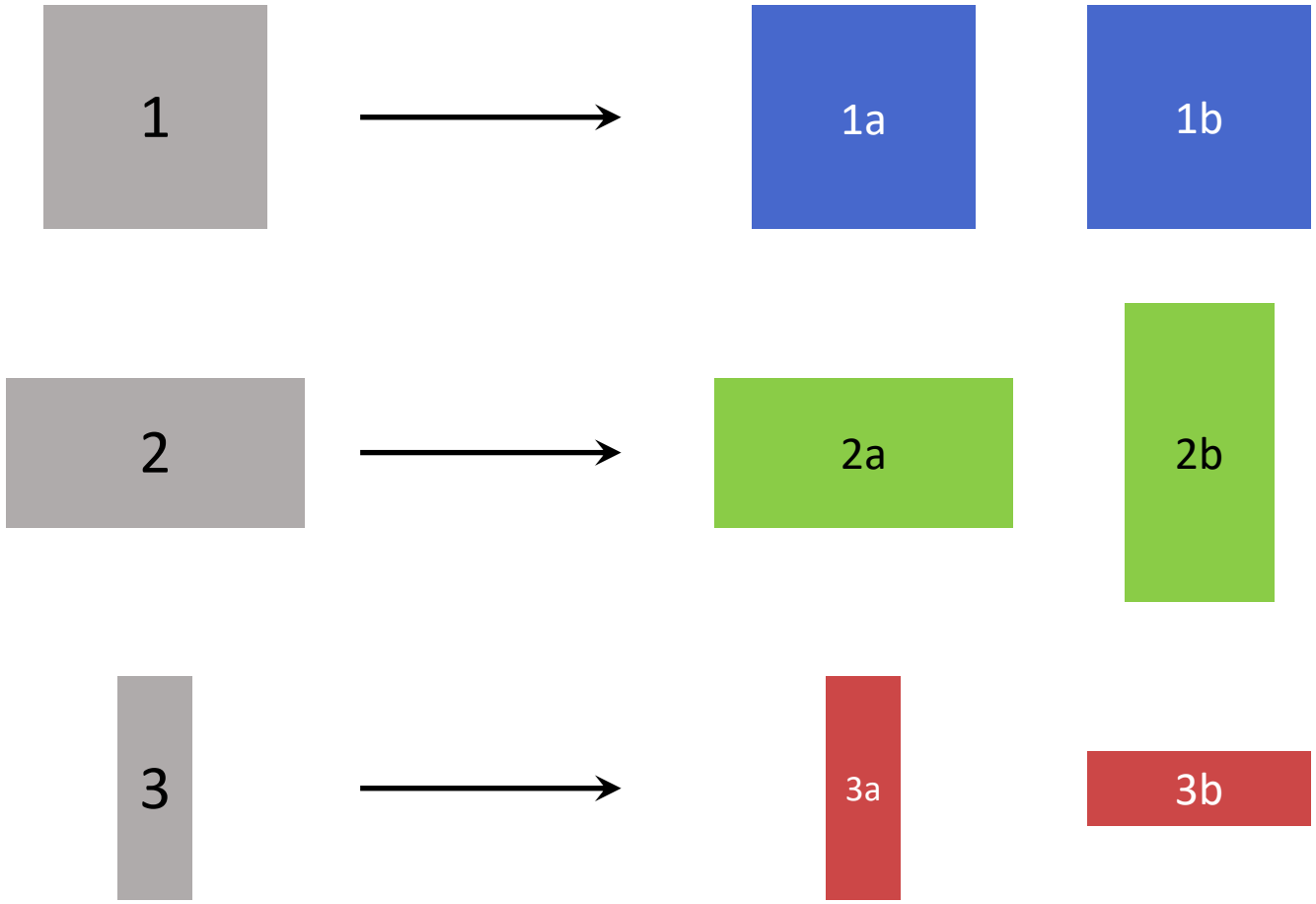
choose one

choose one

# Reducing Rotational $d$BP to $d$MCBP

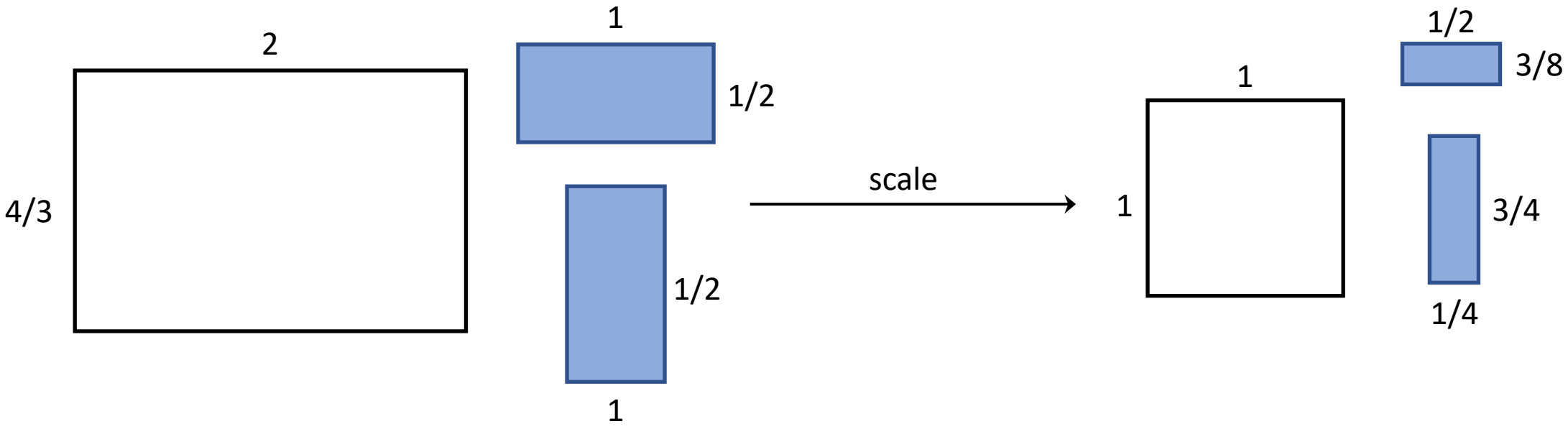
For each item $i$ in BP input, create items in MCBP input corresponding to the allowed orientations of $i$.

This allows different items in the $d$BP input to have different orientation constraints.



# Scaling

- We can assume wlog that bins are unit cubes/squares.
- ∵ in each dimension, we can scale items and bins by same factor.
- After scaling, items of same color may stop being rotations.



# Prior work on other multiple-choice problems

- FPTAS for MC knapsack [Lawler, MOR'79].
- AAR $O(\log d)$ for MC vector BP [Patt-Shamir, Rawitz, JDAM'12].
- PTAS for MC vector knapsack [Patt-Shamir, Rawitz, JDAM'12].

# Example: Extending NFDH to 2MCBP

- NFDH: an algorithm for non-rotational 2BP.
- $\mathrm{NFDH}(K) \leq 4\,\mathrm{area}(K) \,/\, \mathrm{area}(\mathrm{bin}) + 3.$
- $\mathrm{area}(K) \,/\, \mathrm{area}(\mathrm{bin}) \leq \mathrm{opt}_{\mathrm{BP}}(K).$
- Therefore, NFDH is 4-asymptotic-approx for 2BP.

# Example: Extending NFDH to 2MCBP

- MCNFDH: choose item of min area from each color and pack them using NFDH.
- Let $I$ be a 2MCBP instance.
- Let $K^* \subseteq I$ be the assortment chosen in an optimal solution.
- Let $\widehat{K} \subseteq I$ be the assortment chosen by our algorithm.

$$\begin{aligned} \mathrm{NFDH}(\widehat{K}) &\leq 4\,\mathrm{area}(\widehat{K}) \,/\, \mathrm{area}(\mathrm{bin}) + 3 \\ &\leq 4\,\mathrm{area}(K^*) \,/\, \mathrm{area}(\mathrm{bin}) + 3 \\ &\leq 4\,\mathrm{opt}_{\mathrm{BP}}(K^*) + 3 = 4\,\mathrm{opt}_{\mathrm{MCBP}}(I) + 3 \end{aligned}$$

- This gives AAR of 4 for 2MCBP.

Any questions so far?

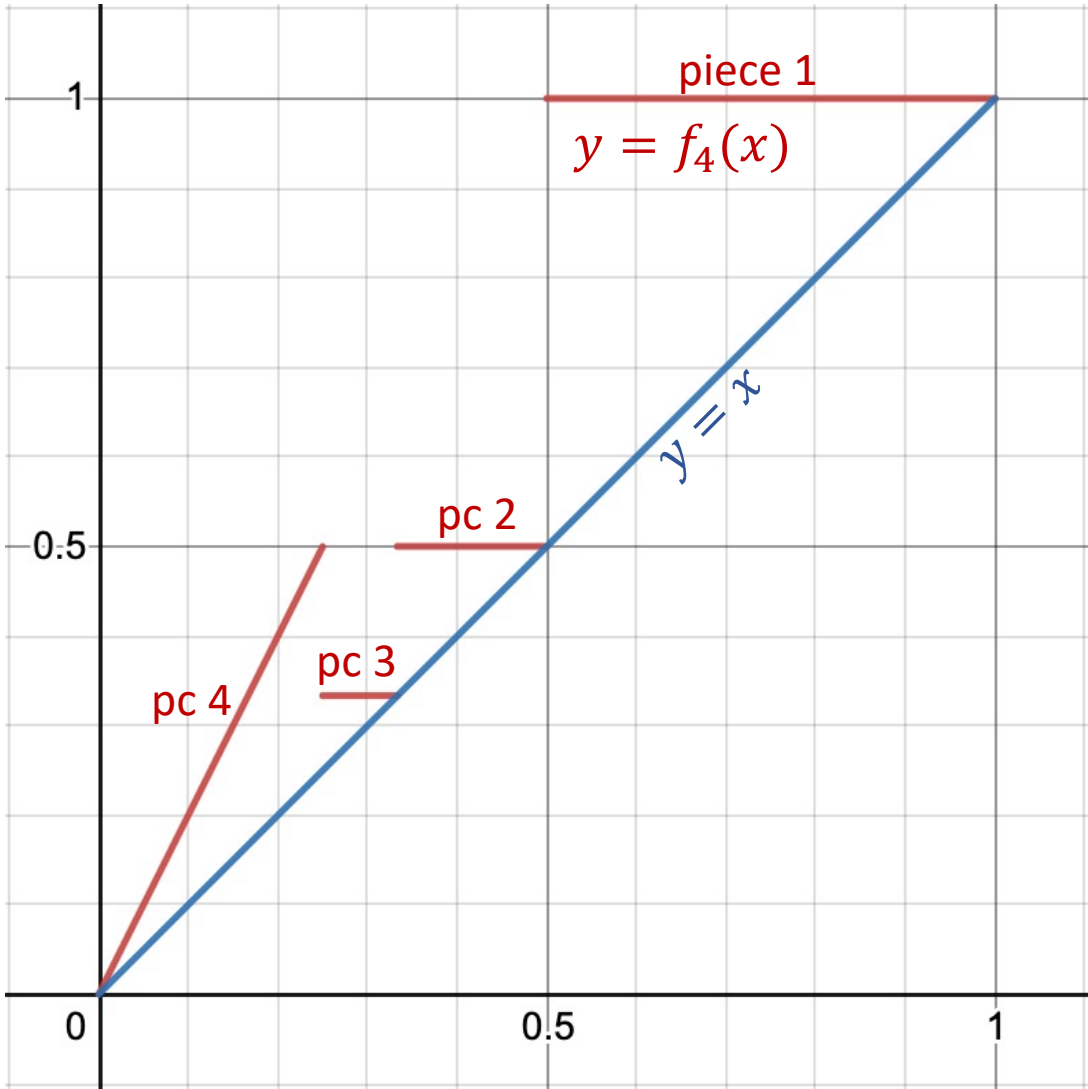
# fullh and preliminaries from HDH

# fullh preview

- The fullh algorithm works similarly. We will describe an additive function $g$ and a constant $T_k$ and show that for every $d$BP input $I$,
  - $\text{fullh}(I) \leq g(I) + O(1)$.
  - $g(I) \leq T_k^d \, \text{opt}_{\text{BP}}(I)$.
- To extend this to $d$MCBP, for each color, we will pick the item $i$ having the least value of $g(i)$, and then pack them using fullh.
- This gives us a $T_k^d$-asymptotic-approx algorithm for $d$MCBP.
- To describe $g$, $T_k$ and fullh, we need preliminaries from HDH.

# Harmonic function [Caprara, MOR'08]

- Let $k \in \mathbb{Z}_{\geq 3}$. Define $f_k: [0,1] \mapsto [0,1]$ as
- $$f_k(x) = \begin{cases} \frac{1}{q} & x \in \left(\frac{1}{q+1}, \frac{1}{q}\right] \text{ for } q \in [k-1] \\ \frac{k}{k-2}x & x \leq \frac{1}{k} \end{cases}$$
- $f_k(x) \geq x$ and $f_k$ has $k$ pieces.
- $\text{type}_k: [0,1] \mapsto \{1, 2, \ldots, k\}$ where $\text{type}_k(x)$ is the piece number that $x$ lies in, where the pieces of $f_k$ are numbered from the right.
- $$\text{type}_k(x) = \begin{cases} q & x \in \left(\frac{1}{q+1}, \frac{1}{q}\right] \text{ for } q \in [k-1] \\ k & x \leq \frac{1}{k} \end{cases}$$



# Harmonic constant

- Let $T_k$ be the value of the following optimization program:

$$\sup_{S \subseteq [0,1]} \sum_{x \in S} f_k(x) \quad \text{where} \sum_{x \in S} x \leq 1 .$$

- $T_k$ decreases as $k$ increases. Let $T_\infty = \lim_{k \to \infty} T_k \approx 1.69103.$
- We can efficiently compute $T_k$ given $k$ [Lee, Lee, JACM'85].

| $k$ | 3 | 4 | 5 | 6 | 7 | $\infty$ |
|---|---|---|---|---|---|---|
| $T_k$ | 3 | 2 | $11/6 = 1.8\overline{3}$ | $7/4 = 1.75$ | $26/15 = 1.7\overline{3}$ | 1.6910302 |

# Lower-bounding $\mathrm{opt}_{\mathrm{BP}}$ using the function $g$

- For an item $i$, let $\ell_j(i)$ be the length of $i$ in the $j^{\text{th}}$ dimension.
- Define $g(i)$, the augmented volume of $i$, as

$$g(i) = \prod_{j=1}^{d} f_k(\ell_j(i)) \quad \text{and} \quad g(S) = \sum_{i \in S} g(i)\,.$$

- Theorem 3: $g(S) \leq T_k^d \,\mathrm{opt}_{\mathrm{BP}}(S)$.
- We will see a proof later.

# HDH-unit-pack$_k$

- For a $d$D item $i$, let type($i$) be a $d$D vector whose $j^{\text{th}}$ component is $\text{type}_k(\ell_j(i))$. So, items have at most $k^d$ different types.
- Let $S$ be a *sequence* of $d$D items where all items have type $t$.
- Let $\text{last}(S)$ be the last item in $S$. Let $g(S - \{\text{last}(S)\}) < 1$.
- Then algorithm HDH-unit-pack$_k$ returns a packing of $S$ into a $d$D unit bin in $O(n \log n)$ time.
- HDH-unit-pack is defined implicitly in [Caprara, MOR'08].

# The fullh$_k$ algorithm for $d$BP

- Partition the items $I$ by type. So, there are at most $k^d$ partitions.
- From each partition, repeatedly pick the smallest prefix $J$ such that $g(J) \geq 1$, and pack $J$ into a $d$D bin using HDH-unit-pack$_k$.
- For each partition, each bin $B$ except the last has $g(B) \geq 1$.
- Hence, number of bins used by fullh$_k$ is at most $g(I) + k^d$.
- We saw that $g(I) \leq T_k^d \, \mathrm{opt}_{\mathrm{BP}}(I)$.
- We can extend fullh$_k$ to a $T_k^d$-asymp-approx algorithm for $d$MCBP.

Any questions so far?

# HGaP

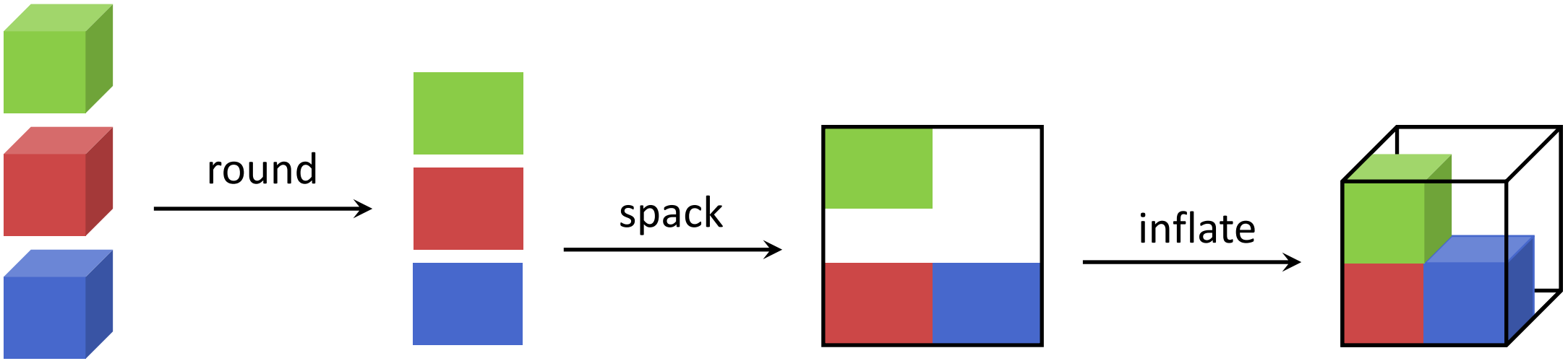
(a.k.a. Harmonic Guess and Pack)

# HGaP$_k$

- Algorithm for $d$MCBP ($d \geq 2$).
- Accepts parameters $k$ and $\varepsilon$, where $k \in \mathbb{Z}_{\geq 3}$ and $\varepsilon \in (0,1)$.
- Has an AAR of $T_k^{d-1}(1+\varepsilon)$.
- Runs in $O\left(N^{O(1/\varepsilon^2)} n^{(1/\varepsilon)^{O(1/\varepsilon)}}\right)$ time, where $N$ is the number of items and $n$ is the number of distinct colors.

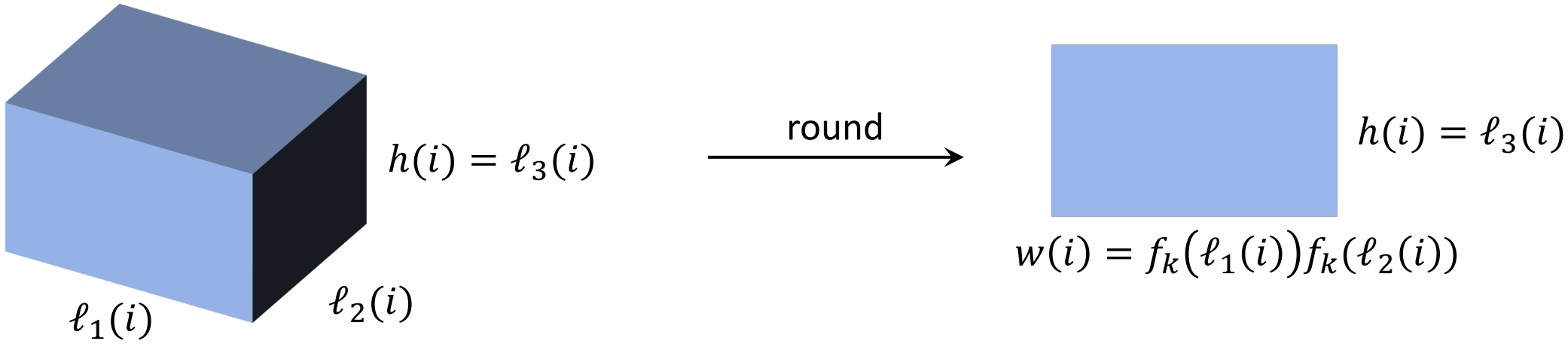
# HGaP: major steps

- Works in 3 major steps: round, spack, inflate.
- spack, which is a 2D packing algorithm, does most of the work.
- So, round and inflate help us bypass complexities of $d$D.



# HGaP: round

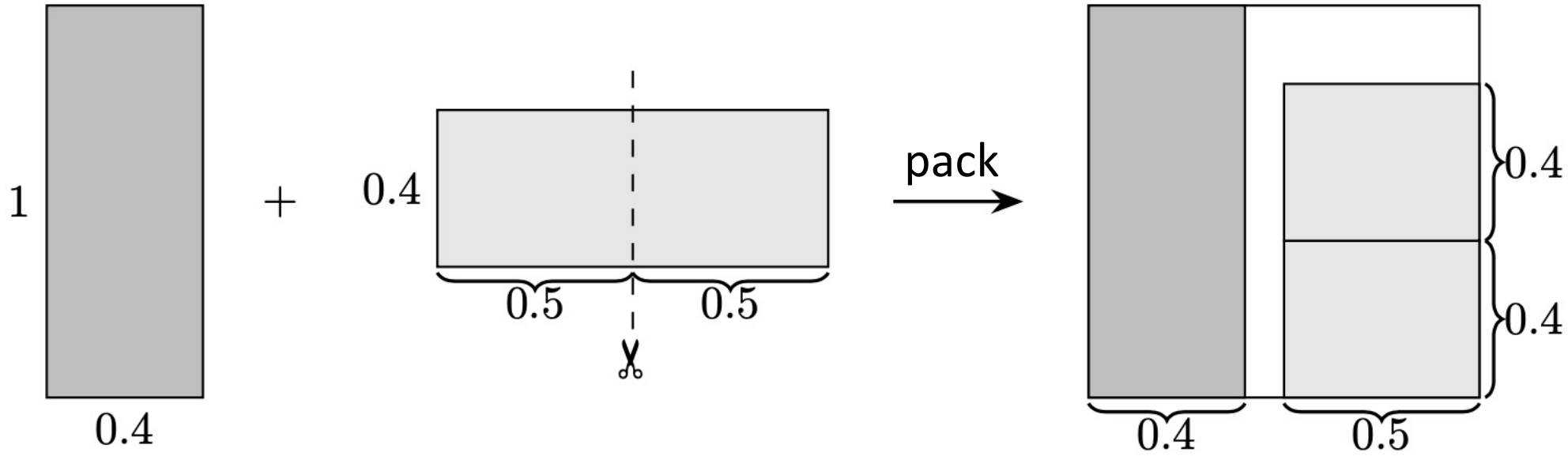
- For a $d$D item $i$, define $\ell_j(i)$ as the length of $i$ in the $j^{\text{th}}$ dimension.
- For a $d$D item $i$, define $h(i) = \ell_d(i)$ and $w(i) = \prod_{j=1}^{d-1} f_k(\ell_j(i))$.
- $\text{round}(i)$ returns a rectangle of width $w(i)$ and height $h(i)$.
- $\text{round}(S) = \{\text{round}(i) : i \in S\}$.



# HGaP: spack

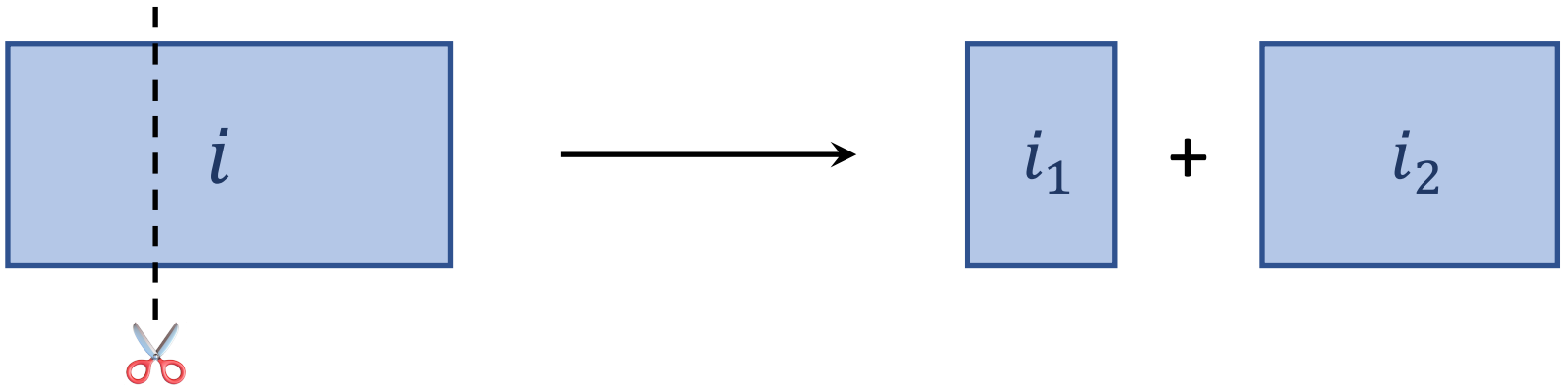
- Let $I$ be a $d$MCBP instance and $\hat{I} = \text{round}(I)$.
- spack finds a near-optimal 'structured $\delta$-fractional' packing of $\hat{I}$, where $\delta = \varepsilon/(2+\varepsilon)$.
- We need to introduce a few concepts to understand what structured and $\delta$-fractional mean.

# HGaP: slicing




- Slicing: cutting an item into 2 parts using a horizontal or vertical cut.
- Slicing can reduce the number of bins needed to pack items.

# HGaP: slicing




- Slicing $i$ using vertical cut: replace with rectangles $i_1$ and $i_2$ such that
  - $h(i) = h(i_1) = h(i_2)$.
  - $w(i_1) + w(i_2) = w(i)$.
- Define slicing using horizontal cut analogously.

# HGaP: $\delta$-fractional packing

- Let $\delta \in (0,1)$ be a constant.
- Let $K$ be a set of rectangular items.
- Items in $K_L = \{i \in K : h(i) > \delta\}$ are called $\delta$-large.
- Items in $K_S = K - K_L$ are called $\delta$-small.
- A $\delta$-fractional bin packing of $K$ is a packing where we can:
  - slice items in $K_L$ recursively using vertical cuts only.
  - slice items in $K_S$ recursively using both horizontal and vertical cuts.

# HGaP: $\delta$-small items

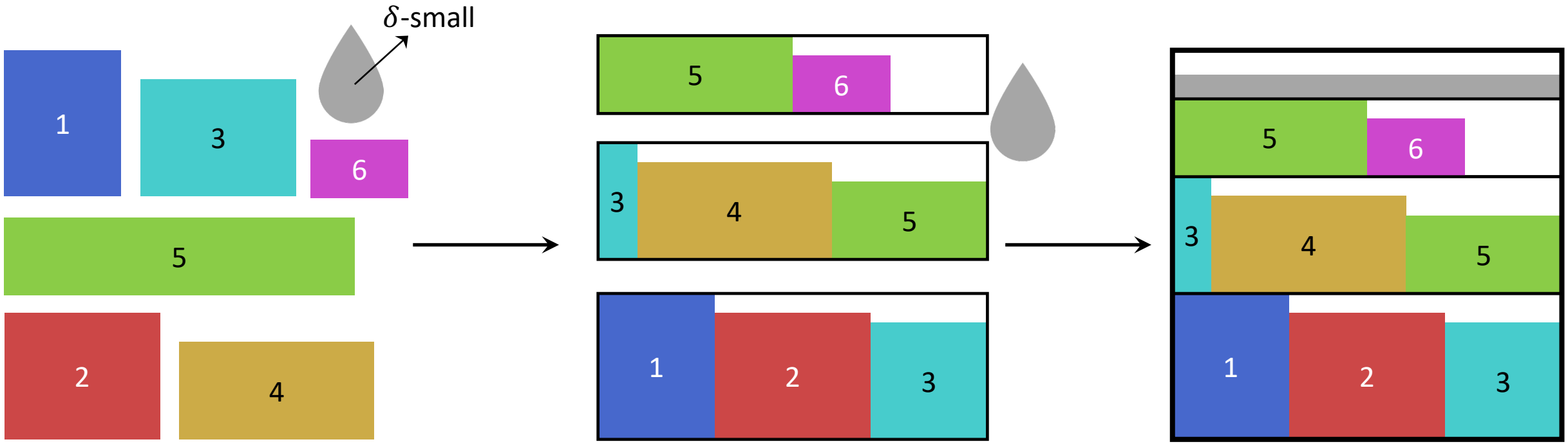
- $\delta$-small items can be cut into tiny pieces of any size.
- So, we can treat them like liquid.
- As a real-life analogy, sand is made up of solid particles, but for many practical applications, we can treat it as a liquid.





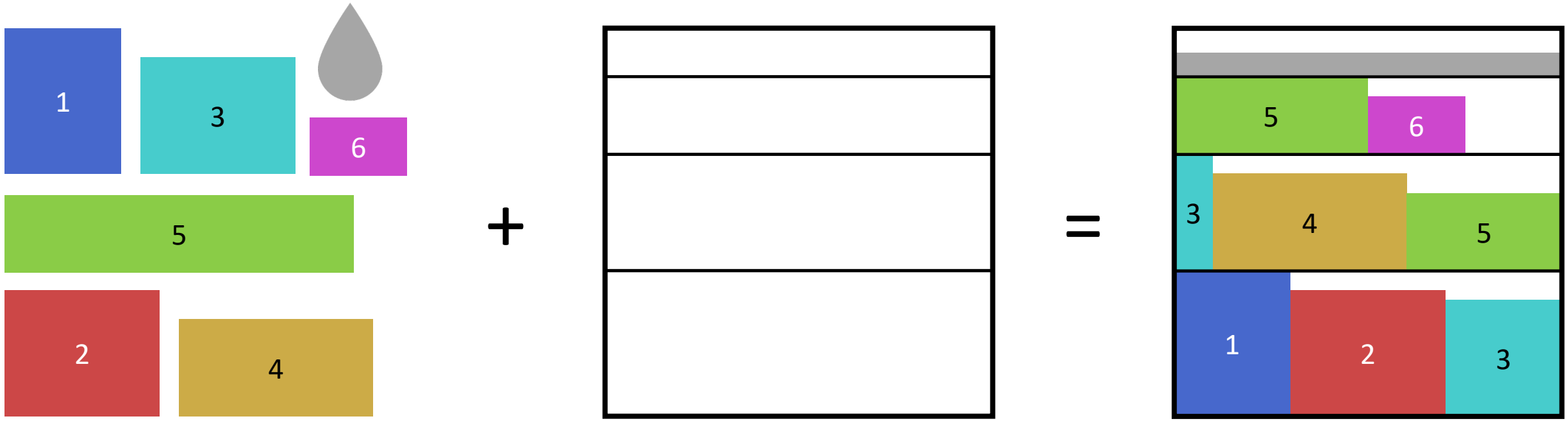

# HGaP: shelf-based $\delta$-fractional packing

- Pack items in two stages: $\delta$-large items into shelves (rectangular containers of width 1), then shelves and $\delta$-small items into bins.
- In each shelf, bottom edge of each item touches bottom edge of shelf.



# HGaP: shelf-based $\delta$-fractional packing

Different perspective: First create shelves and then pack items.



# HGaP: structured $\delta$-fractional packing

- A shelf-based $\delta$-fractional packing of items $K$ is called ‘structured’ iff
  - the number of distinct heights of shelves is at most $\lceil 1/\delta^2 \rceil$.
  - the height of each shelf equals the height of some item in $K$.
- This allows us to guess the heights of shelves.
- $\mathrm{sopt}_\delta(K)$: number of bins in an optimal structured $\delta$-fractional packing of rectangles $K$.

$$\mathrm{soptmc}_\delta(\hat{I}) = \min_{\widehat{K}=\Psi(\hat{I})} \mathrm{sopt}_\delta(\widehat{K})$$

# HGaP: spack

- Recall that HGaP has 3 steps: round, spack, inflate.
- Let $\hat{I} = \mathrm{round}(I)$.
- spack computes a structured $\delta$-fractional packing of $\hat{I}$ into at most $\mathrm{soptmc}_{\delta}(\hat{I}) + 1$ bins.

# HGaP: spack

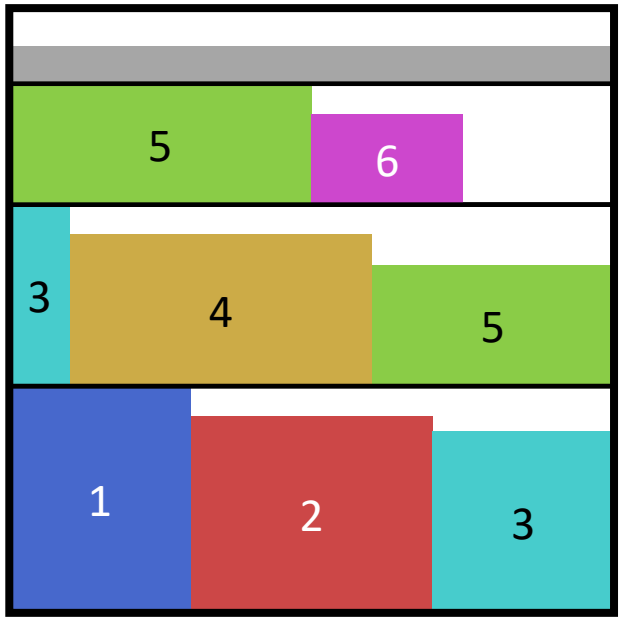
- spack works using a mix of brute-force and dynamic programming.
- First guess the $\lceil 1/\delta^2 \rceil$ distinct heights of shelves.
- Guess the number of shelves of each height and their bin packing. Suppose there are $m$ bins.
- Using dynamic programming, we can either pack the items into $m + 1$ bins or declare that the items cannot be packed into these $m$ bins.
- The state space of the DP contains the total width of shelves of each type used to pack a prefix of the items.

# HGaP: inflate

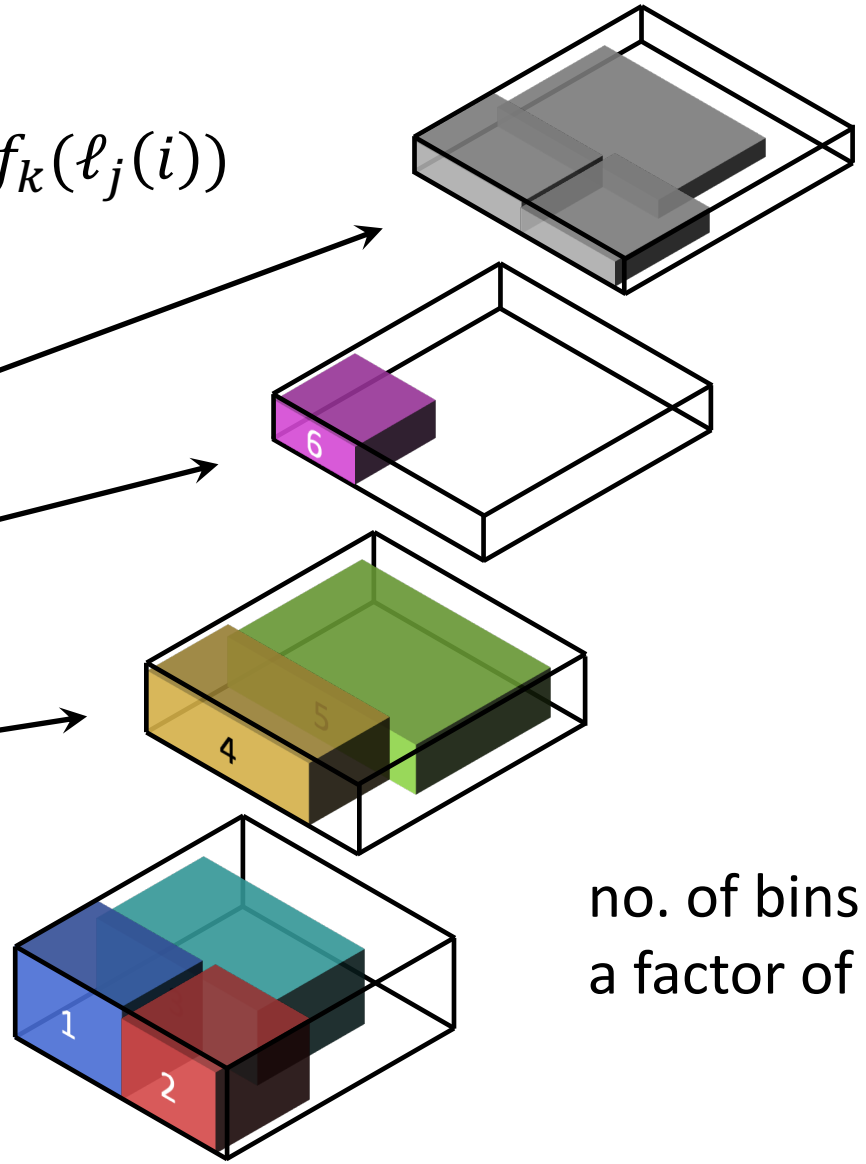
$$w(i) = \prod_{j=1}^{d-1} f_k(\ell_j(i)).$$

$$g(i) = \prod_{j=1}^{d} f_k(\ell_j(i))$$

$g(I - \{last(I)\}) < 1 \Longrightarrow$
HDH-unit-pack can pack $I$




HDH-unit-pack




no. of bins increases ≈ by a factor of $1/(1-\delta)$.

# Overview of analysis of HGaP

- Total number of bins used is $\approx \mathrm{soptmc}_\delta(\hat{I})$, where $\hat{I} = \mathrm{round}(I)$.
- We need to upper-bound $\mathrm{soptmc}_\delta(\hat{I})$ in terms of $\mathrm{opt}_{\mathrm{MCBP}}(I)$.
- Structural Theorem: Let $\widehat{K} = \mathrm{round}(K)$. Then

$$\mathrm{sopt}_\delta(\widehat{K}) < T_k^{d-1}(1+\delta)\,\mathrm{opt}_{\mathrm{BP}}(K) + O(1/\delta^2).$$

- Let $K \in \Psi(I)$ s.t. $\mathrm{opt}_{\mathrm{MCBP}}(I) = \mathrm{opt}_{\mathrm{BP}}(K)$. Let $\widehat{K} = \mathrm{round}(K)$. Then

$$\mathrm{soptmc}_\delta(\hat{I}) \le \mathrm{sopt}_\delta\left(\widehat{K}\right) \lessapprox T_k^{d-1}\,\mathrm{opt}_{\mathrm{BP}}(K) = T_k^{d-1}\,\mathrm{opt}_{\mathrm{MCBP}}(I)\,.$$

definition of soptmc — Structural theorem — definition of $K$

Any questions so far?

# Structural Theorem for HGaP

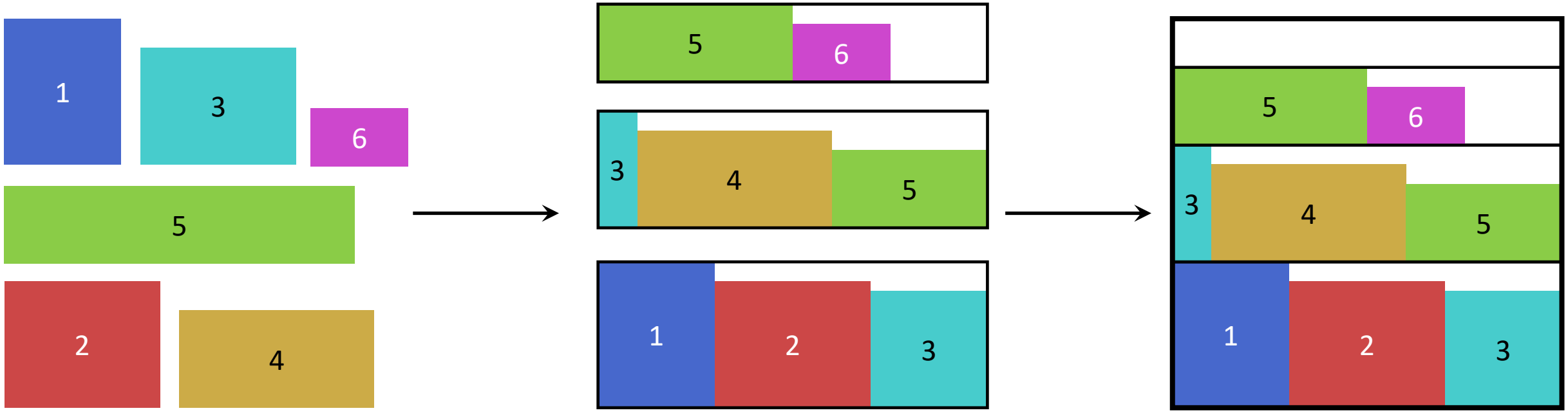
$$\mathrm{sopt}_{\delta}(\mathrm{round}(I)) < T_k^{d-1}(1+\delta)\,\mathrm{opt}_{\mathrm{BP}}(I) + O(1/\delta^2)$$

# Shelf-based packing

- Let $\hat{I}$ be a set of rectangular items. Let $\delta \in (0,1)$ be a constant.
- Assume (for simplicity) that $\hat{I}$ has no $\delta$-small items.
- structured = shelf-based + height-constraints.
- Can we say something meaningful about the optimal shelf-based $\delta$-fractional packing?

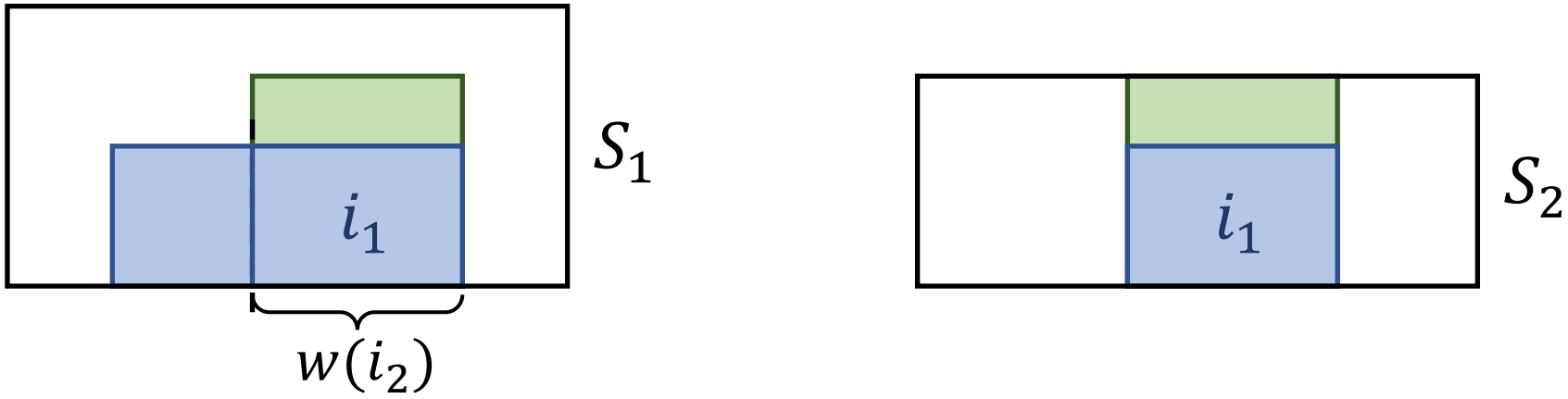
# Optimal shelf-based packing

- Two stages: pack items into shelves, then pack shelves into bins.
- Second stage is the 1BP problem.
- Can we say something about what happens in the first stage?



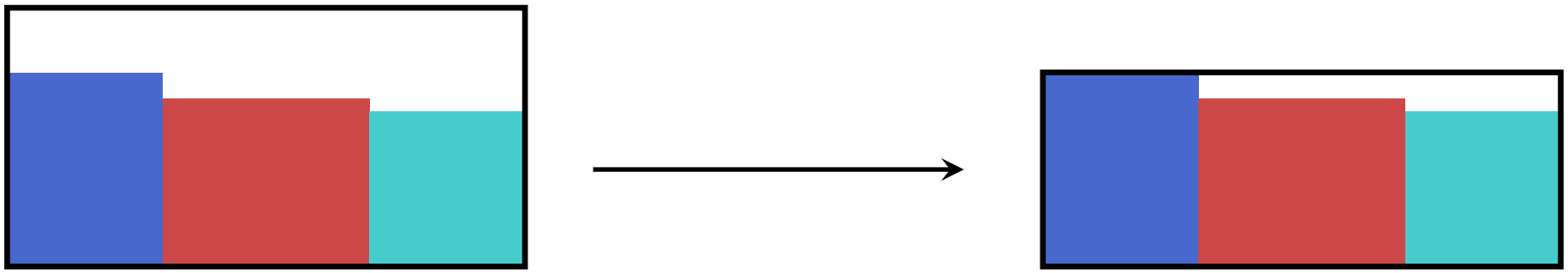
# Swapping items

- Consider shelves $S_1$ and $S_2$, where $h(S_1) \geq h(S_2)$.
- Consider items $i_1 \in S_1$ and $i_2 \in S_2$.
- WLoG assume $w(i_1) = w(i_2)$ (otherwise slice the wider item).
- Suppose $h(i_1) < h(i_2)$. We can swap $i_1$ and $i_2$.
- So, we can rearrange items so that taller shelves have taller items.



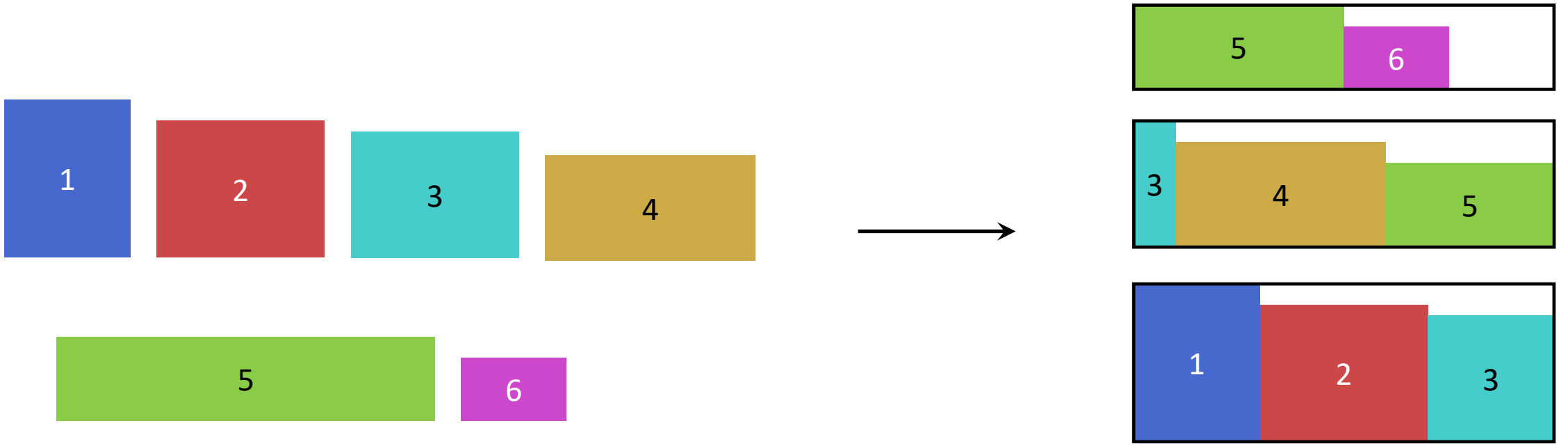
# Tightening shelves

- Tightening: reduce height of shelf to that of tallest item in shelf.
- We can tighten all shelves and the number of bins won’t increase.



# Greedily packing items into shelves

- We can assume wLoG that the shelves in an optimal shelf-based packing are produced by the following greedy algorithm:
  - Arrange items in decreasing order of height.
  - Pack into them into tight shelves one-by-one, slicing them if needed.

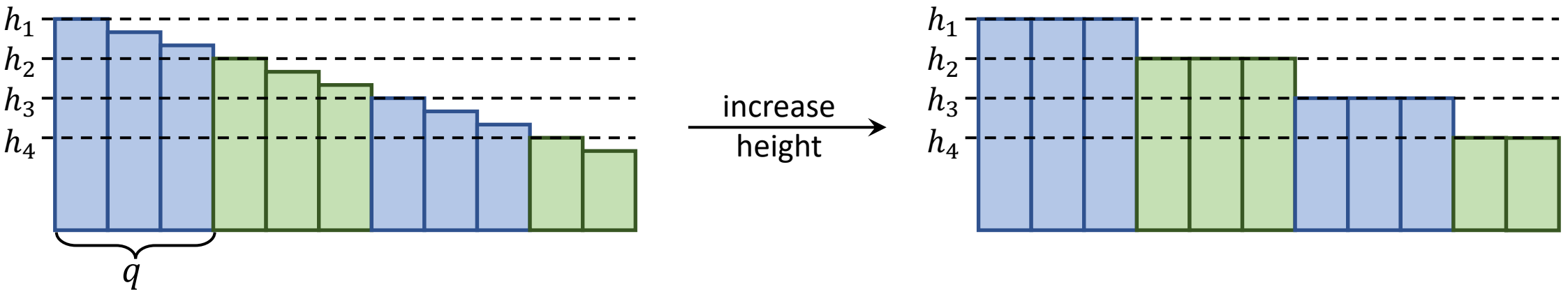


# Optimal shelf-based packing

- Let $J$ be the shelves output by the greedy algorithm on $\hat{I}$.
- No. of bins in the optimal shelf-based packing of $\hat{I}$ is $\mathrm{opt}_{\mathrm{1BP}}(J)$.
- The shelves $J$ may not be structured.
- We can fix this using linear grouping.

# Linear Grouping [Lueker, Vega, Combinatorica'81]

- Let $q = \lfloor \delta \, \text{size}(J) \rfloor + 1$.
- Arrange shelves in decreasing order of height. Make groups of $q$ shelves each.
- $J_j$: items in $j^{\text{th}}$ group. $h_j$: height of tallest item in $j^{\text{th}}$ group.
- Linear grouping: increase height of shelves in each $J_j$ to $h_j$. Let $J'$ be the result.
- No. of distinct heights in $J'$ is at most $\lceil |J|/q \rceil \leq \lceil 1/\delta^2 \rceil$.
- We can prove that $\text{opt}(J') \leq (1+\delta)\text{opt}(J) + 1$.
  (because $J' - J'_1$ can be packed inside $J$ and $|J'_1| \leq q \leq \delta \text{opt}(J) + 1$)



# Linear Grouping

- $J'$ is a set of structured shelves, so

$$\mathrm{sopt}_{\delta}(\hat{I}) \leq \mathrm{opt}_{\mathrm{1BP}}(J') \leq (1+\delta)\mathrm{opt}_{\mathrm{1BP}}(J) + 1$$

- So, we'll now try to upper-bound $\mathrm{opt}_{\mathrm{1BP}}(J)$ in terms of $\mathrm{opt}_{d\mathrm{BP}}(I)$.
- To do this, we will use linear programs.

# Linear Programming Relaxation

- We can express 1BP as an integer linear program.
- Let $C \subseteq J$ such that $\text{sum}(C) \leq 1$. Then $C$ is called a configuration of $J$.
- Packing $J$ into bins is the same as deciding each bin's configuration.

$$\text{LP}(J) = \begin{cases} \min\limits_{x} & \sum\limits_{C} x_C & \\ \text{where} & \sum\limits_{C \ni i} x_C \geq 1 & \forall\, i \in J \\ \text{and} & x_C \geq 0 & \forall \text{ config } C \end{cases}$$

# Dual Linear Program

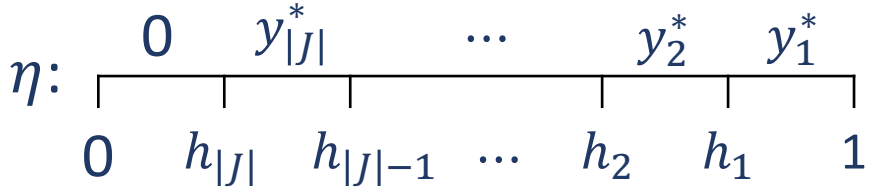
- From Karmarkar and Karp's work, we get $\mathrm{opt}(J) \approx \mathrm{opt}\big(\mathrm{LP}(J)\big)$.

$$\mathrm{DLP}(J) = \begin{cases} \max\limits_{y} & \sum\limits_{i} y_i & \\ \text{where} & \sum\limits_{i \in C} y_i \leq 1 & \forall \text{ config } C \\ \text{and} & y_i \geq 0 & \forall\, i \end{cases}$$

- By strong duality, we get $\mathrm{opt}(\mathrm{DLP}(J)) = \mathrm{opt}(\mathrm{LP}(J))$.
- So, we'll now try to upper-bound $\mathrm{opt}(\mathrm{DLP}(J))$ in terms of $\mathrm{opt}_{d\mathrm{BP}}(I)$.

# Monotonic function $\eta$

- Let $y^*$ be the optimal solution to $\mathrm{DLP}(J)$.
- Let $h_j$ be the height of the $j^{\text{th}}$-tallest shelf.
- Define $\eta$: $[0,1] \mapsto [0,1]$ as



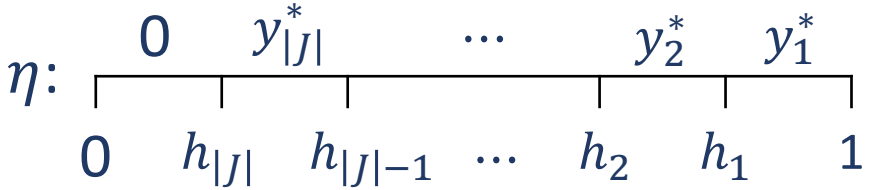

$$\eta(x) = \begin{cases} y_1^* & x \geq h_1 \\ y_{j+1}^* & x \in [h_{j+1}, h_j) \text{ for } 1 \leq j \leq |J| - 1 \\ 0 & x < h_{|J|} \end{cases}$$

- We can assume wLoG that $y_1^* \geq y_2^* \geq \cdots \geq y_{|J|}^*$ [Caprara, MOR'08]. Hence, $\eta$ is a monotonic function.

# Additive function $p$

- Recall that $I$ is a set of $d$D items, $\hat{I} = \mathrm{round}(I)$, and $J$ is the set of shelves obtained by running the greedy algorithm on $\hat{I}$.
- For $i \in I$, define $p(i) = w(i)\eta(h(i))$. Define $p(I) = \sum_{i \in I} p(i)$.
- We will prove that $\mathrm{opt}\big(\mathrm{DLP}(J)\big) - 1 \leq p(I) \leq T_k^{d-1}\mathrm{opt}_{d\mathrm{BP}}(I)$.
- This will complete the proof of the structural theorem.

# Lower-bounding $p(I)$

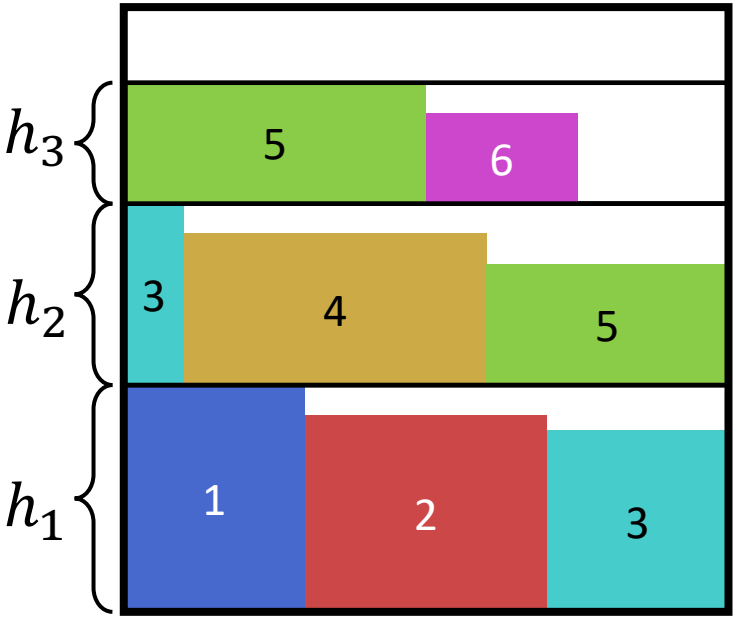



- Let $S_j$ be the set of items in the $j^{\text{th}}$-tallest shelf.
- The items in $S_j$ have height in $[h_{j+1}, h_j]$ ($\because$ of greedy algo).

$$p(I) = \sum_{j=1}^{|J|} \sum_{i \in S_j} p(i) = \sum_{j=1}^{|J|} \sum_{i \in S_j} w(i)\eta(h(i))$$

$$\geq \sum_{j=1}^{|J|-1} \sum_{i \in S_j} w(i) y^*_{j+1} = \sum_{j=1}^{|J|-1} y^*_{j+1}$$

$$= \text{opt}\big(\text{DLP}(J)\big) - y^*_1 \geq \text{opt}\big(\text{DLP}(J)\big) - 1$$



# Weighting functions

- Weighting functions help us get lower bounds on $\mathrm{opt}_{\mathrm{BP}}(I)$.
- $v\colon [0,1] \mapsto [0,1]$ is a weighting function iff $\forall S \subseteq [0,1]$,

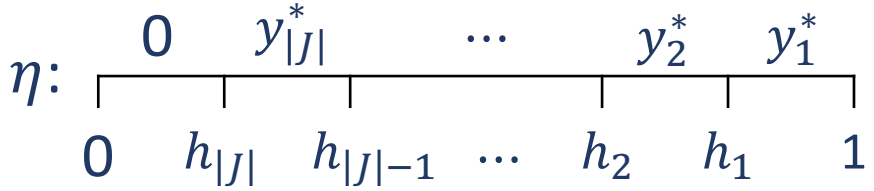
$$\sum_{x \in S} x \leq 1 \Longrightarrow \sum_{x \in S} v(x) \leq 1.$$

- Example: $v(x) = \begin{cases} 1 & x > 1/2 \\ 0 & x \leq 1/2 \end{cases}$.

# Weighting functions that we have seen

Let $H(x) = f_k(x)/T_k$. Then $H$ is a weighting function by defn of $T_k$.

$\eta$ is a weighting function [Caprara, MOR'08]:




- Let $S \subseteq [0,1]$ such that $\sum_{x \in S} x \leq 1$.
- For each $x \in [h_{j+1}, h_j) \cap S$, create an item of size $h_{j+1}$.
- The items have total size at most 1, so they form a configuration.
- In DLP, we have the constraint $\sum_{i \in C} y_i \leq 1$ for each configuration $C$.
- $\forall x \in [h_{j+1}, h_j)$, we have $\eta(x) = y^*_{j+1}$. Hence, $\sum_{x \in S} \eta(x) \leq 1$.

# Weighting function theorem

- Let $I$ be a set of $d$D items that can be packed into a bin.
- For $i \in I$, let $\ell_j(i)$ be the length of $i$ in the $j^{\text{th}}$ dimension for $j \in [d]$.
- Let $g_1, g_2, \ldots, g_d$ be weighting functions.

$$\text{Weighting function Theorem: } \sum_{i \in I} \prod_{j=1}^{d} g_j(\ell_j(i)) \leq \text{opt}_{d\text{BP}}(I).$$

- BCS [FOCS'06] give a proof sketch for $d = 2$.
  My paper has full proof (theorem 2).

# Upper-bounding $p(I)$

Recall the definition of $p(i)$:

$$p(i) = w(i)\eta(h(i)) = \left(\prod_{j=1}^{d-1} f_k(\ell_j(i))\right)\eta(\ell_d(i))$$

Hence, by weighting function theorem, $p(I) \leq T_k^{d-1}\mathrm{opt}_{d\mathrm{BP}}(I)$.

# Summary

$$\hat{I} = \text{round}(I)$$

$$\begin{aligned}
\text{sopt}_\delta(\hat{I}) &\approx \text{opt}_{1\text{BP}}(J) \\
&\approx \text{LP}(J) \\
&\approx \text{DLP}(J) \\
&\leq p(I) + 1 \\
&\leq T_k^{d-1}\text{opt}_{d\text{BP}}(I) + 1
\end{aligned}$$

1. Pack $\hat{I}$ into shelves greedily to get $J$. By linear grouping, we get $\text{sopt}_\delta(\hat{I}) \approx \text{opt}_{1\text{BP}}(J)$.
2. Express 1BP as linear program. $\text{opt}_{1\text{BP}}(J) \approx \text{opt}(\text{LP}(J))$ by KK.
3. Use strong duality.
4. Use optimal solution to DLP to define $\eta$ and $p$. Prove $\text{DLP}(J) \leq p(I) + 1$.
5. Use weighting functions to prove $p(I) \leq T_k^{d-1}\text{opt}_{d\text{BP}}(I)$.

Any questions so far?

# Conclusion

# Conclusion

- We discussed the geometric bin packing problem with rotations and observed that prior work has limitations in terms of the AAR and assumptions about item orientability and bin sizes.
- We describe the multiple-choice bin packing problem, and show that it's a good framework for studying bin packing with item rotations.
- We give algorithms fullh and HGaP, which have AARs $T_k^d$ and $T_k^{d-1}(1+\varepsilon)$, respectively. For $d \geq 3$, our algorithms close the gap in AAR between the rotational and non-rotational versions of $d$BP.

# Open problems in geometric bin packing

- AAR better than 1.405 for 2BP.
- AAR better than $T_\infty^2 \approx 2.860$ for 3BP.
- Better hardness for $d$BP. For $d \geq 2$, lower bound is $\approx 1.00026$. For $d \geq 3$, upper bound is $\approx 1.691^{d-1}$.
- Bridging gap between theory and practice:
  - Practical approximation algorithms.
  - Finding important special cases and getting algorithms for them.
  - Beyond worst-case analysis.

# Practical approximation algorithms

- For non-rotational 3BP, HDH+FirstFitDecreasing runs in $O(n \log n)$ time and has AAR $\frac{11}{9}T_\infty^2 \approx 3.495$.
- For rotational 3BP, fullh runs in $O(n \log n)$ time and has AAR $T_\infty^3 \approx 4.836$.
- There's a significant gap here.
- Upper-bounding First-Fit Decreasing (or some other practical algorithm) in terms of a weighting function can improve the practical bound for rotational $d$BP.

# Thank You

Questions?

arXiv:2011.10963