Integrality gap
for the
bin-packing problem

Gennady Shmonin

Ecole Polytechnique Fédérale de Lausanne, Switzerland

Joint work with

András Sebő

Laboratoire G-SCOP, Grenoble, France

Outline

- Introduction
  - Bin-packing vs. Stock-cutting
- Integer programming formulations
  - Kantorovich' assignment formulation
  - Gilmore–Gomory formulation
  - **Polynomial-size formulations**
- Integrality gaps

The bin-packing problem

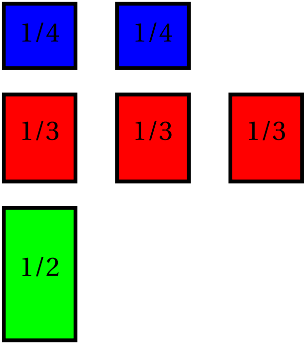
Given n items of sizes $s_1, s_2, \ldots, s_n$

Find min #bins of capacity 1 needed to pack all items

The bin-packing problem

Given n items of sizes $s_1, s_2, \ldots, s_n$

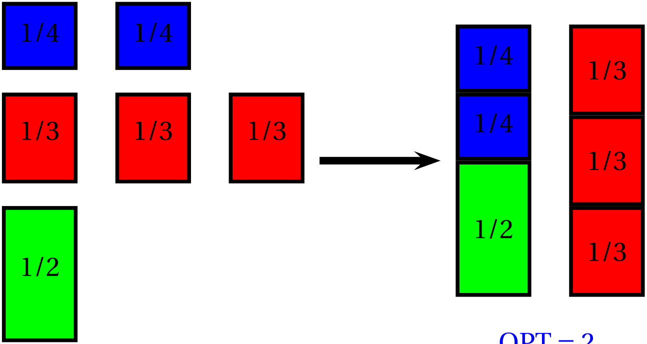
Find min #bins of capacity 1 needed to pack all items



The bin-packing problem

Given n items of sizes $s_1, s_2, \ldots, s_n$

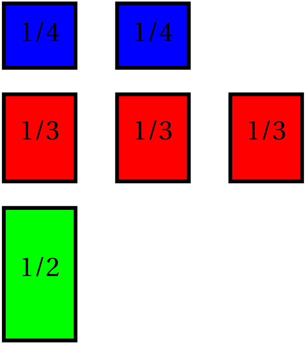
Find min #bins of capacity 1 needed to pack all items



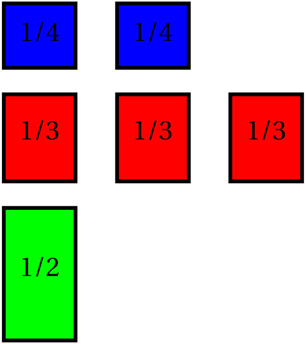
OPT = 2

The stock-cutting problem

The stock-cutting problem



The stock-cutting problem

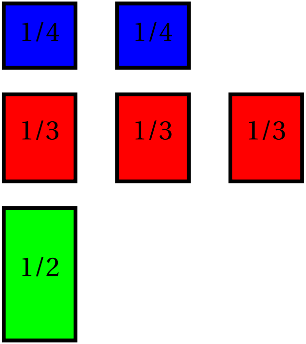


$a_1 = 1/4$, $b_1 = 2$

$a_2 = 1/3$, $b_2 = 3$

$a_3 = 1/2$, $b_3 = 1$

The stock-cutting problem



$a_1 = 1/4$, $b_1 = 2$

$a_2 = 1/3$, $b_2 = 3$

$a_3 = 1/2$, $b_3 = 1$

Given a size-vector $a \in [0,1]^d$ and a multiplicity-vector $b \in \mathbb{Z}_+^d$

Find min #bins of capacity 1 needed to pack all items

Cutting stock vs. Bin packing

In principle, the problems are equivalent

Cutting stock vs. Bin packing

In principle, the problems are equivalent

BUT:

- polynomial-time algorithm for the bin-packing problem is not necessarily polynomial-time for the stock-cutting problem!
- polynomial-size formulation for the bin-packing problem is not necessarily polynomial-size for the stock-cutting problem!

Cutting stock vs. Bin packing

In principle, the problems are equivalent

BUT:

- polynomial-time algorithm for the bin-packing problem is not necessarily polynomial-time for the stock-cutting problem!
- polynomial-size formulation for the bin-packing problem is not necessarily polynomial-size for the stock-cutting problem!

Reason:

- Size of the input for the stock-cutting problem is exponentially small compared to the input for the bin-packing problem

Kantorovich formulation

Items: $s_1, s_2, \ldots, s_n$
Bins: $B_1, B_2, \ldots, B_n$

Variables

- $y_j \in \{0,1\}$: bin B_j is open
- $x_{ij} \in \{0,1\}$: item s_i assigned to bin B_j

Objective

- minimize $\sum_j y_j$ (open as few bins as possible)

Constraints

- $x_{ij} \leqslant y_j$: put items only to open bins
- $\sum_j x_{ij} = 1$: each item is assigned to exactly one bin
- $\sum_i s_i x_{ij} \leqslant 1$: capacity constraints

Kantorovich formulation

Vectors: $a = [a_1, a_2, \ldots, a_d]$ and $b = [b_1, b_2, \ldots, b_d]$
Bins: $B_1, B_2, \ldots, B_n$

Variables

- $y_j \in \{0, 1\}$: bin B_j is open
- $x_{ij} \in \{0, 1, \ldots, b_i\}$: # items of type i assigned to bin B_j

Objective

- minimize $\sum_j y_j$ (open as few bins as possible)

Constraints

- $x_{ij} \leqslant b_i y_j$: put items only to open bins
- $\sum_j x_{ij} = b_i$: exactly b_i items of type i are assigned to bins
- $\sum_i s_i x_{ij} \leqslant 1$: capacity constraints

Kantorovich formulation

Vectors: $a = [a_1, a_2, \ldots, a_d]$ and $b = [b_1, b_2, \ldots, b_d]$
Bins: $B_1, B_2, \ldots, B_n$ $\quad (n = \sum_i b_i)$

Variables

- $y_j \in \{0, 1\}$: bin B_j is open
- $x_{ij} \in \{0, 1, \ldots, b_i\}$: # items of type i assigned to bin B_j

Objective

- minimize $\sum_j y_j$ (open as few bins as possible)

Constraints

- $x_{ij} \leqslant b_i y_j$: put items only to open bins
- $\sum_j x_{ij} = b_i$: exactly b_i items of type i are assigned to bins
- $\sum_i s_i x_{ij} \leqslant 1$: capacity constraints

Kantorovich formulation

Advantages:

- looks quite natural, easy to come up with

Kantorovich formulation

Advantages:

- looks quite natural, easy to come up with

Disadvantages:

- exponential-size for stock-cutting

Kantorovich formulation

Advantages:

- looks quite natural, easy to come up with

Disadvantages:

- exponential-size for stock-cutting
- moreover, even the optimum solution itself for stock-cutting has exponentially many nonzero components

Kantorovich formulation

Advantages:

- looks quite natural, easy to come up with

Disadvantages:

- exponential-size for stock-cutting
- moreover, even the optimum solution itself for stock-cutting has exponentially many nonzero components
- very bad LP relaxation:

  (opt) – (opt of LP relaxation) can be $\approx n/2$

$\Longrightarrow$ absolutely impractical

Gilmore–Gomory formulation

Size-vectors: $a = [a_1, a_2, \dots, a_d]$
Multiplicity-vector: $b = [b_1, b_2, \dots, b_d]$

Gilmore–Gomory formulation

Size-vectors: $a = [a_1, a_2, \ldots, a_d]$
Multiplicity-vector: $b = [b_1, b_2, \ldots, b_d]$

Pattern : *"a way to pack a bin"*

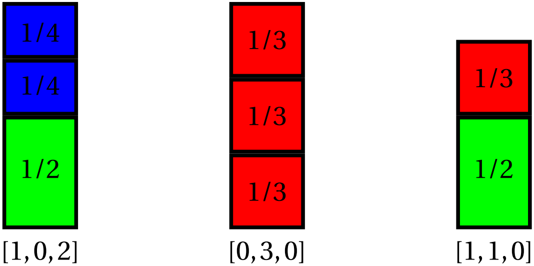
vector $v \in \mathbb{Z}_+^d$ such that $\sum_i a_i v_i \leqslant 1$

Gilmore–Gomory formulation

Size-vectors: $a = [a_1, a_2, \ldots, a_d]$
Multiplicity-vector: $b = [b_1, b_2, \ldots, b_d]$

Pattern : *"a way to pack a bin"*

vector $v \in \mathbb{Z}_+^d$ such that $\sum_i a_i v_i \leqslant 1$



Gilmore–Gomory formulation

Size-vectors: $a = [a_1, a_2, \dots, a_d]$
Multiplicity-vector: $b = [b_1, b_2, \dots, b_d]$
All possible patterns: $v_1, v_2, \dots, v_k$

Variables

- $\lambda_j \in \mathbb{Z}_+$: # pattern v_j to be used in the packing

Objective

- minimize $\sum_j \lambda_j$: total # patterns = total # bins

Constraints

- $\sum_j \lambda_j v_{ij} = b_i$: all items of type i are packed

  equiv. $\sum_j \lambda_j v_j = b$

Gilmore–Gomory formulation

$$\begin{array}{ll} \min & \sum_j \lambda_j \\ & \sum_j \lambda_j v_j = b \\ & \lambda_j \geqslant 0 \text{ integer} \end{array}$$

Disadvantages:

- exponential-size for both bin-packing and stock-cutting (in fact, much larger than that of Kantorovich)

Gilmore–Gomory formulation

$$\begin{array}{ll} \min & \sum_j \lambda_j \\ & \sum_j \lambda_j v_j = b \\ & \lambda_j \geqslant 0 \text{ integer} \end{array}$$

Disadvantages:

- exponential-size for both bin-packing and stock-cutting (in fact, much larger than that of Kantorovich)

Advantages:

- there is an optimum solution with polynomially many nonzero components (Eisenbrand & S. 2006)

Gilmore–Gomory formulation

$$
\begin{array}{ll}
\min & \sum_j \lambda_j \\
 & \sum_j \lambda_j v_j = b \\
 & \lambda_j \geqslant 0 \text{ integer}
\end{array}
$$

Disadvantages:

- exponential-size for both bin-packing and stock-cutting (in fact, much larger than that of Kantorovich)

Advantages:

- there is an optimum solution with polynomially many nonzero components (Eisenbrand & S. 2006)
- very strong LP relaxation

Gilmore–Gomory formulation

$$
\begin{array}{ll}
\min & \sum_j \lambda_j \\
 & \sum_j \lambda_j v_j = b \\
 & \lambda_j \geqslant 0 \text{ integer}
\end{array}
$$

Disadvantages:

- exponential-size for both bin-packing and stock-cutting (in fact, much larger than that of Kantorovich)

Advantages:

- there is an optimum solution with polynomially many nonzero components (Eisenbrand & S. 2006)
- very strong LP relaxation
- column-generation methods help to deal with "exponential size issue"

Gilmore–Gomory formulation

$$\begin{array}{ll} \min & \sum_j \lambda_j \\ & \sum_j \lambda_j v_j = b \\ & \lambda_j \geqslant 0 \text{ integer} \end{array}$$

Disadvantages:

- exponential-size for both bin-packing and stock-cutting (in fact, much larger than that of Kantorovich)

Advantages:

- there is an optimum solution with polynomially many nonzero components (Eisenbrand & S. 2006)
- very strong LP relaxation **???**
- column-generation methods help to deal with "exponential size issue"

Subpattern formulation

(Belov & Weismantel 2004)

Size-vectors: $a = [a_1, a_2, \ldots, a_d]$
Multiplicity-vector: $b = [b_1, b_2, \ldots, b_d]$

Subpattern : $u_{ik} = 2^k e_i$

- polynomially many subpatterns
- each pattern v can be expressed as

$$v = \sum_{i,k} \mu_{ik} u_{ik} \text{ with } \mu_{ik} \in \{0,1\}$$

Idea: rewrite Gilmore–Gomory formulation in terms of subpatterns

$$b = \sum_j \lambda_j v_j \quad \longrightarrow \quad b = \sum_{i,j,k} \lambda_j \mu_{jik} u_{ik} = \sum_{i,j,k} \eta_{jik} u_{ik}$$

Subpattern formulation

(Belov & Weismantel 2004)

Size-vectors: $a = [a_1, a_2, \ldots, a_d]$
Multiplicity-vector: $b = [b_1, b_2, \ldots, b_d]$
Subpatterns: $u_{ik} = 2^k e_i$

$v_1, v_2, \ldots, v_t$: patterns to be used in a solution (unknowns!)

Variables

- $\lambda_j \in \mathbb{Z}_+$: # patterns v_j to be used in the packing
- $\mu_{jik} \in \{0, 1\}$: coefficients to express pattern v_j
- $\eta_{jik} \in \mathbb{Z}_+$: ("switch") $\eta_{jik} = \lambda_j$ if $\mu_{jik} = 1$; $\eta_{jik} = 0$ otherwise

Objective

- minimize $\sum_{j=1}^{t} \lambda_j$

Subpattern formulation

(Belov & Weismantel 2004)

Size-vectors: $a = [a_1, a_2, \dots, a_d]$
Multiplicity-vector: $b = [b_1, b_2, \dots, b_d]$
Subpatterns: $u_{ik} = 2^k e_i$

$v_1, v_2, \dots, v_t$: patterns to be used in a solution (unknowns!)

Constraints

- $\sum_{i,j,k} \eta_{jik} u_{ij} = b$: all items are packed
- $\sum_{i,k} \mu_{jik} a^{\mathrm{T}} u_{ij} \leqslant 1$: capacity constraint
- $\lambda_j \geqslant \eta_{jik}$
- $\eta_{jik} \leqslant M \cdot \mu_{jik}$ (for M sufficiently large)

Another polynomial-size formulation

$$\begin{array}{ll} \min & \sum_j \lambda_j \\ & \sum_j \lambda_j v_j = b \\ & \lambda_j \geqslant 0 \text{ integer} \end{array}$$

Let $\lambda_1, \lambda_2, \ldots, \lambda_t$ be optimum solution, $b = \sum_j \lambda_j v_j$

Only polynomially many (say m nonzeros)

$\Longrightarrow$ for some pattern v_j the coefficient $\lambda_j \geqslant \mathrm{OPT}/m$

$\Longrightarrow$ $b = \frac{\mathrm{OPT}}{m} v + b'$ (where v is unknown)

Inductively, we obtain

$$b = \sum_j \lambda_j^* v_j,$$

where $v_j \in \mathbb{Z}_+^d$'s are unknown patterns and λ_j^* are numbers

Integrality gap

Size-vectors: $a = [a_1, a_2, \ldots, a_d]$
Multiplicity-vector: $b = [b_1, b_2, \ldots, b_d]$
All possible patterns: $v_1, v_2, \ldots, v_k$

Gilmore–Gomory formulation

$$\begin{array}{ll} \min & \sum_j \lambda_j \\ & \sum_j \lambda_j v_j = b \\ & \lambda_j \geqslant 0 \text{ integer} \end{array}$$

$\mathrm{OPT}(a, b)$: optimum value
$\mathrm{LIN}(a, b)$: optimum value of LP relaxation
$\mathrm{SIZE}(a, b) = a^{\mathrm{T}} b$: total size

Integrality gap

Size-vectors: $a = [a_1, a_2, \dots, a_d]$
Multiplicity-vector: $b = [b_1, b_2, \dots, b_d]$
All possible patterns: $v_1, v_2, \dots, v_k$

$$\begin{array}{ll} \min & \sum_j \lambda_j \\ & \sum_j \lambda_j v_j = b \\ & \lambda_j \geqslant 0 \text{ integer} \end{array}$$

- vast majority of the instances have "integer round-up property"

$$\mathrm{OPT}(a,b) = \lceil \mathrm{LIN}(a,b) \rceil$$

- no instance is known to violate "modified integer round-up property"

$$\mathrm{OPT}(a,b) \leqslant \lceil \mathrm{LIN}(a,b) \rceil + 1$$

Integrality gap

Conjecture [Scheithauer & Terno 1996]

For all $a \in [0,1]^d$ and $b \in \mathbb{Z}_+^d$,

$$\mathrm{OPT}(a,b) \leqslant \lceil \mathrm{LIN}(a,b) \rceil + 1$$

Integrality gap

Conjecture [Scheithauer & Terno 1996]

For all $a \in [0,1]^d$ and $b \in \mathbb{Z}_+^d$,

$$\mathrm{OPT}(a,b) \leqslant \lceil \mathrm{LIN}(a,b) \rceil + 1$$

Theorem [Scheithauer & Terno 1996]

For all $a \in [0,1]^6$ and $b \in \mathbb{Z}_+^6$,

$$\mathrm{OPT}(a,b) \leqslant \lceil \mathrm{LIN}(a,b) \rceil + 1$$

Integrality gap

Conjecture [Scheithauer & Terno 1996]

For all $a \in [0,1]^d$ and $b \in \mathbb{Z}_+^d$,

$$\mathrm{OPT}(a,b) \leqslant \lceil \mathrm{LIN}(a,b) \rceil + 1$$

Theorem [Scheithauer & Terno 1996]

For all $a \in [0,1]^6$ and $b \in \mathbb{Z}_+^6$,

$$\mathrm{OPT}(a,b) \leqslant \lceil \mathrm{LIN}(a,b) \rceil + 1$$

Theorem [Karmarkar & Karp 1982]

For all $a \in [0,1]^d$ and $b \in \mathbb{Z}_+^d$,

$$\mathrm{OPT}(a,b) \leqslant \lceil \mathrm{LIN}(a,b) \rceil + O(\log^2 d)$$

Integrality gap

Theorem

For all $a \in [0,1]^7$ and $b \in \mathbb{Z}_+^7$,

$$\mathrm{OPT}(a,b) \leqslant \lceil \mathrm{LIN}(a,b) \rceil + 1$$

Sketch of the proof

Residual instances

There is an optimum solution of LP relaxation with at most d nonzero components:

$$\mathrm{LIN}(a,b) = \sum_{i=1}^{d} \lambda_i \qquad \text{and} \qquad b = \sum_{i=1}^{d} \lambda_i v_i$$

Residual instance : defined a and $b' = \sum_{i=1}^{d} \{\lambda_i\} v_i$

$\mathrm{OPT}(a,b) \leqslant \mathrm{OPT}(a,b') + \sum_{i=1}^{d} \lfloor \lambda_i \rfloor$

$\mathrm{LIN}(a,b) = \mathrm{LIN}(a,b') + \sum_{i=1}^{d} \lfloor \lambda_i \rfloor$

$$\Longrightarrow \quad \boxed{\mathrm{OPT}(a,b') - \lceil \mathrm{LIN}(a,b') \rceil \geqslant \mathrm{OPT}(a,b) - \lceil \mathrm{LIN}(a,b) \rceil}$$

Sketch of the proof

Small items

Suppose $a_1 < \frac{1}{\mathrm{SIZE}(a,b)}$

Consider instance given by
size-vector $a' = [a_2, a_3, \ldots, a_d]$
multiplicity-vector $b' = [b_2, b_3, \ldots, b_d]$

Algorithm

(1) Find $\mathrm{OPT}(a', b')$

(2) Pack items of type 1 greedily

$\Longrightarrow$ either $\boxed{\mathrm{OPT}(a,b) - \lceil \mathrm{LIN}(a,b) \rceil \leqslant \mathrm{OPT}(a',b') - \lceil \mathrm{LIN}(a',b') \rceil}$

or $\boxed{\mathrm{OPT}(a,b) - \lceil \mathrm{LIN}(a,b) \rceil \leqslant 1}$

Sketch of the proof

Corollary

For all $a \in [0,1]^d$ and $b \in \mathbb{Z}_+^d$,

if $1/a_i \in \mathbb{Z}$ $(i = 1, 2, \ldots, d)$, then

$$\mathrm{OPT}(a, b) \leqslant \lceil \mathrm{LIN}(a, b) \rceil + 1$$

Sketch of the proof

Corollary

For all $a \in [0,1]^d$ and $b \in \mathbb{Z}_+^d$,

if $1/a_i \in \mathbb{Z}$ $(i = 1, 2, \ldots, d)$, then

$$\mathrm{OPT}(a,b) \leqslant \lceil \mathrm{LIN}(a,b) \rceil + 1$$

Explore the **minimal counter-example**:

- w.l.o.g., residual instance $\Longrightarrow$ $\mathrm{OPT}(a,b) \leqslant d$
- w.l.o.g., no small items $\Longrightarrow$ $a_i > \frac{1}{\mathrm{SIZE}(a,b)} \geqslant \frac{1}{\mathrm{OPT}(a,b)} \geqslant \frac{1}{d}$

Sketch of the proof

Corollary

For all $a \in [0,1]^d$ and $b \in \mathbb{Z}_+^d$,

if $1/a_i \in \mathbb{Z}$ $(i = 1, 2, \ldots, d)$, then

$$\mathrm{OPT}(a,b) \leqslant \lceil \mathrm{LIN}(a,b) \rceil + 1$$

Explore the **minimal counter-example**:

- w.l.o.g., residual instance $\implies$ $\mathrm{OPT}(a,b) \leqslant d$
- w.l.o.g., no small items $\implies$ $a_i > \frac{1}{\mathrm{SIZE}(a,b)} \geqslant \frac{1}{\mathrm{OPT}(a,b)} \geqslant \frac{1}{d}$

$$1, \tfrac{1}{2}, \tfrac{1}{3}, \ldots, \tfrac{1}{d} \ldots \text{ oops}$$

Sketch of the proof

Duality

Size-vectors: $a = [a_1, a_2, \dots, a_d]$
Multiplicity-vector: $b = [b_1, b_2, \dots, b_d]$
All possible patterns: $v_1, v_2, \dots, v_k$

$$\begin{array}{ll} \min & \sum_j \lambda_j \\ & \sum_j \lambda_j v_j = b \\ & \lambda_j \geqslant 0 \text{ integer} \end{array}$$

Optimum solution: $b = u_1 + u_2 + \dots + u_{m+1}$

Sketch of the proof

Duality

Size-vectors: $a = [a_1, a_2, \ldots, a_d]$
Multiplicity-vector: $b = [b_1, b_2, \ldots, b_d]$
All possible patterns: $v_1, v_2, \ldots, v_k$

$$\begin{array}{ll} \min & \sum_j \lambda_j \\ & \sum_j \lambda_j v_j = b \\ & \lambda_j \geqslant 0 \text{ integer} \end{array}$$

Optimum solution: $b = u_1 + u_2 + \ldots + u_{m+1}$

Assign **weights** $w_1, w_2, \ldots, w_d$

- $\boxed{w^{\mathrm{T}} v \leqslant 1}$ whenever $a^{\mathrm{T}} v \leqslant 1$

  $\implies$ *"patterns remain patterns"*

  $\implies \mathrm{LIN}(w, b) \geqslant \mathrm{LIN}(a, b)$
- $\boxed{w^{\mathrm{T}} b > m - 1}$

Sketch of the proof

Duality

Size-vectors: $a = [a_1, a_2, \ldots, a_d]$
Multiplicity-vector: $b = [b_1, b_2, \ldots, b_d]$
All possible patterns: $v_1, v_2, \ldots, v_k$

$$\begin{array}{ll} \min & \sum_j \lambda_j \\ & \sum_j \lambda_j v_j = b \\ & \lambda_j \geqslant 0 \text{ integer} \end{array}$$

Optimum solution: $b = u_1 + u_2 + \ldots + u_{m+1}$

Assign **weights** $w_1, w_2, \ldots, w_d$

- $\boxed{w^{\mathrm{T}} v \leqslant 1}$ whenever $a^{\mathrm{T}} v \leqslant 1$

  $\Longrightarrow$ *"patterns remain patterns"*

  $\Longrightarrow$ $\mathrm{LIN}(w, b) \geqslant \mathrm{LIN}(a, b)$

- $\boxed{w^{\mathrm{T}} b > m - 1}$ $\Longrightarrow$ not a counter-example!

Sketch of the proof

Case $d \leqslant 7$

Sketch of the proof

Case $d \leqslant 7$

By **Residual instances**: need to check $\lceil \mathrm{LIN}(a,b) \rceil = 1,2,3,4,5,6,7$

Sketch of the proof

Case $d \leqslant 7$

By **Residual instances**: need to check $\lceil \mathrm{LIN}(a, b) \rceil = 1, 2, 3, 4, 5, 6, 7$

Suppose $\mathrm{LIN}(a, b) = 4$

Sketch of the proof

Case $d \leqslant 7$

By **Residual instances**: need to check $\lceil \mathrm{LIN}(a,b) \rceil = 1,2,3,4,5,6,7$

Suppose $\mathrm{LIN}(a,b) = 4$

By **Small items**: all items are bigger than $1/3$

Sketch of the proof

Case $d \leqslant 7$

By **Residual instances**: need to check $\lceil \mathrm{LIN}(a, b) \rceil = 1, 2, 3, 4, 5, 6, 7$

Suppose $\mathrm{LIN}(a, b) = 4$

By **Small items**: all items are bigger than $1/3$

Optimum packing: $b = u_1 + u_2 + u_3 + u_4 + R$

- either $\|u_i\|_1 = 1$ or $\|u_i\|_1 = 2$
- $\|R\|_1 = 3$ and R contains the smallest item

Sketch of the proof

Case $d \leqslant 7$

By **Residual instances**: need to check $\lceil \mathrm{LIN}(a, b) \rceil = 1, 2, 3, 4, 5, 6, 7$

Suppose $\mathrm{LIN}(a, b) = 4$

By **Small items**: all items are bigger than $1/3$

Optimum packing: $b = u_1 + u_2 + u_3 + u_4 + R$

- either $\|u_i\|_1 = 1$ or $\|u_i\|_1 = 2$
- $\|R\|_1 = 3$ and R contains the smallest item

Weights:

1 for items in 1-bins

$1/2$ for items in 2-bins and largest item in R

Open problems

- **Polynomial-time algorithm** when d is fixed
  - $d = 2$: McCormick, Smallwood, Spieksma (2001)
  - $d = 3$: **open**
- **Integrality gap** for $d \geqslant 8$?
- **Approximation** "+1"-algorithm?

$$\mathscr{A}(a, b) \leqslant \mathrm{OPT}(a, b) + 1$$

Thank You for Your attention!