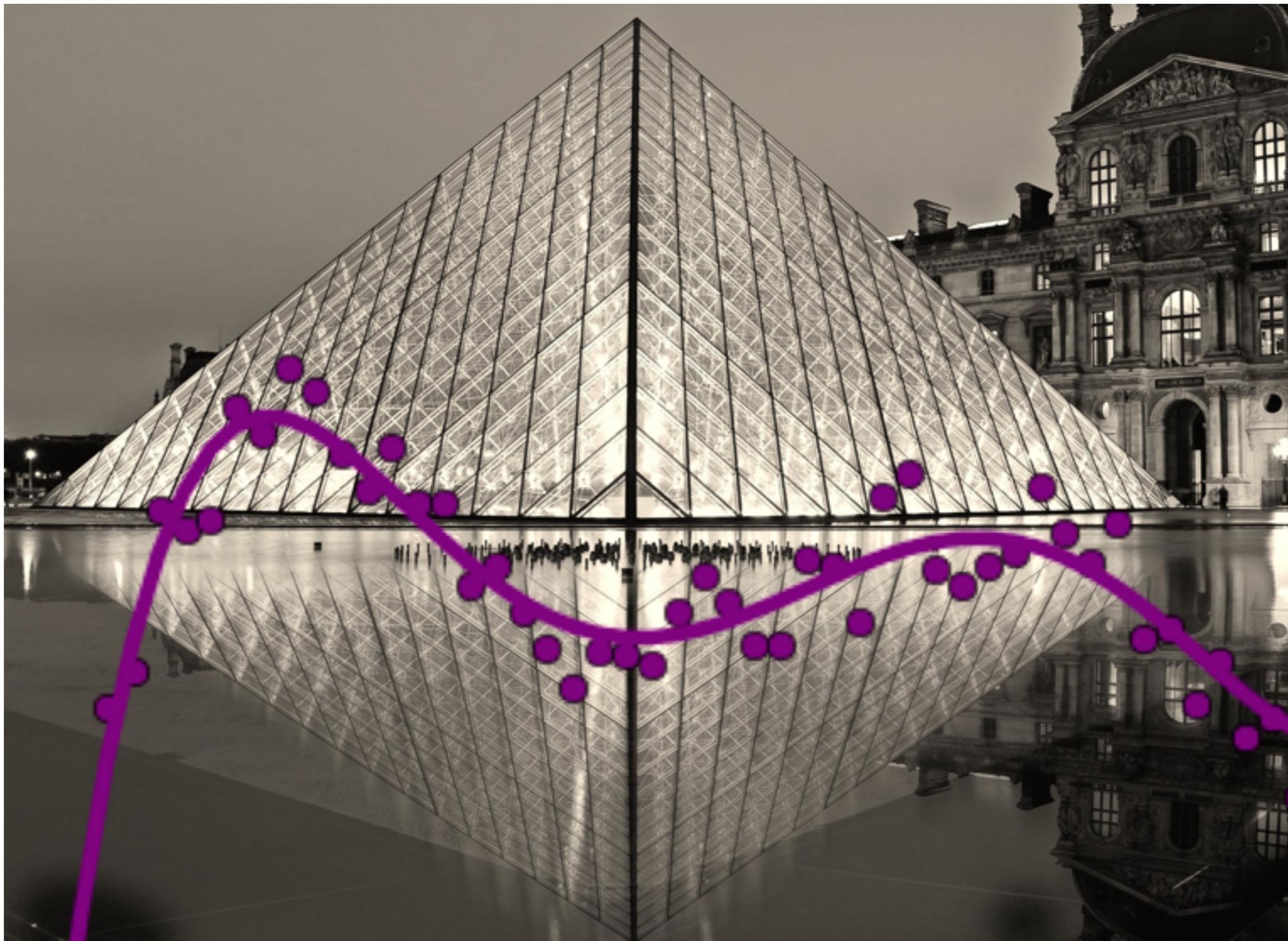


Approximation Algorithms



Combinatorial optimization:

Scheduling classes

Planning delivery routes for trucks

“
Most are NP-hard

What's that?



| Lundi | Mardi | Mercredi | Jeudi | Vendredi |
|---|---|---|---|--|
| TD proba 1 Systèmes Dynamiques Topo 1 | Algèbre 1 Proba 2 Logique | Topo 1 Algèbre 2 | TD Topo 1 EDP cours TD Algèbre 1 TD EDP | Structures et Algorithmes Aléatoires (A Bouillard) 8h30-12h15 Cours et TD Salle R Début : |
| Langages de programmation et compilation (JC Filliatre) 13h15-15h15 cours salle W Début : 28 sept | TD Algèbre 1 TD proba 2 Système Digital (J. Vuillemin) 13h15- 17h cours TD (peut-être2) | TD Statistiques Algèbre 1 Statistiques Proba 1 | Langages formels (D. Vergnaud) 13h15-17h cours TD | Algorithmique et programmation (C Mathieu) 13h15-15h15 cours salle UV Début : 15h15-17h00 |
| 15h15-17h00 TD 1 compil salle Info 4 TD 2 Algo salle W | Algèbre 2 salle UV Début : 29-sept | TD Topo 1 Modelisation et Simulation Numérique Thé du DMA | Proba 1 TD Systèmes Dynamiques GL | TD 1 Algo salle W TD 2 Compil salle Info 4 TD Algèbre 2 |
| GT TD Logique | | Séminaires DI 17h00 -19h00 salle Henri Cartan | | |

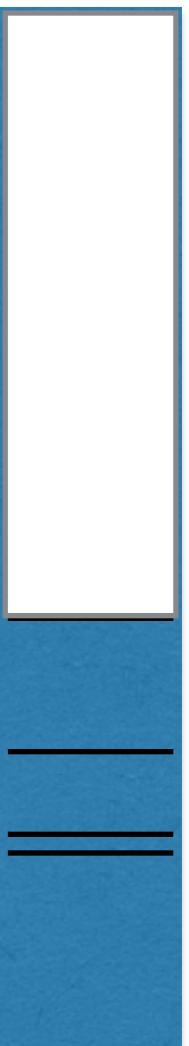
Dealing with NP-hard problems

- Give up?
- Roll up our sleeves?
- Try something, hoping for luck?
- Do a rough but good enough job

In polynomial time, we find a solution whose value is provably within a "small" factor of the optimal solution

Designing approximation algorithms

- Real-life problems are too complex
- Study idealized problems
- Theorems: new algorithmic and structural insights



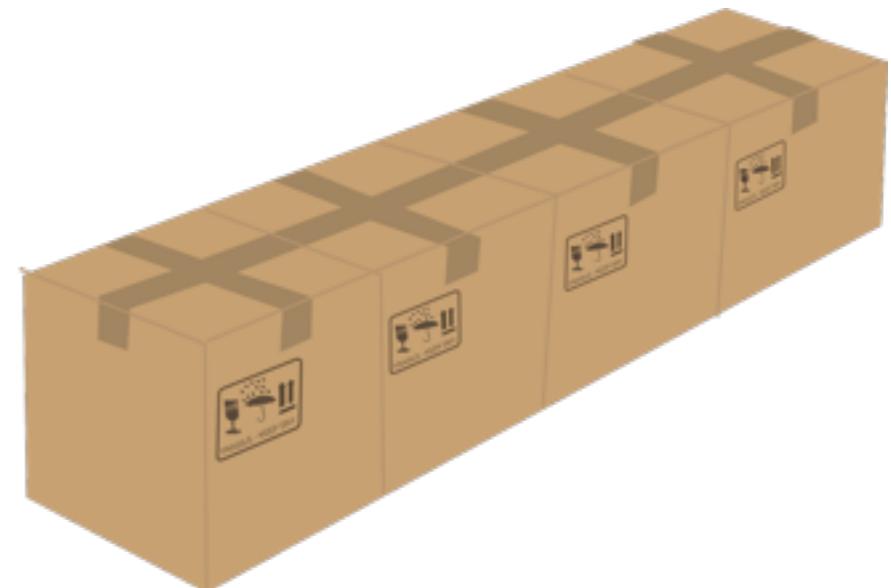
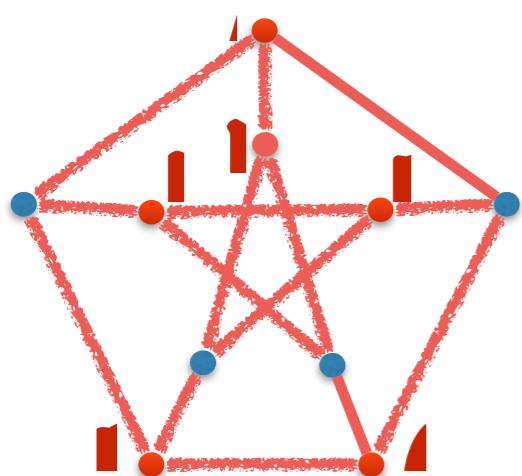
The interface with real life



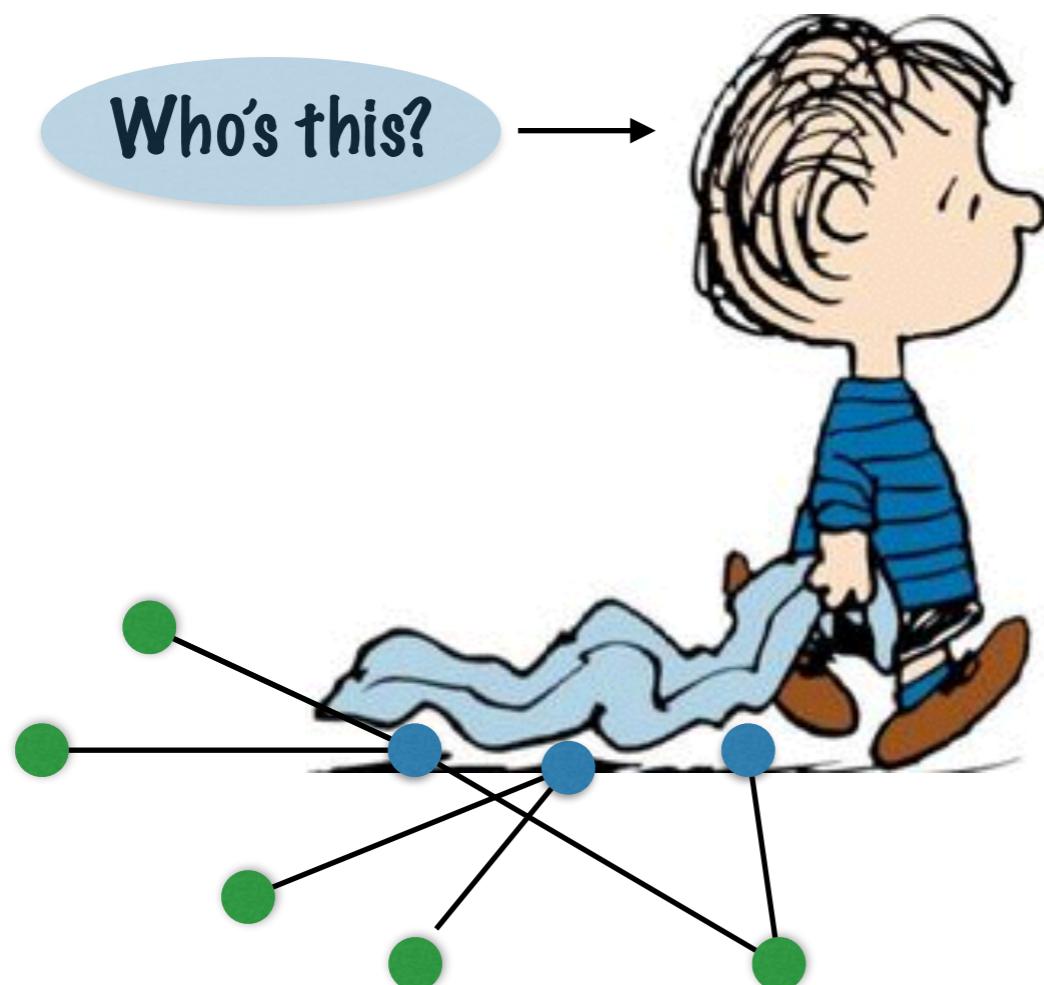
A course with two parts

Approximation algorithms, Part I

- Vertex cover
- Knapsack
- Bin packing
- Set cover
- Multiway cut



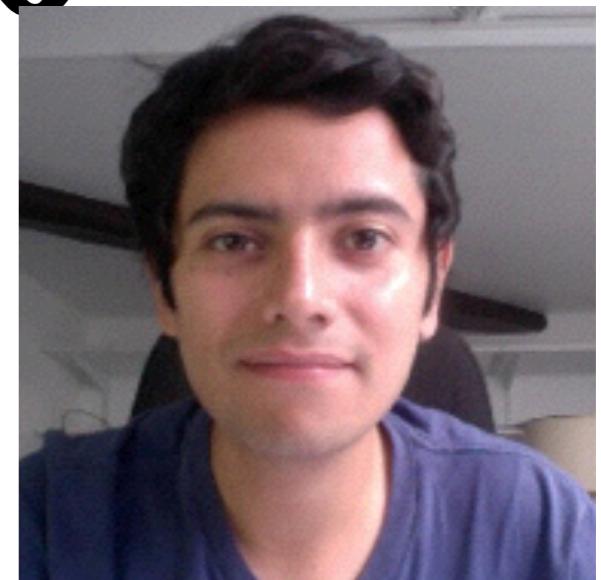
Approximation algorithms, vertex cover, and linear programming



$$\begin{aligned} & \min c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11} x_1 + a_{12} x_2 + \cdots + a_{1n} x_n & \geq b_1 \\ a_{21} x_1 + a_{22} x_2 + \cdots + a_{2n} x_n & \geq b_2 \\ \dots \\ a_{m1} x_1 + a_{m2} x_2 + \cdots + a_{mn} x_n & \geq b_m \end{array} \right. \\ & \forall i : 0 \leq x_i \leq 1 \\ & \forall i : x_i \text{ real number} \end{aligned}$$

Teaching staff behind the scenes

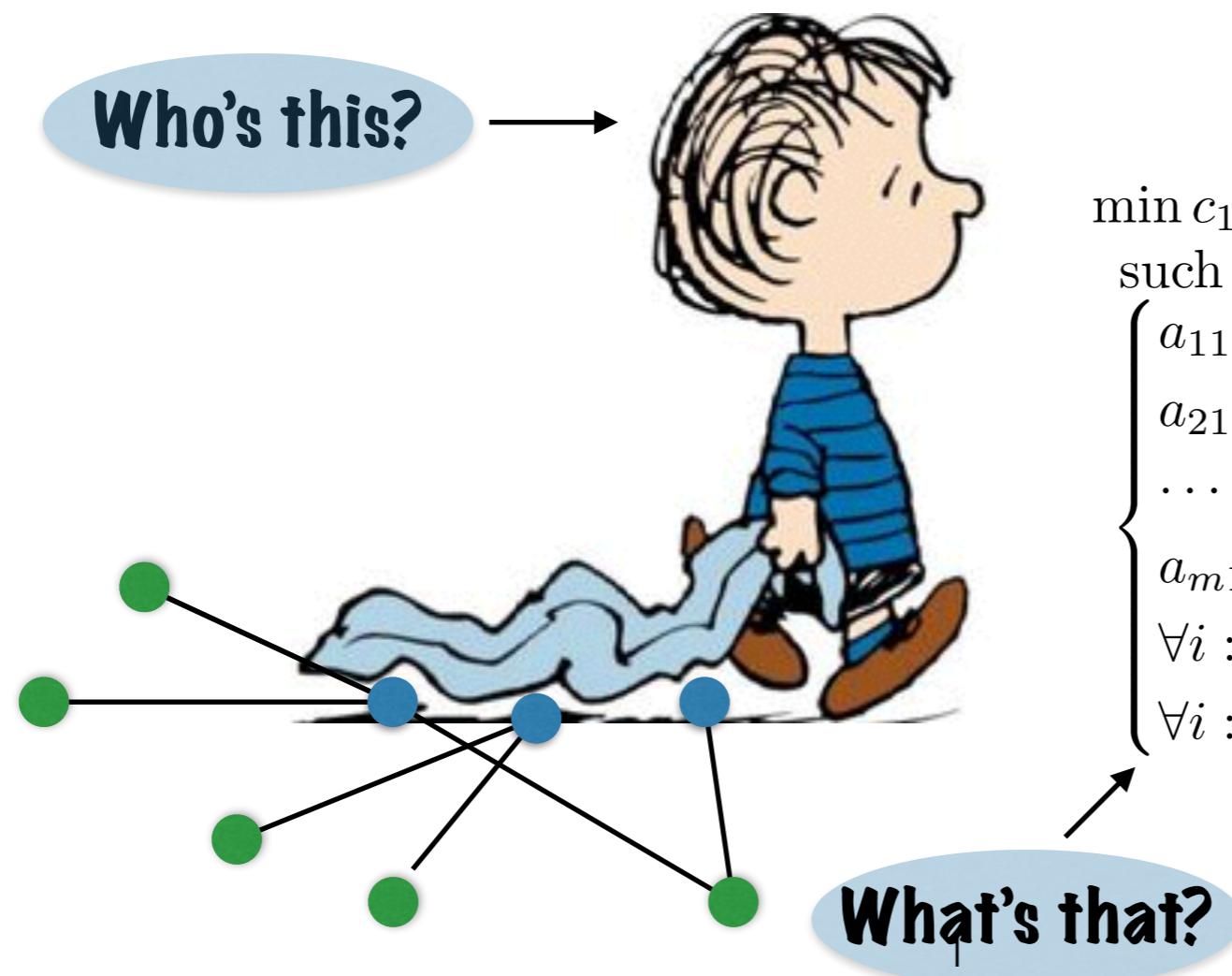
- **Vincent Cohen-Addad**
- **Frederik Mallmann-Trenn**
- **Victor Verdugo**



Film director: Nordine Méziane

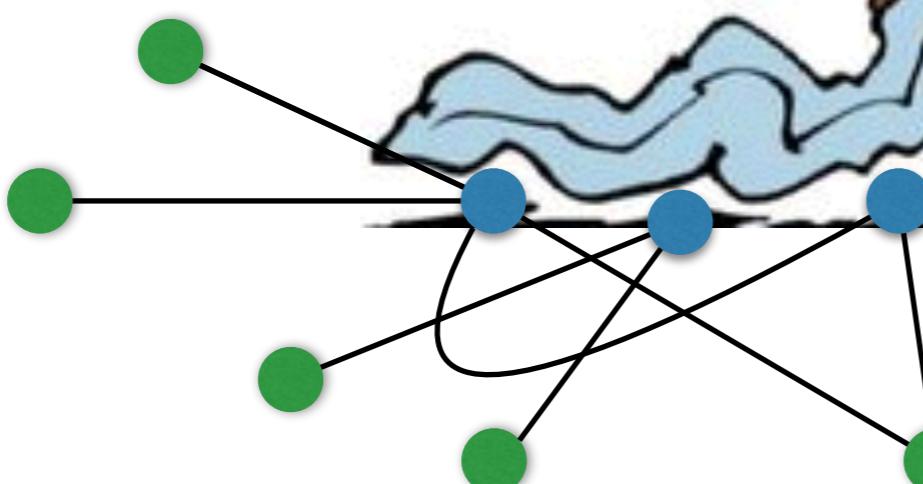
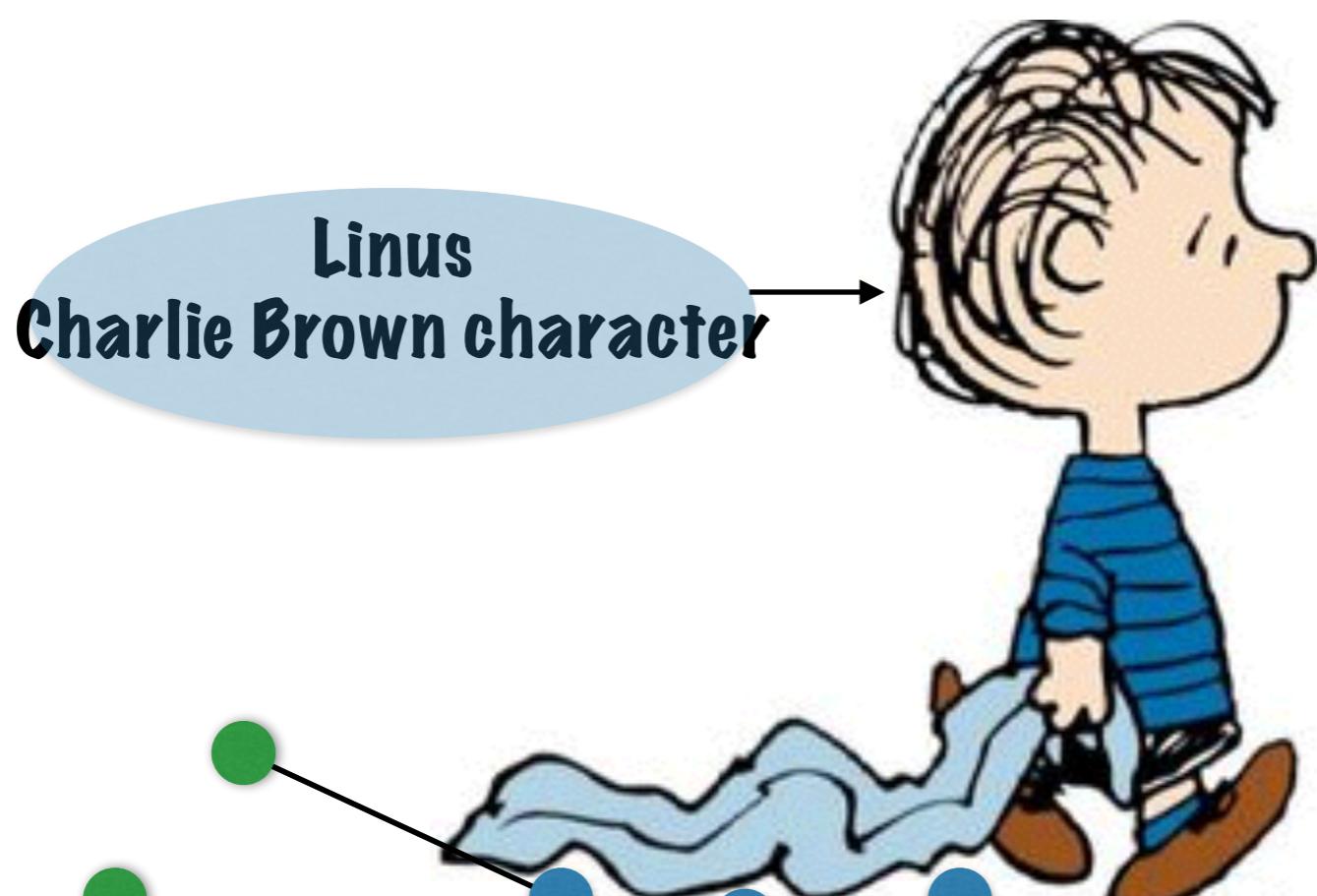
Cameraman: Jovanny Parvedy

Approximation algorithms, vertex cover, and linear programming



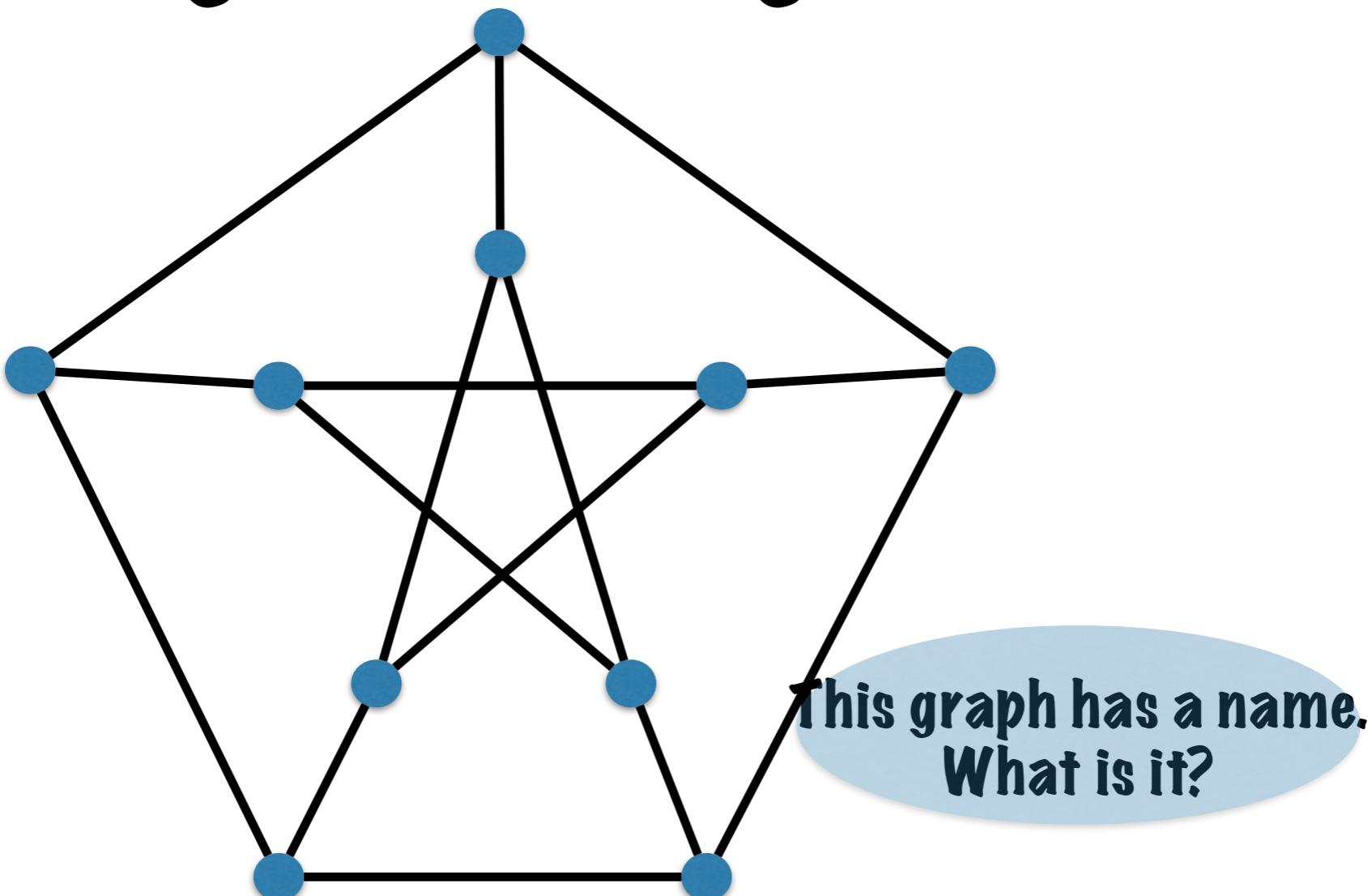
$$\begin{aligned} & \min c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \\ \forall i : 0 \leq x_i \leq 1 \\ \forall i : x_i \text{ real number} \end{array} \right. \end{aligned}$$

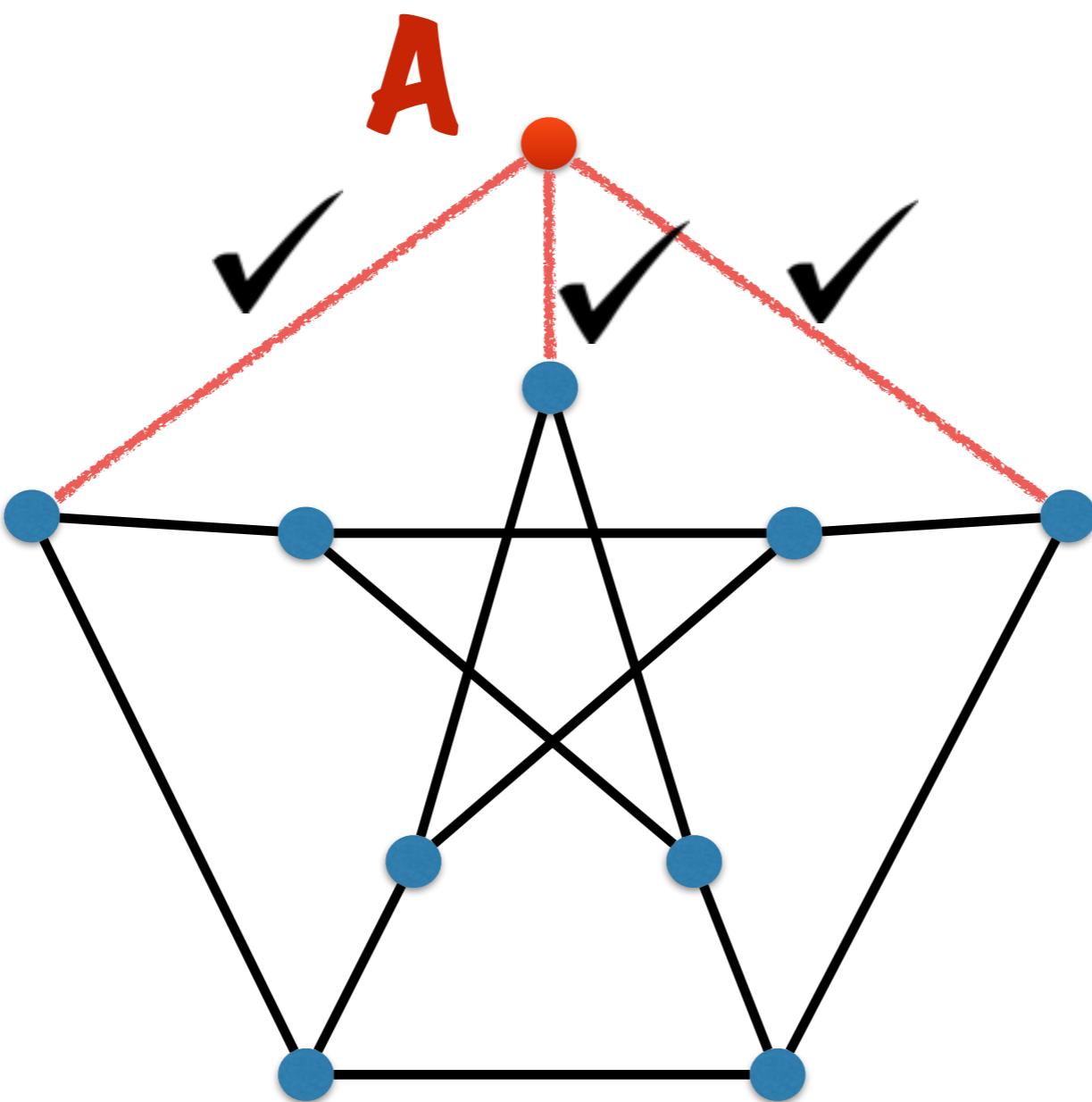
Vertex Cover

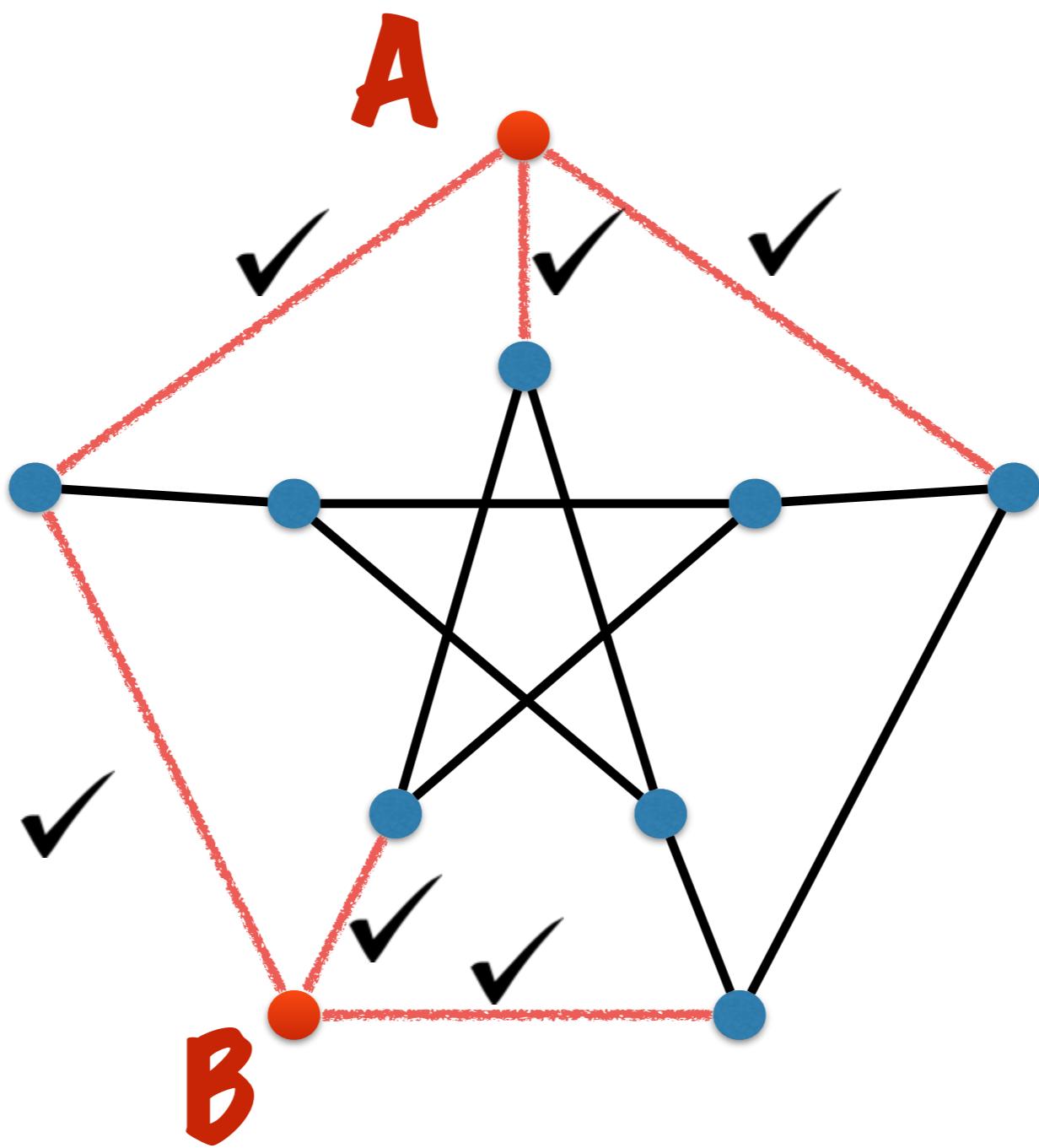


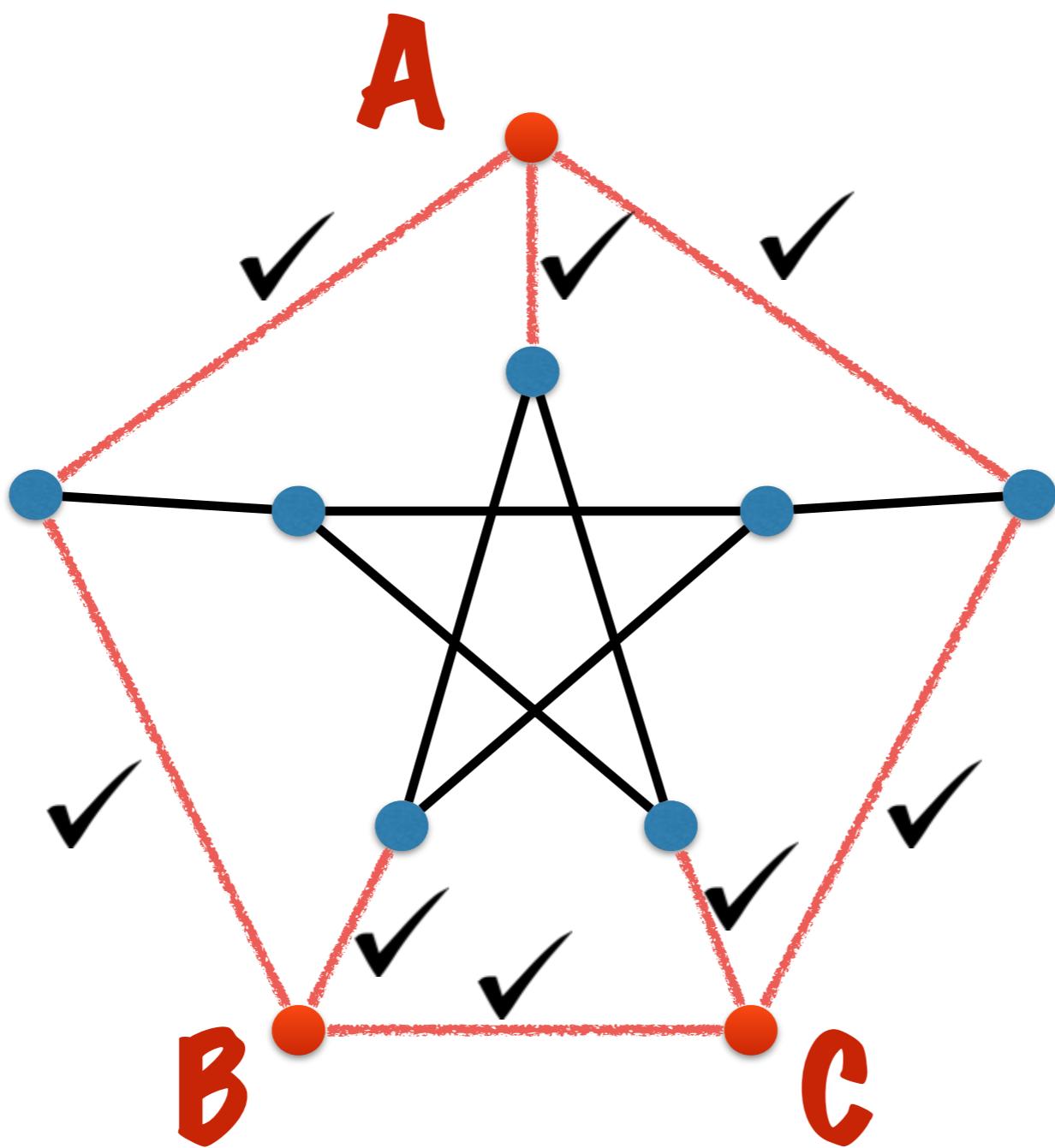
Vertex cover

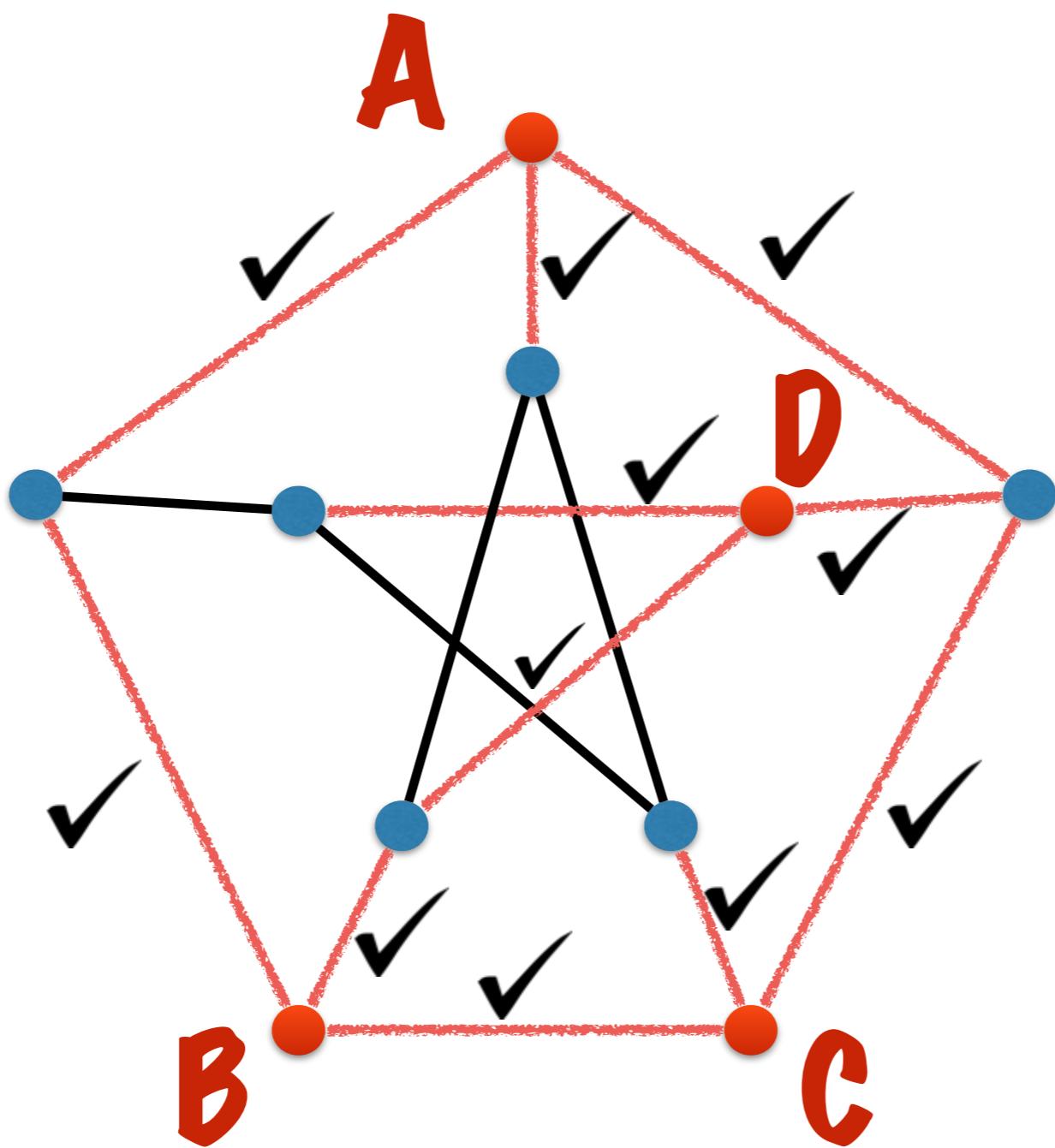
Given graph with vertex weights,
cover edges with lightest vertices

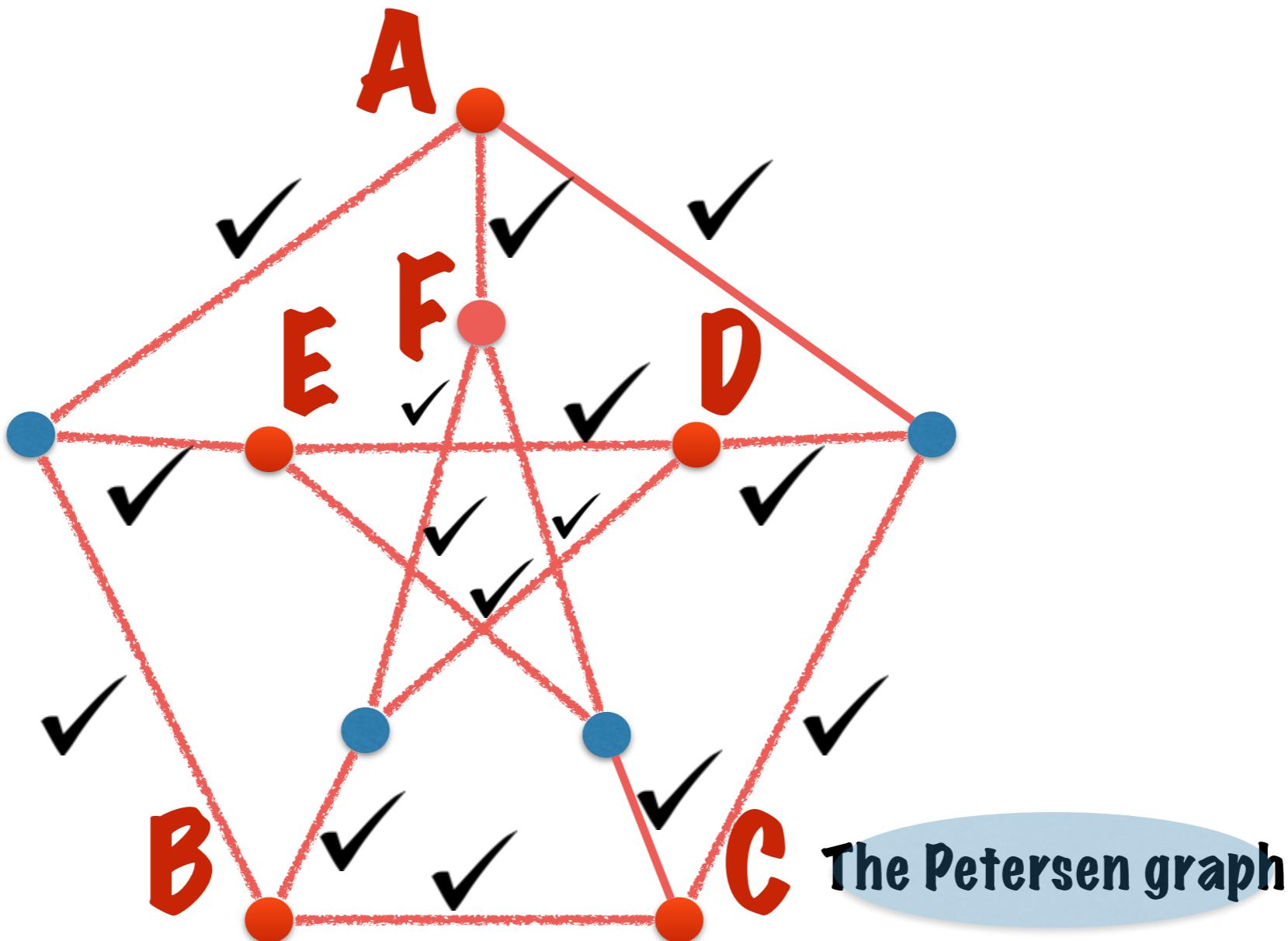




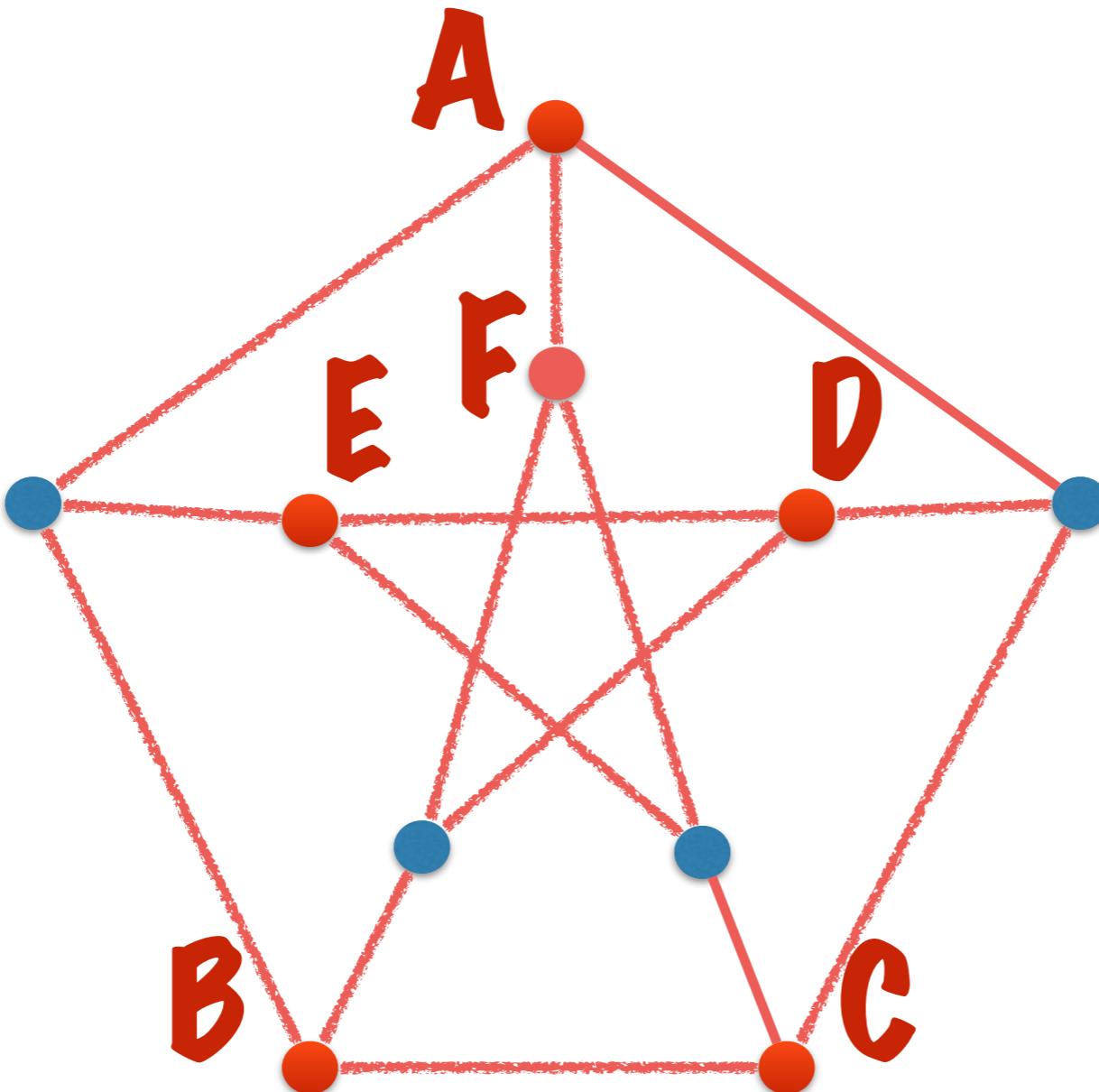






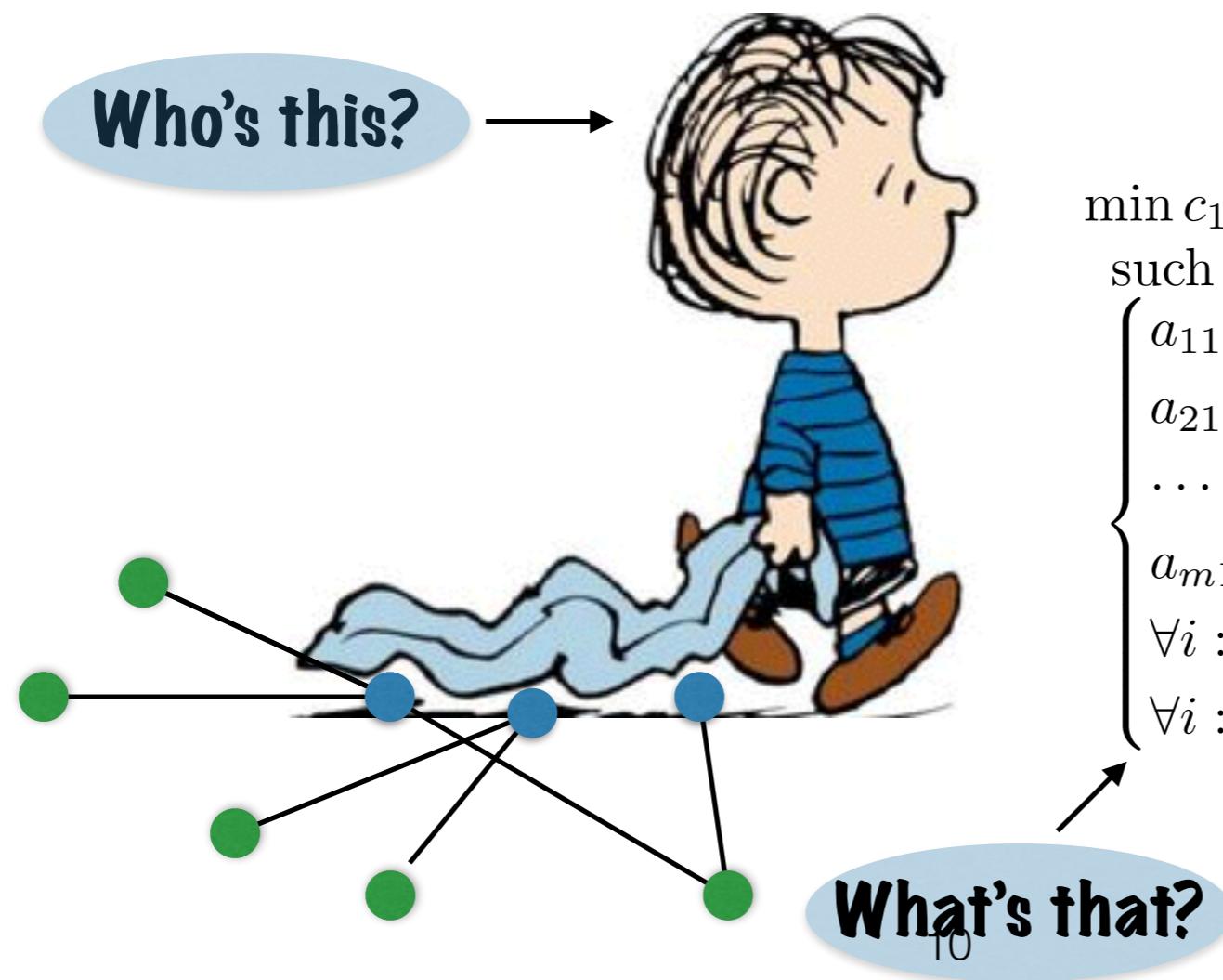


Vertex weights: 1.
Cover edges with 6 vertices.
Optimal : cannot cover with 5



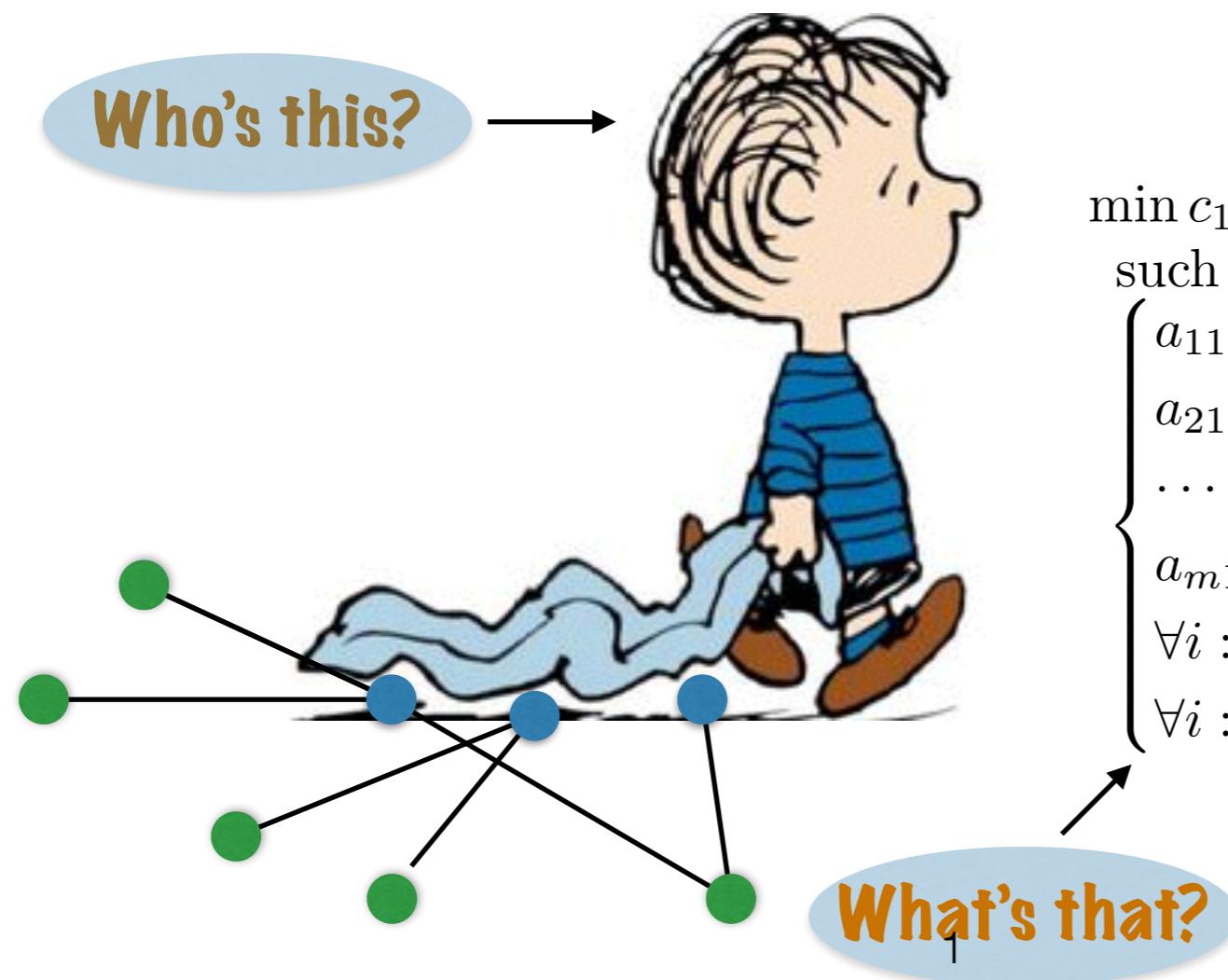
Given graph with vertex weights,
cover edges with lightest vertices

Approximation algorithms, vertex cover, and linear programming



$$\begin{aligned} & \min c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \end{array} \right. \\ & \forall i : 0 \leq x_i \leq 1 \\ & \forall i : x_i \text{ real number} \end{aligned}$$

Approximation algorithms, vertex cover, and linear programming

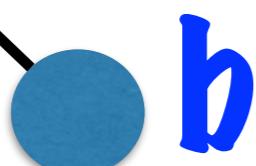
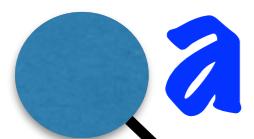


$$\begin{aligned} & \min c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \\ \forall i : 0 \leq x_i \leq 1 \\ \forall i : x_i \text{ real number} \end{array} \right. \end{aligned}$$

Variables

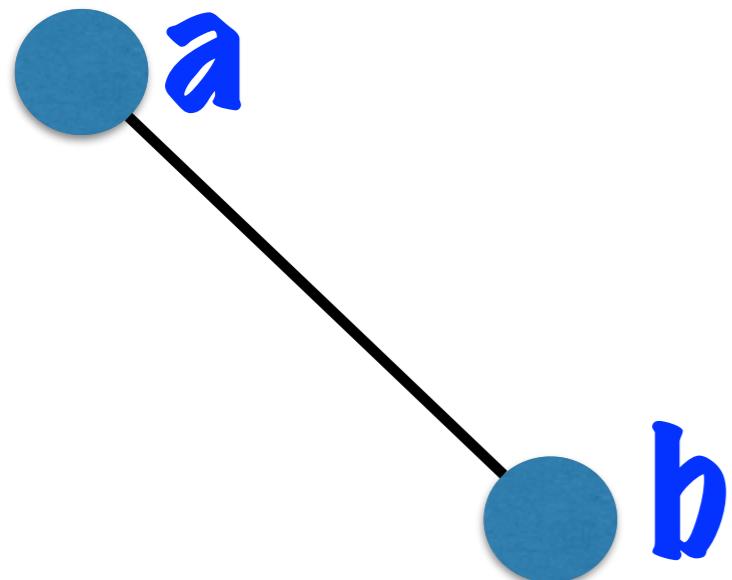
$\{a, b\} \in E :$

a or b must be in cover



$$x_a = \begin{cases} 1 & \text{if } a \text{ in cover} \\ 0 & \text{otherwise} \end{cases}$$

Constraints



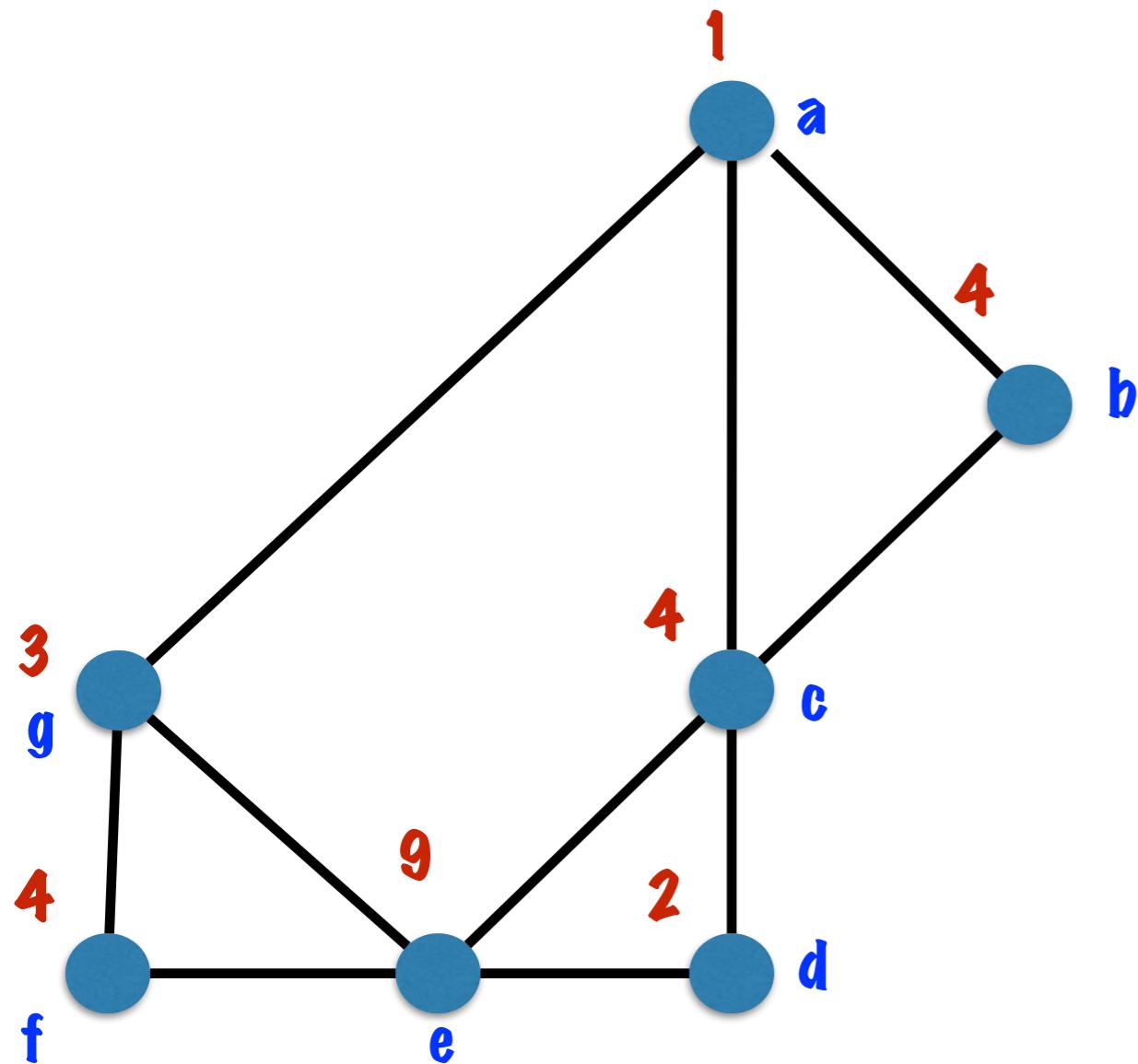
| x_a | x_b | edge covered? | $x_a + x_b$ |
|-------|-------|---------------|-------------|
| 0 | 0 | no | 0 |
| 1 | 0 | yes | 1 |
| 0 | 1 | yes | 1 |
| 1 | 1 | yes | 2 |

$\{a, b\}$ covered



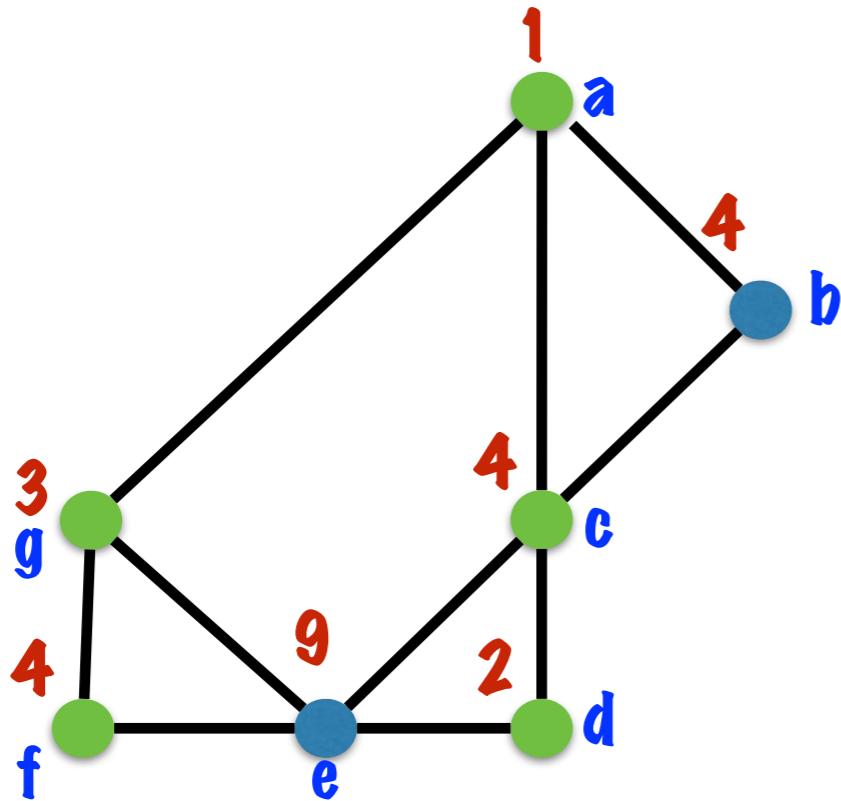
$$x_a + x_b \geq 1$$

Objective



$$\left\{ \begin{array}{l} x_a + x_b \geq 1 \\ x_a + x_c \geq 1 \\ x_a + x_g \geq 1 \\ x_b + x_c \geq 1 \\ x_c + x_d \geq 1 \\ x_c + x_e \geq 1 \\ x_d + x_e \geq 1 \\ x_e + x_f \geq 1 \\ x_e + x_g \geq 1 \\ x_f + x_g \geq 1 \end{array} \right.$$

$$\min x_a + 4x_b + 4x_c + 2x_d + 9x_e + 4x_f + 3x_g$$



$\{a, c, d, f, g\}$ vertex cover

$$\left\{ \begin{array}{l} x_a + x_b \geq 1 \\ x_a + x_c \geq 1 \\ x_a + x_g \geq 1 \\ x_b + x_c \geq 1 \\ x_c + x_d \geq 1 \\ x_c + x_e \geq 1 \\ x_d + x_e \geq 1 \\ x_e + x_f \geq 1 \\ x_e + x_g \geq 1 \\ x_f + x_g \geq 1 \end{array} \right.$$

$$\min x_a + 4x_b + 4x_c + 2x_d + 9x_e + 4x_f + 3x_g$$

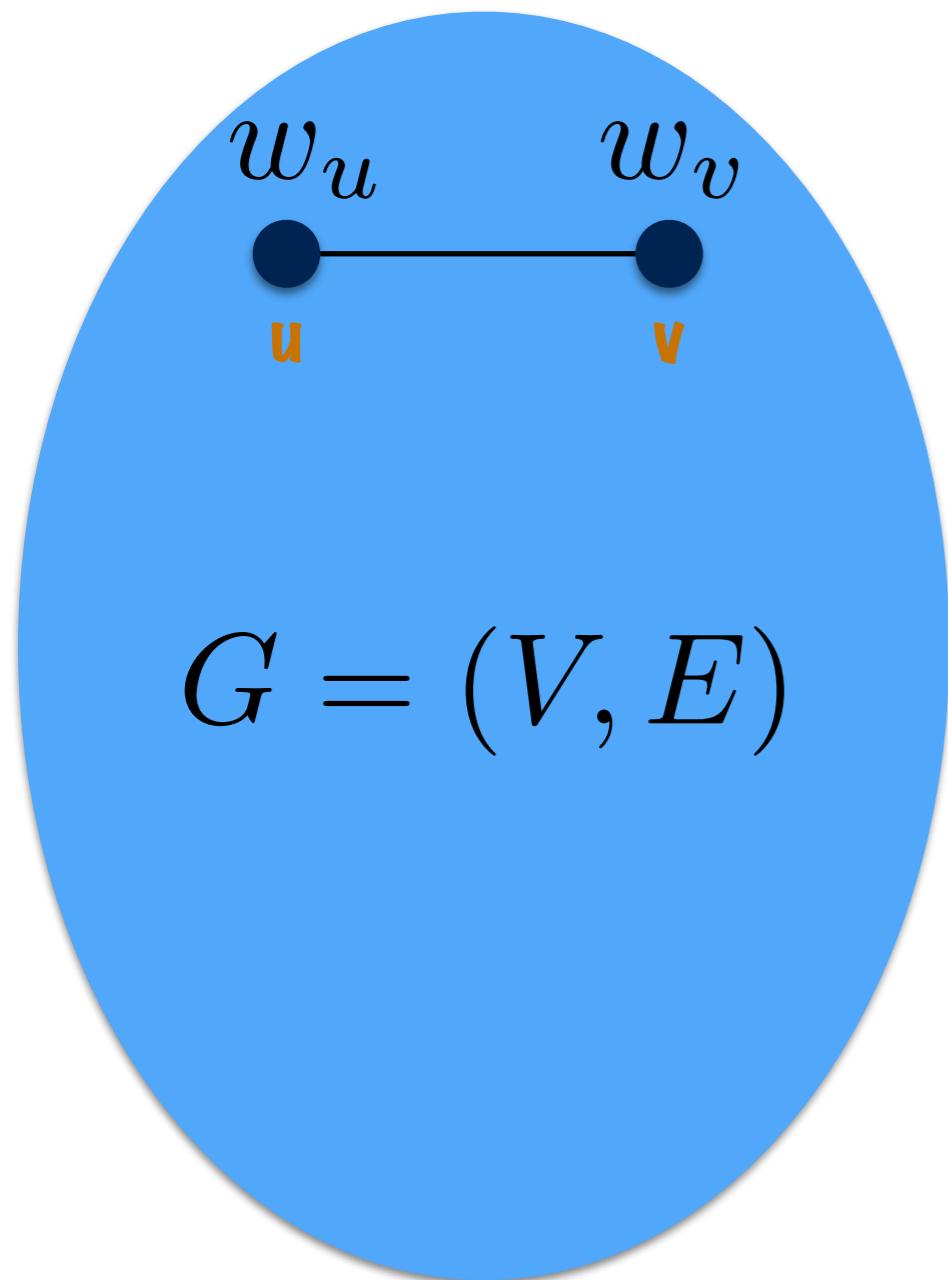
$$x_a = x_c = x_d = x_f = x_g = 1$$

$$x_b = x_e = 0$$

satisfies all constraints

value = 14

in general



Constraints:

$$\forall u \in V : x_u = 0 \text{ or } 1$$

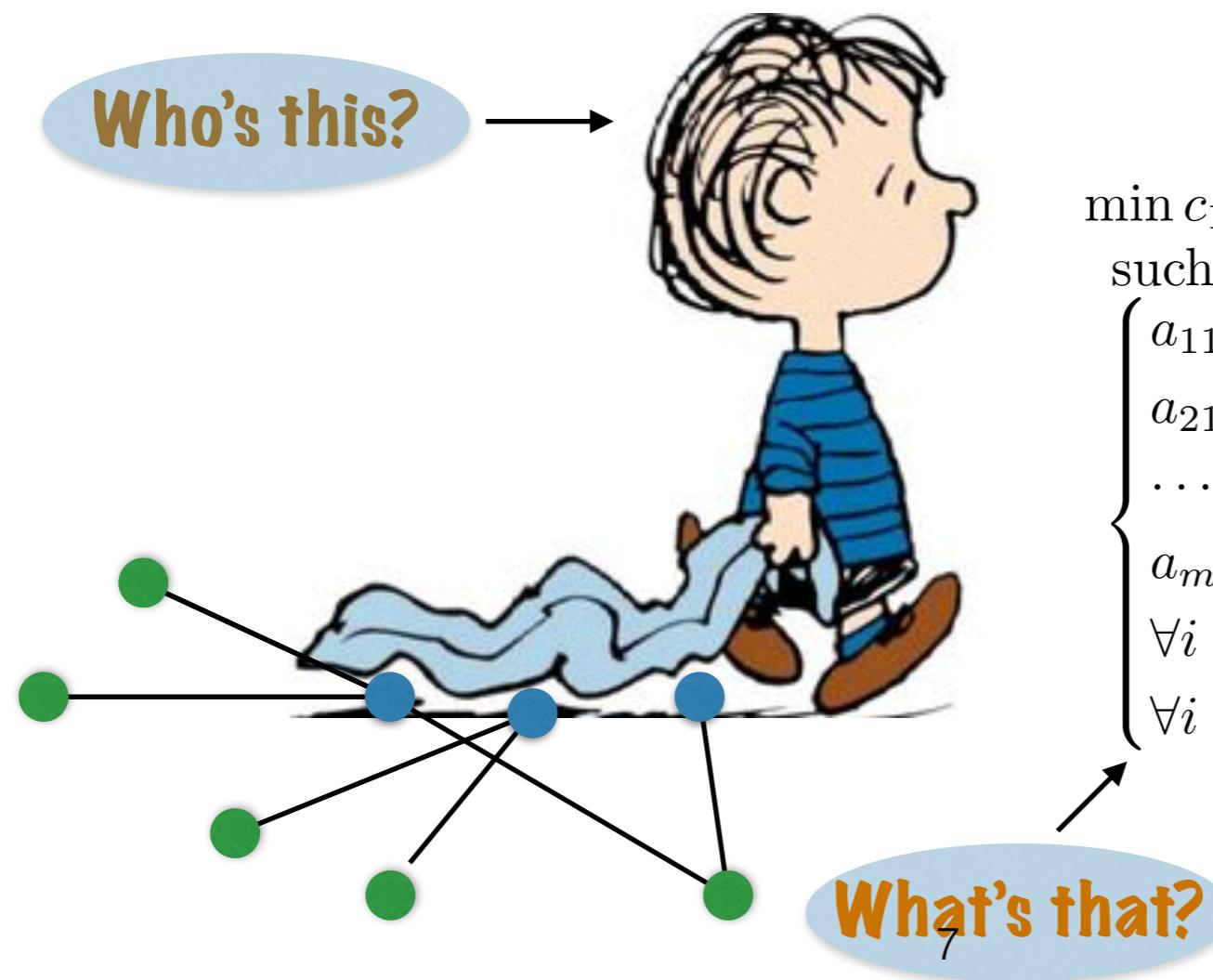
$$\forall \{u, v\} \in E : x_u + x_v \geq 1$$

Objective: $\min \sum_u w_u x_u$



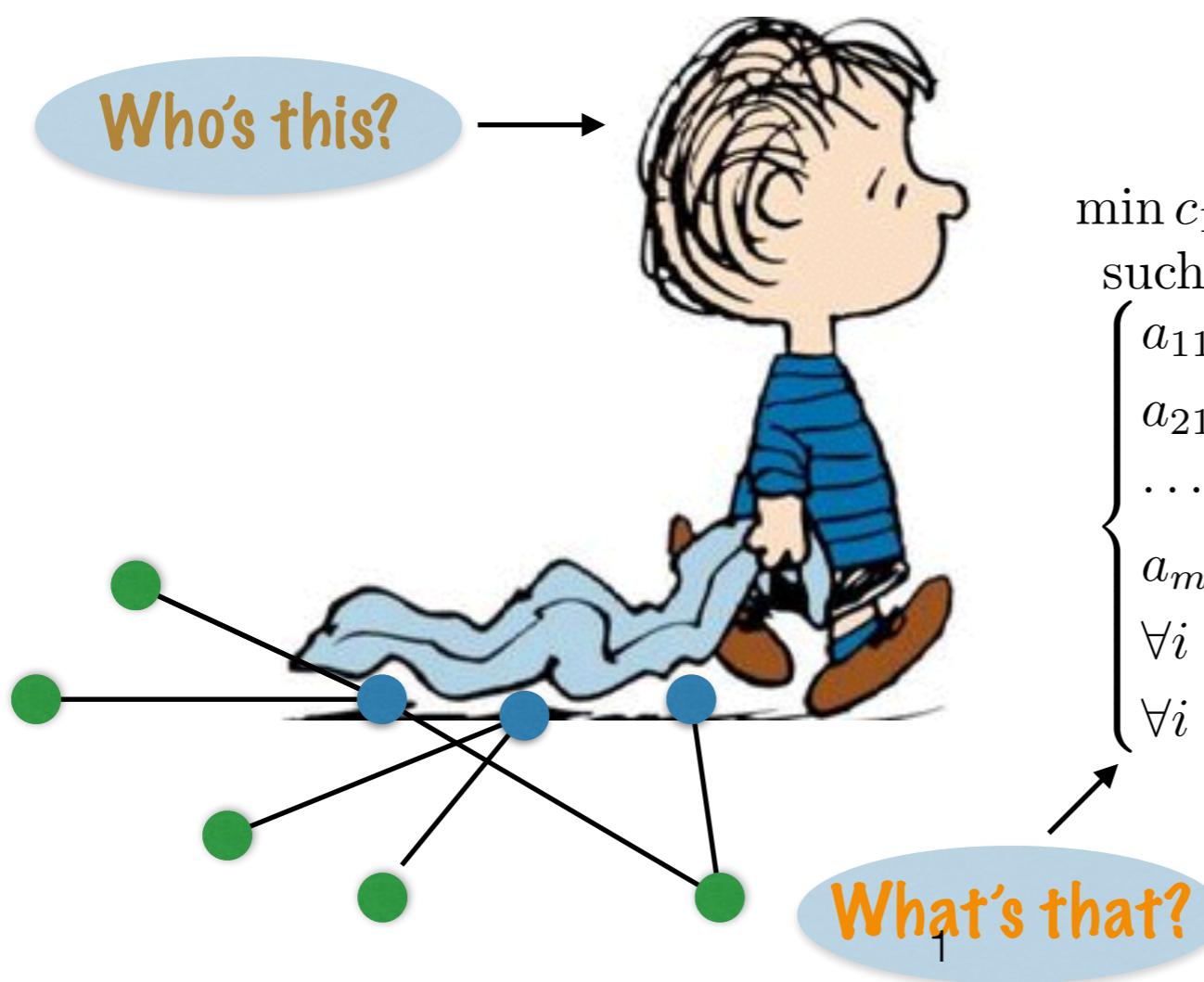
integer program

Approximation algorithms, vertex cover, and linear programming



$$\begin{aligned} & \min c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \\ \forall i : 0 \leq x_i \leq 1 \\ \forall i : x_i \text{ real number} \end{array} \right. \end{aligned}$$

Approximation algorithms, vertex cover, and linear programming



$$\begin{aligned} & \min c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \\ \forall i : 0 \leq x_i \leq 1 \\ \forall i : x_i \text{ real number} \end{array} \right. \end{aligned}$$

Integer program

$$\min c_1x_1 + c_2x_2 + \cdots + c_nx_n$$

such that

$$\left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \\ \forall i : 0 \leq x_i \leq 1 \\ \forall i : x_i \text{ integer} \end{array} \right.$$

NP-hard

Linear program

$$\min c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$$

such that

$$\left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \\ \forall i : 0 \leq x_i \leq 1 \\ \forall i : x_i \text{ real number} \end{array} \right.$$

polynomial time

Two ways to present

$$\min c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$$

such that

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \\ \forall i : 0 \leq x_i \leq 1 \\ \forall i : x_i \text{ real number} \end{cases}$$

$\min c \cdot x$ such that

$$\begin{cases} Ax \geq b \\ x \in [0, 1]^n \\ x \text{ vector of } \mathbb{R}^n \end{cases}$$

Linear program

Same linear program

Integer vs. linear programs

IP

$$\min c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$$

such that

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \\ \forall i : 0 \leq x_i \leq 1 \\ \forall i : x_i \text{ integer} \end{cases}$$

LP

$$\min c_1 x_1 + c_2 x_2 + \cdots + c_n x_n$$

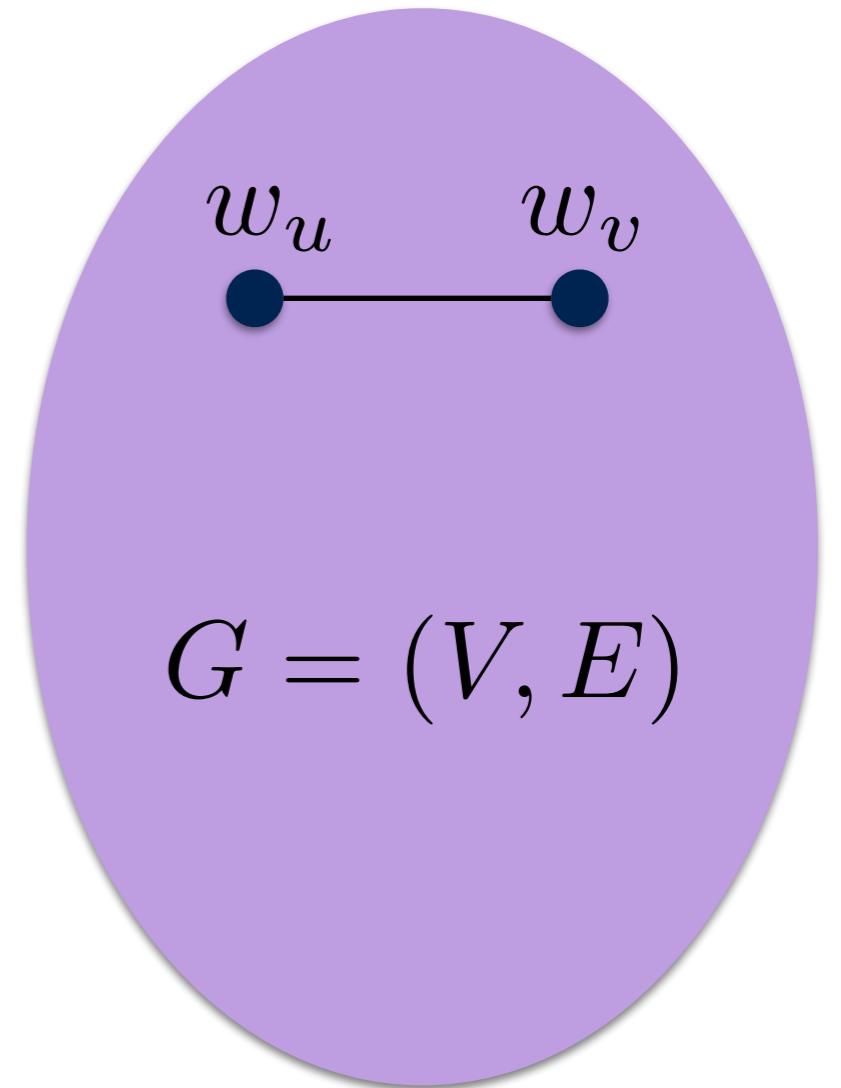
such that

$$\begin{cases} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \\ \forall i : 0 \leq x_i \leq 1 \\ \forall i : x_i \text{ real number} \end{cases}$$

NP-hard

polynomial time

Vertex cover linear program



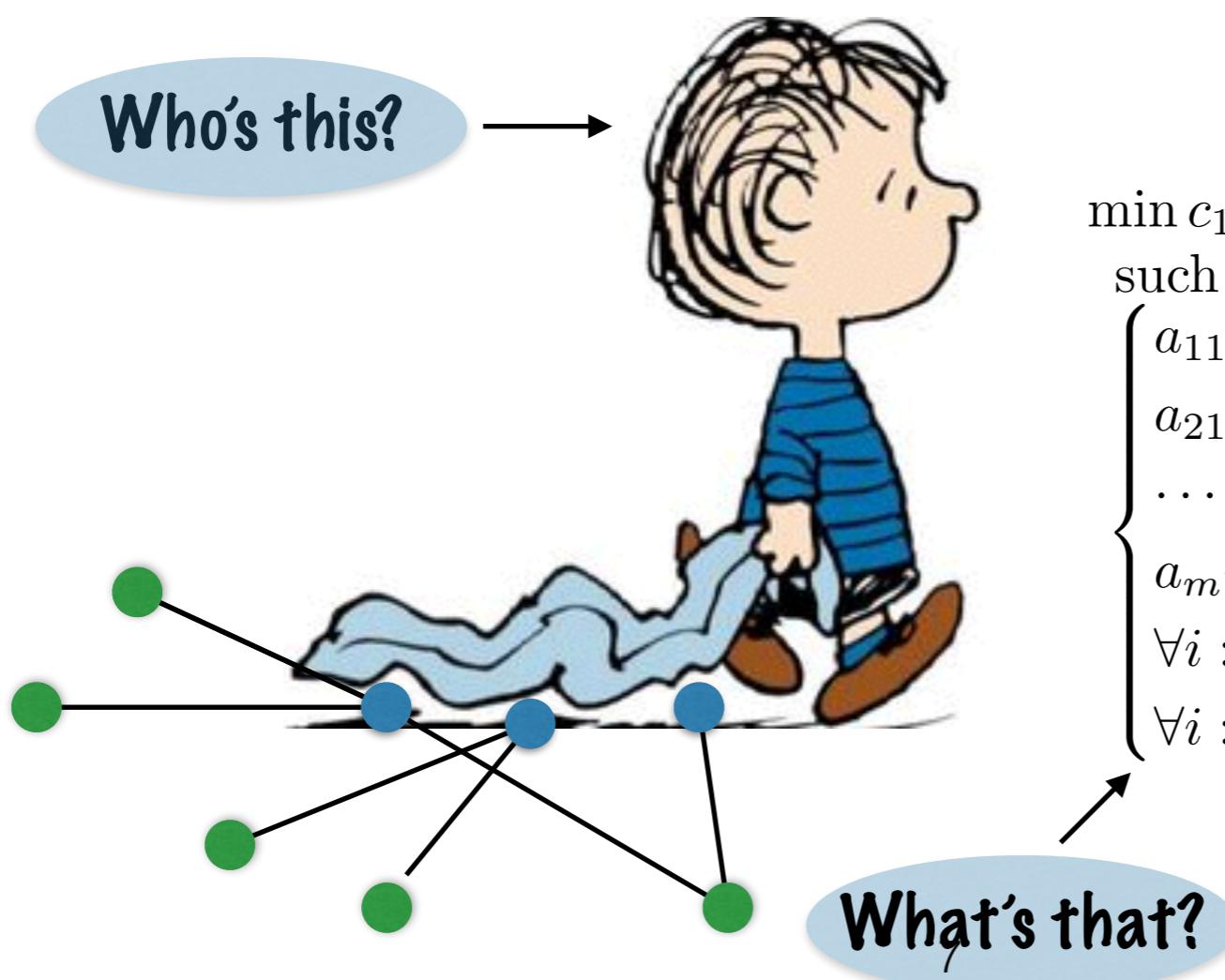
Constraints:

$$\forall u \in V : 0 \leq x_u \leq 1$$

$$\forall \{u, v\} \in E : x_u + x_v \geq 1$$

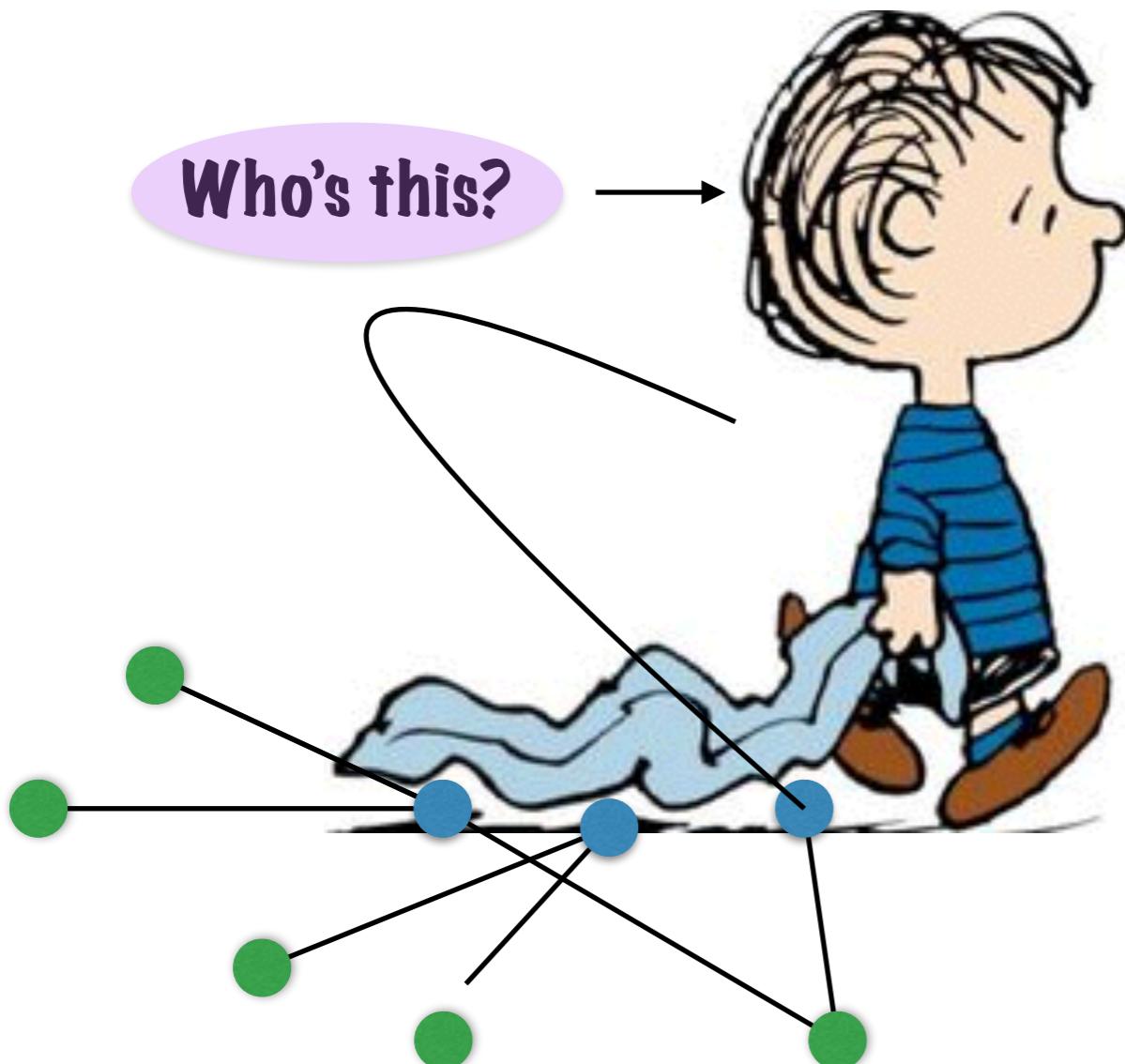
Objective: $\min \sum_u w_u x_u$

Approximation algorithms, vertex cover, and linear programming



$$\begin{aligned} & \min c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \\ \forall i : 0 \leq x_i \leq 1 \\ \forall i : x_i \text{ real number} \end{array} \right. \end{aligned}$$

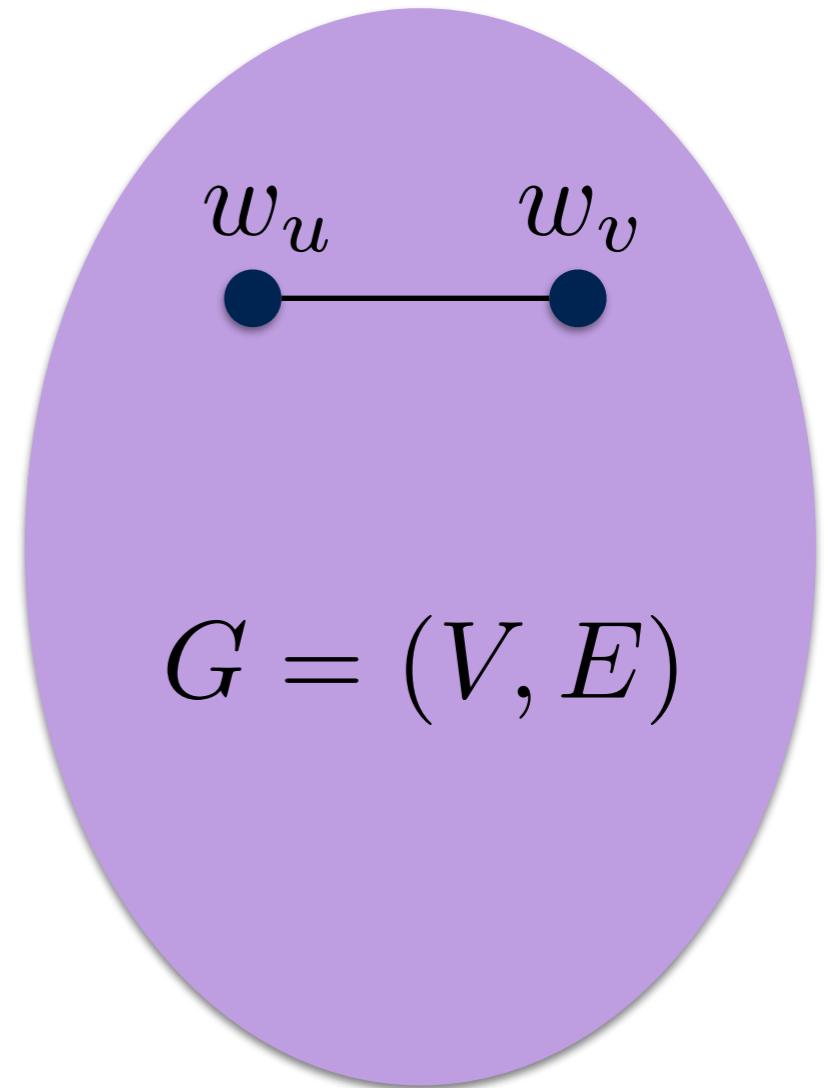
Approximation algorithms, vertex cover, and linear programming



$$\begin{aligned} & \min c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots & \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \end{array} \right. \\ & \forall i : 0 \leq x_i \leq 1 \\ & \forall i : x_i \text{ real number} \end{aligned}$$

What's that?

Using the LP (1/3)



Constraints:

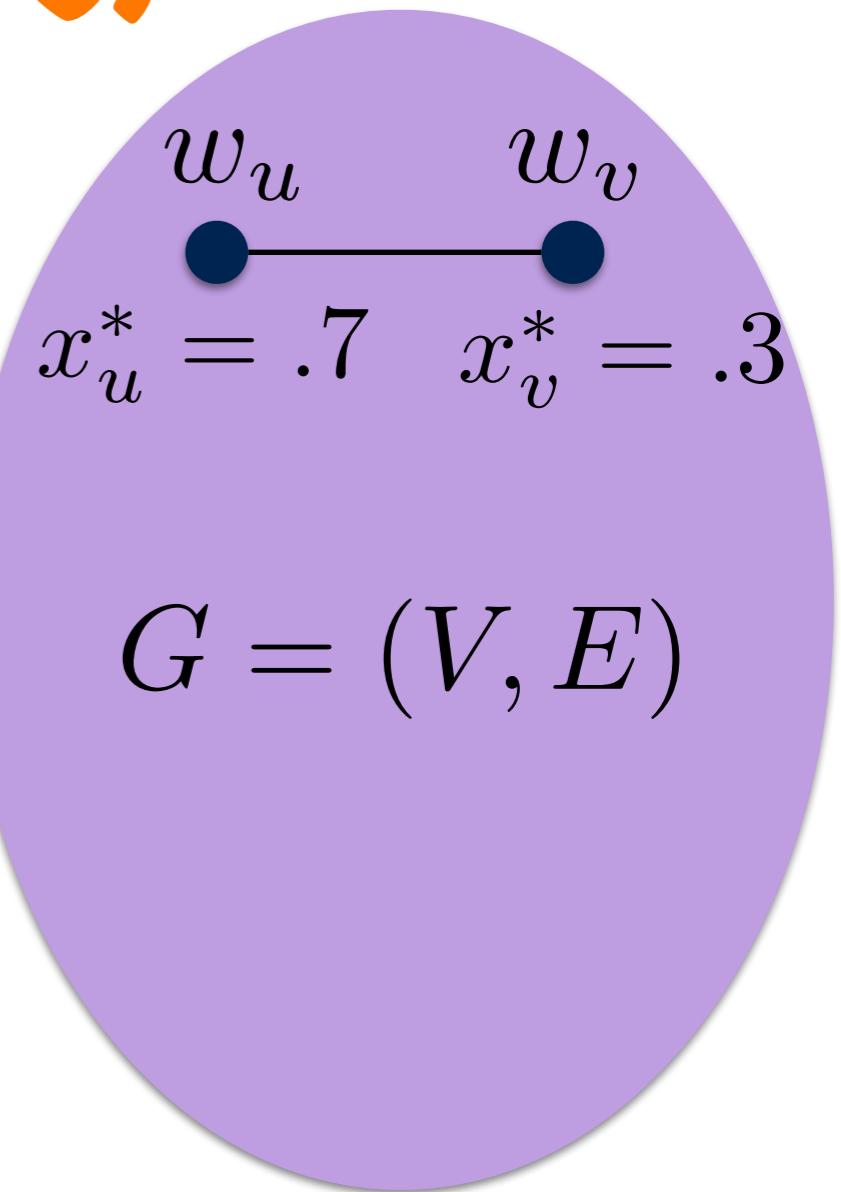
$$\forall u \in V : 0 \leq x_u \leq 1$$

$$\forall \{u, v\} \in E : x_u + x_v \geq 1$$

Objective: $\min \sum_u w_u x_u$

Using the LP (2/3)

1. Solving the LP



$\implies (x_u^*)_{u \in V}$ such that

$$\forall u \in V : 0 \leq x_u^* \leq 1$$

$$\forall \{u, v\} \in E : x_u^* + x_v^* \geq 1$$

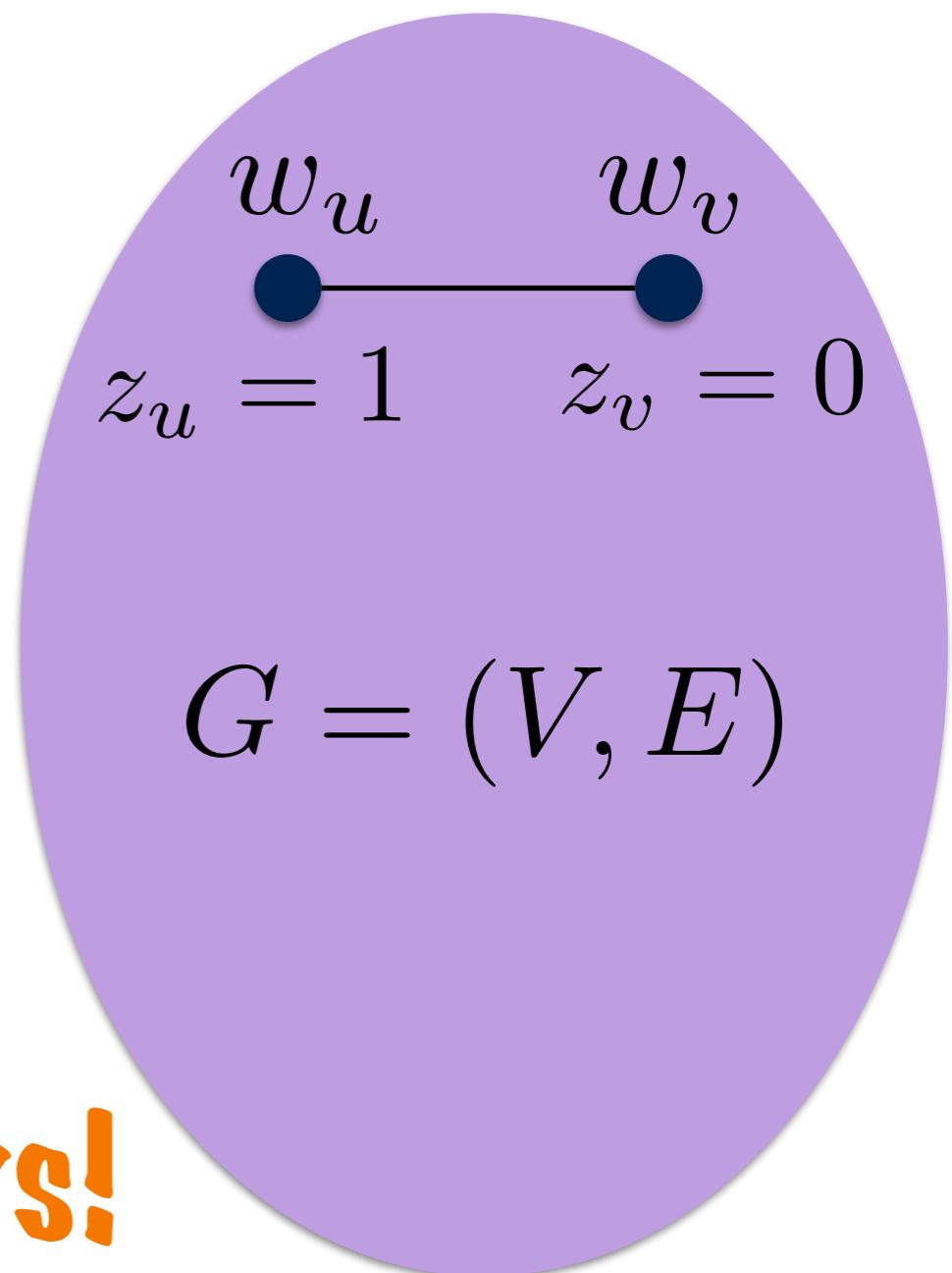
$$\sum_u w_u x_u^* \text{ minimum}$$

Using the LP (3/3)

2. Rounding the LP solution

$\implies (z_u)_{u \in V}$ defined by

$$z_u = \begin{cases} 1 & \text{if } x_u^* \geq .5 \\ 0 & \text{otherwise} \end{cases}$$



We are back to integers!

Runtime

1. Solve the LP

$(x_u^*)_{u \in V}$ such that

$$\forall u \in V : 0 \leq x_u^* \leq 1 \quad \longleftarrow$$

$$\forall \{u, v\} \in E : x_u^* + x_v^* \geq 1$$

$$\sum_u w_u x_u^* \text{ minimum}$$

Polynomial time

2. Round the LP solution

$\implies (z_u)_{u \in V}$ defined by

$$z_u = \begin{cases} 1 & \text{if } x_u^* \geq .5 \\ 0 & \text{otherwise} \end{cases}$$

Linear time

3. Output

$\{u \in V \text{ such that } z_u = 1\}$

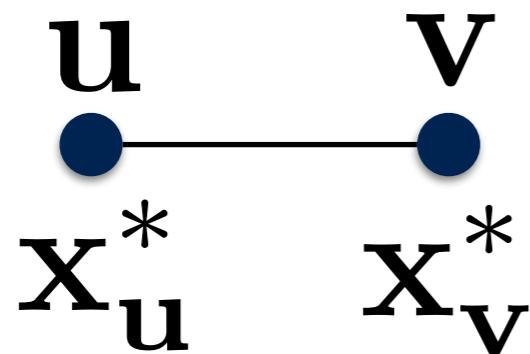
Correctness

... is it a vertex cover?

Does output cover all edges?

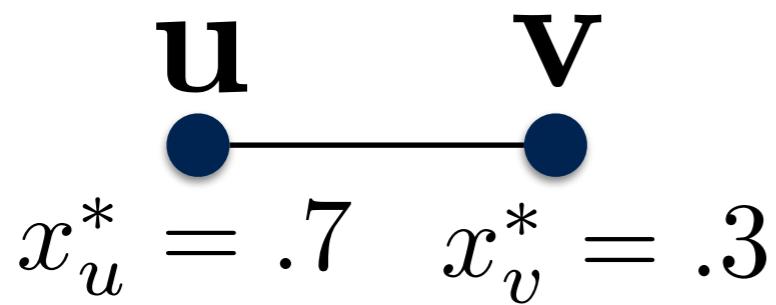


$$\{u, v\} : x_u^* + x_v^* \geq 1$$



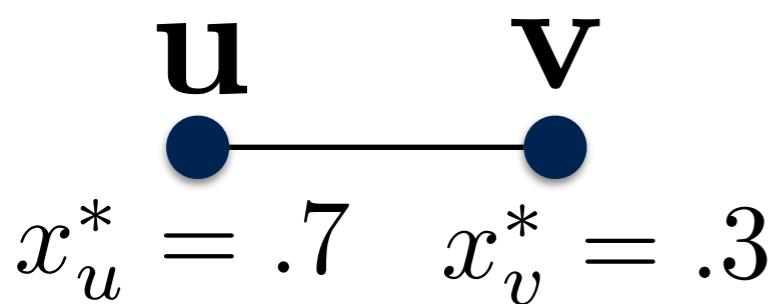
Does output cover all edges?

$$x_u^* + x_v^* \geq 1$$



$$x_u^* \geq .5 \text{ or } x_v^* \geq .5$$

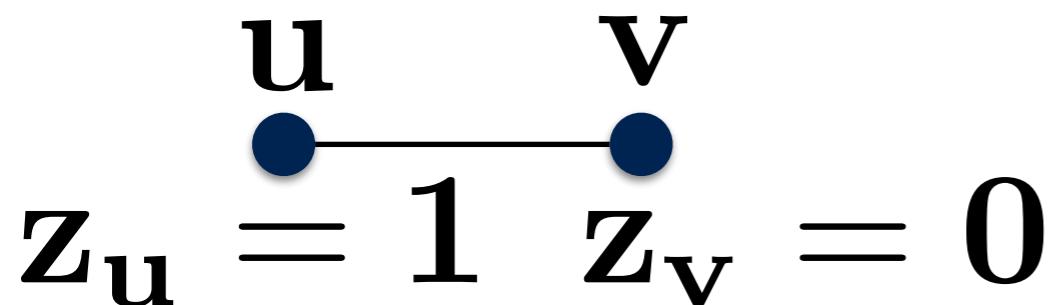
Does output cover all edges?



$\implies (z_u)_{u \in V}$ defined by

$$z_u = \begin{cases} 1 & \text{if } x_u^* \geq .5 \\ 0 & \text{otherwise} \end{cases}$$

$$x_u^* \geq .5 \text{ or } x_v^* \geq .5$$

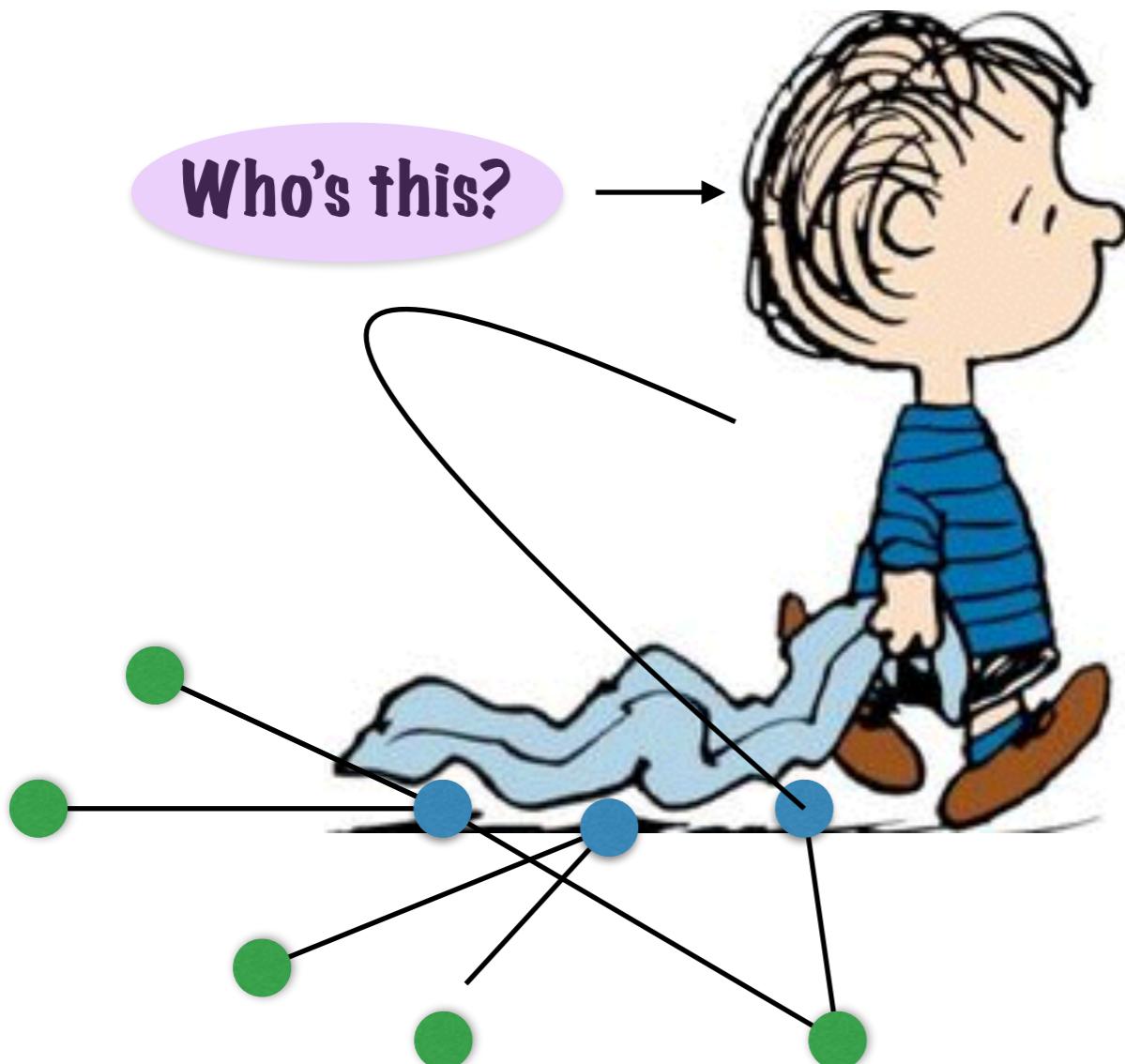


u is in output

u

$$z_u = 1 \text{ or } z_v = 1$$

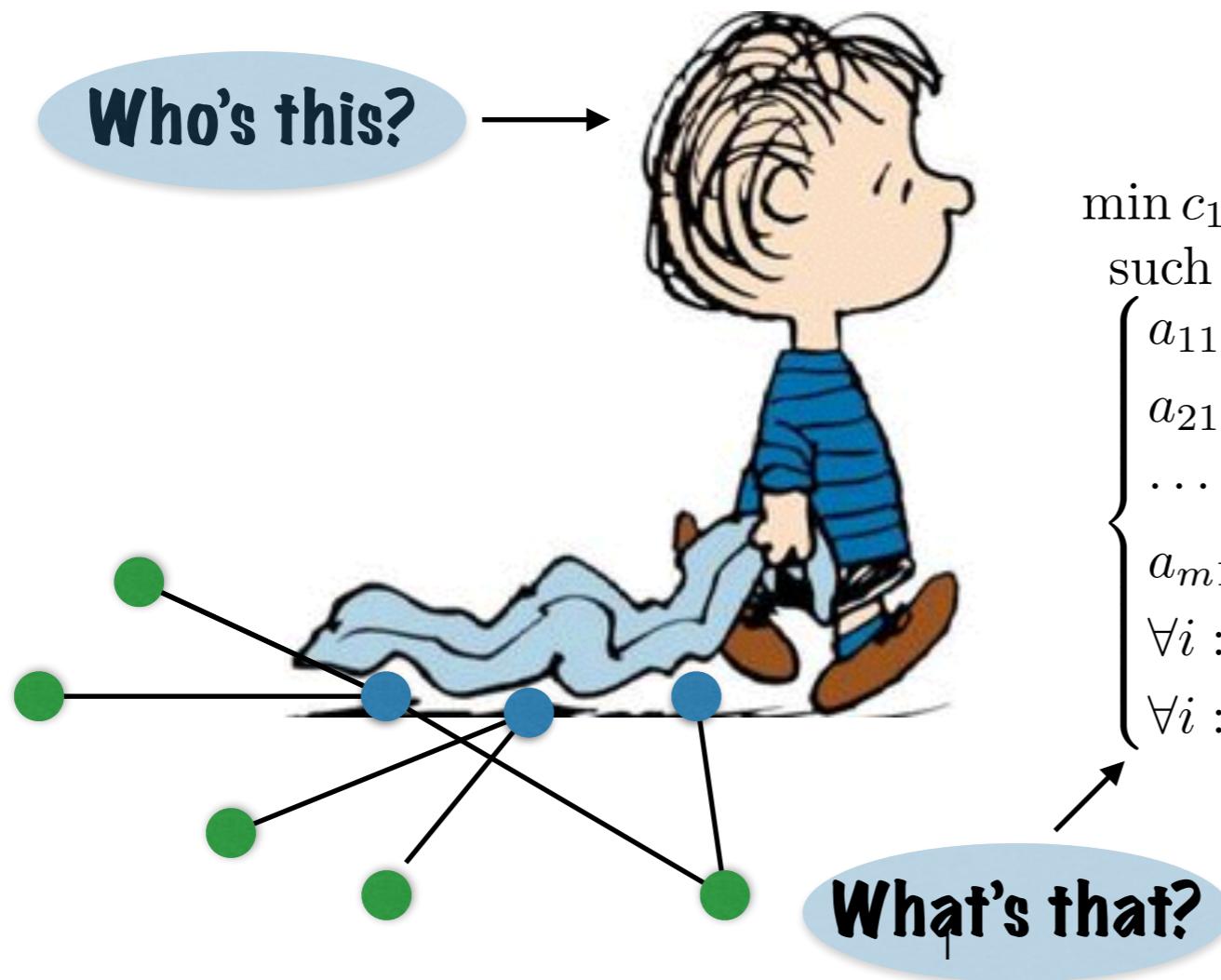
Approximation algorithms, vertex cover, and linear programming



$$\begin{aligned} & \min c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots & \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \end{array} \right. \\ & \forall i : 0 \leq x_i \leq 1 \\ & \forall i : x_i \text{ real number} \end{aligned}$$

What's that?

Approximation algorithms, vertex cover, and linear programming



$$\begin{aligned} & \min c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \end{array} \right. \\ & \forall i : 0 \leq x_i \leq 1 \\ & \forall i : x_i \text{ real number} \end{aligned}$$

Quality of output?

2. Round the LP solution

$$z_u \in \{0, 1\}$$

3. Output $\{u : z_u = 1\}$

Output cost = $\sum_u w_u z_u$

1. Solve the LP

$$(\mathbf{x}_u^*)$$

2. Round the LP solution

$$z_u = \begin{cases} 1 & \text{if } x_u^* \geq 1/2 \\ 0 & \text{otherwise} \end{cases}$$

Observe: $z_u \leq 2x_u^*$

1. Solve the LP

IP min: $x(u)=1$
iff u in optimum
vertex cover

$$\min \sum_u w_u x_u$$

$$x_u + x_v \geq 1$$

$$x_u \in \{0, 1\}$$

LP min: $x^*(u)$

$$\min \sum_u w_u x_u^*$$

$$x_u^* + x_v^* \geq 1$$

$$0 \leq x_u^* \leq 1$$

The LP is a relaxation of the IP

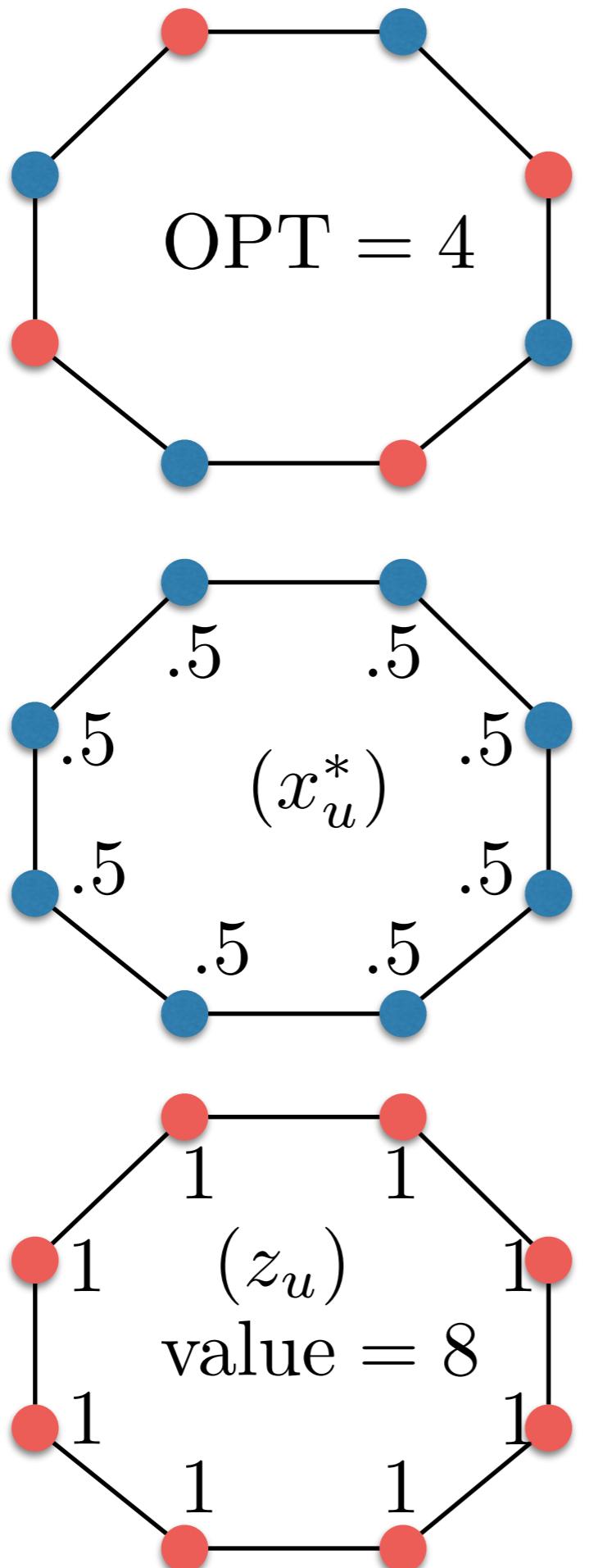
$$\sum_u w_u x_u^* \leq \text{OPT}$$

Combine

$$\begin{aligned}\text{Output cost} &= \sum_u w_u z_u \\ &\leq 2 \sum_u w_u x_u^* \\ &\leq 2 \text{OPT}\end{aligned}$$

Thm: output is a vertex cover
of value at most 2 OPT

Is the analysis tight?



How good is that?

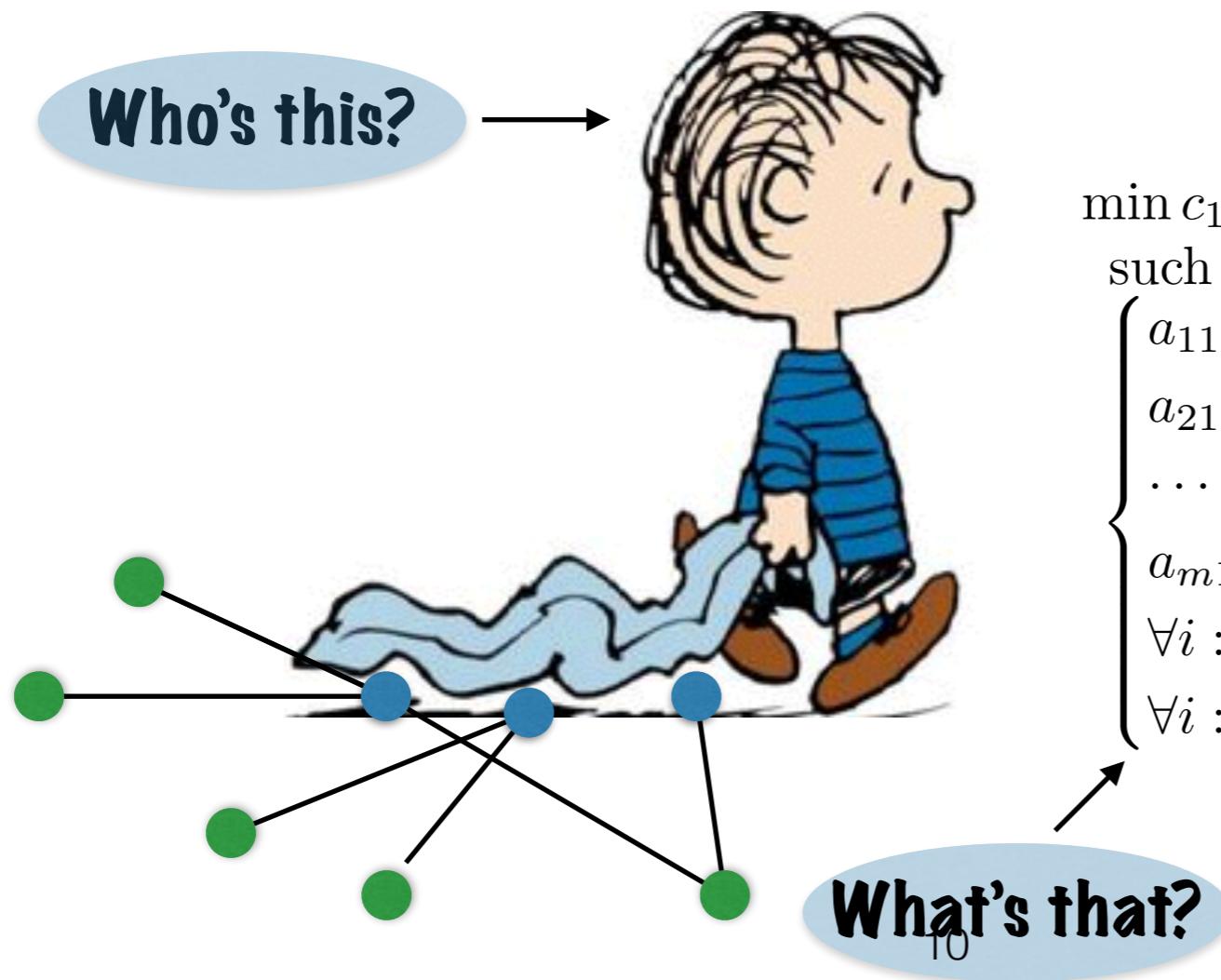
Typical performance
(hearsay):
within 10% of optimum

How do we know?

Can compare output
value to

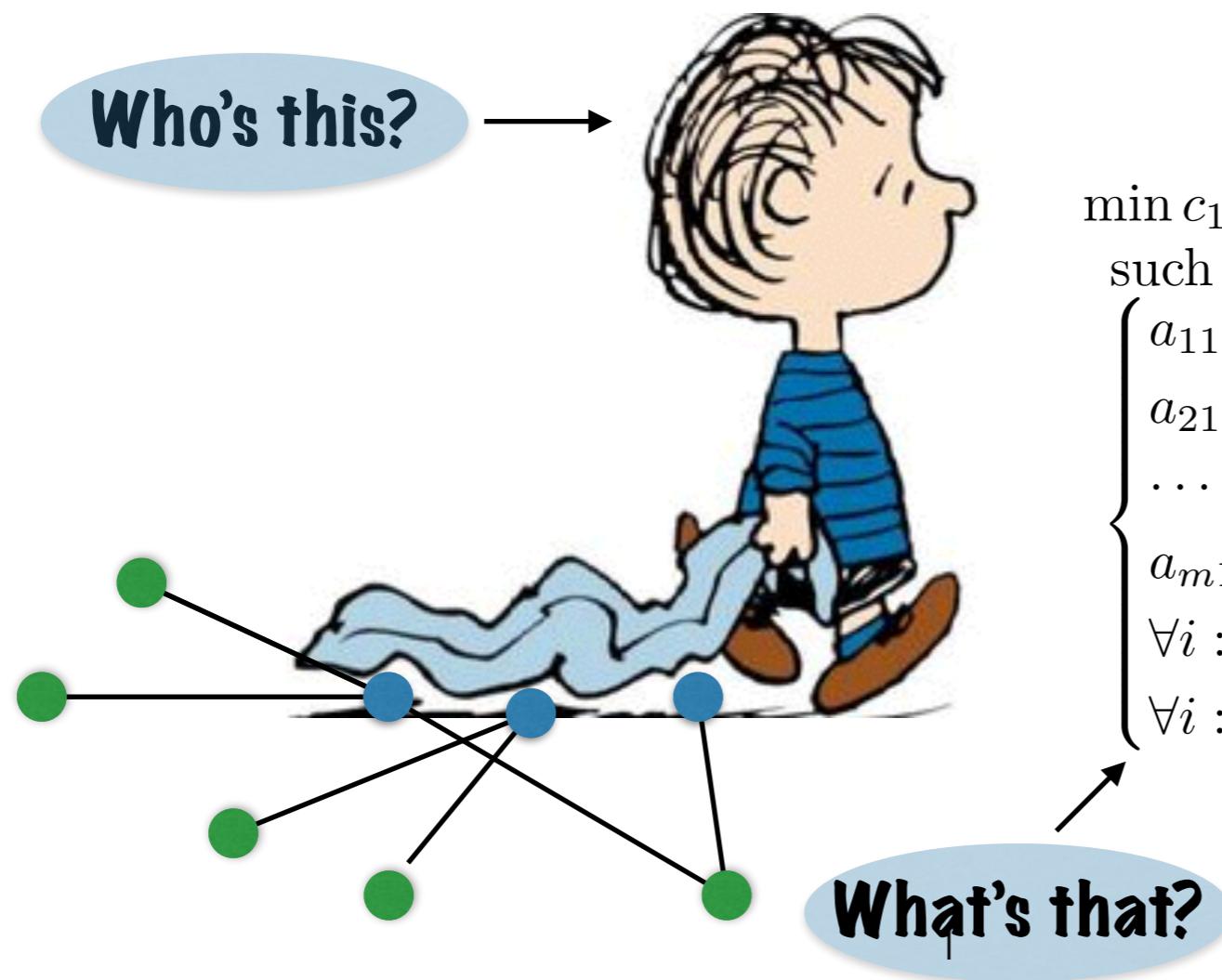
$$\sum_u w_u x_u^*$$

Approximation algorithms, vertex cover, and linear programming



$$\begin{aligned} & \min c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \\ \forall i : 0 \leq x_i \leq 1 \\ \forall i : x_i \text{ real number} \end{array} \right. \end{aligned}$$

Approximation algorithms, vertex cover, and linear programming



$$\begin{aligned} & \min c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \end{array} \right. \\ & \forall i : 0 \leq x_i \leq 1 \\ & \forall i : x_i \text{ real number} \end{aligned}$$

The method

Meta approximation algorithm

- **Find IP**
- **Solve LP relaxation**
- **Round solution to integers**

Analysis

- correct: does it satisfy conditions?
- efficient: polynomial runtime?
- good: value of output solution within factor of optimal value?

How good is it?

Method

- Output can be related to LP value

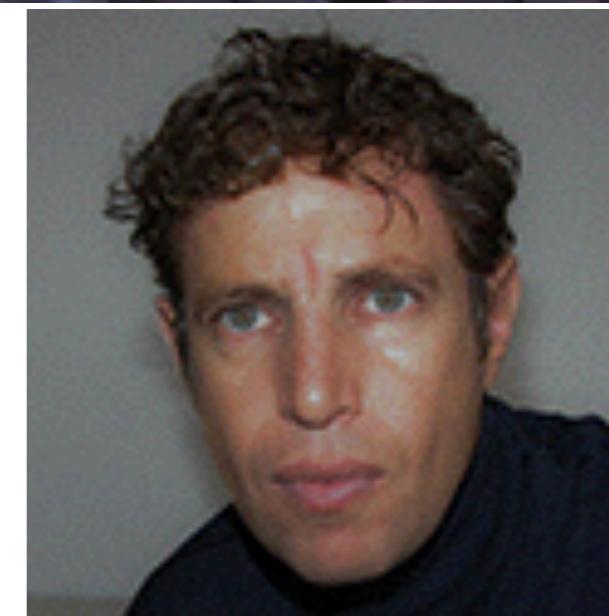
- OPT can be bounded by LP value

Combine

Message: for analysis,
focus on LP value.



Karp (1972)
NP-complete

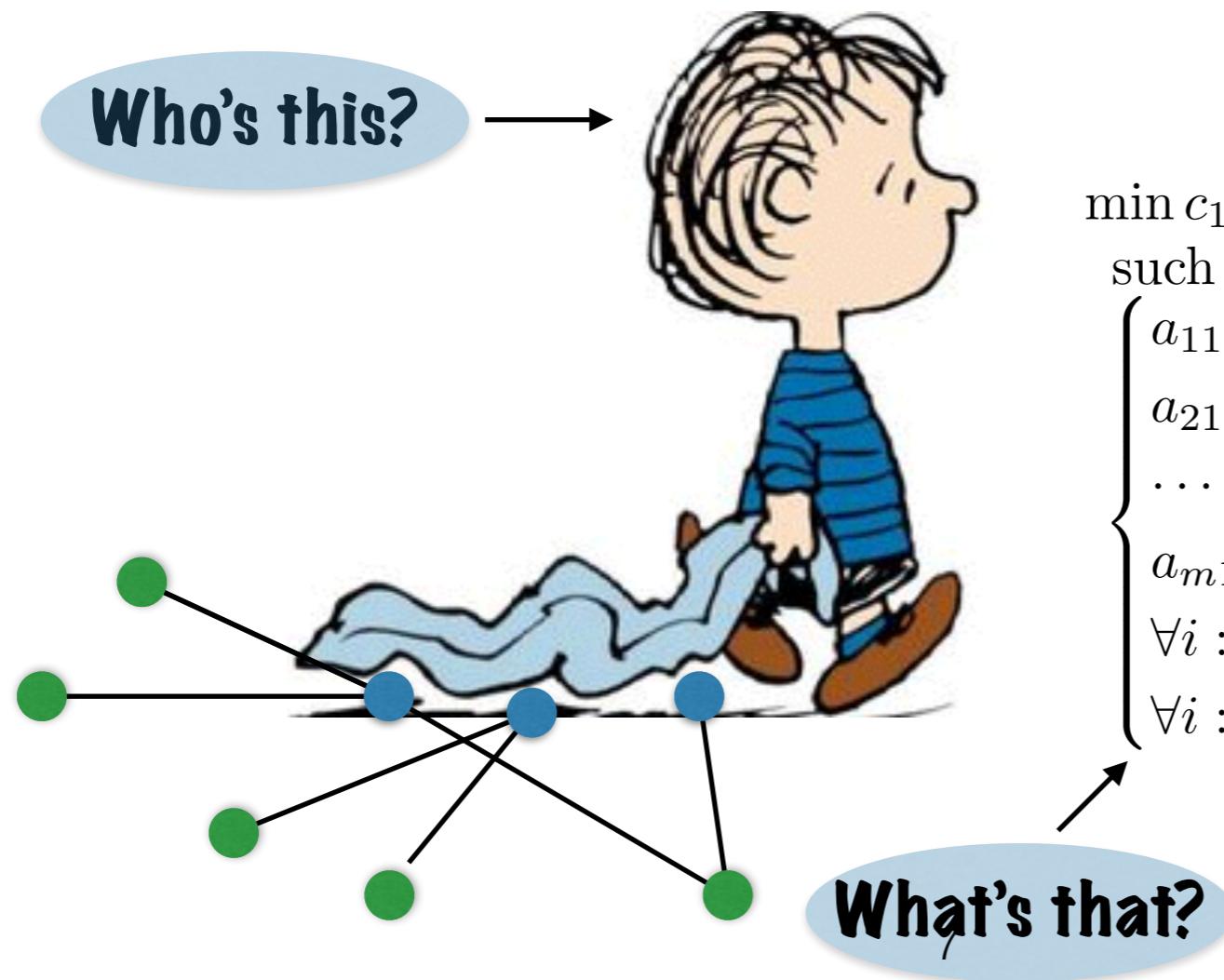


**Khot
Regev
(2003)**
Conditional <2 hard

**Papadimitriou
Yannakakis:**
1.0001 hard (1991)

**Dinur
Safra:**
1.36 hard (2002)

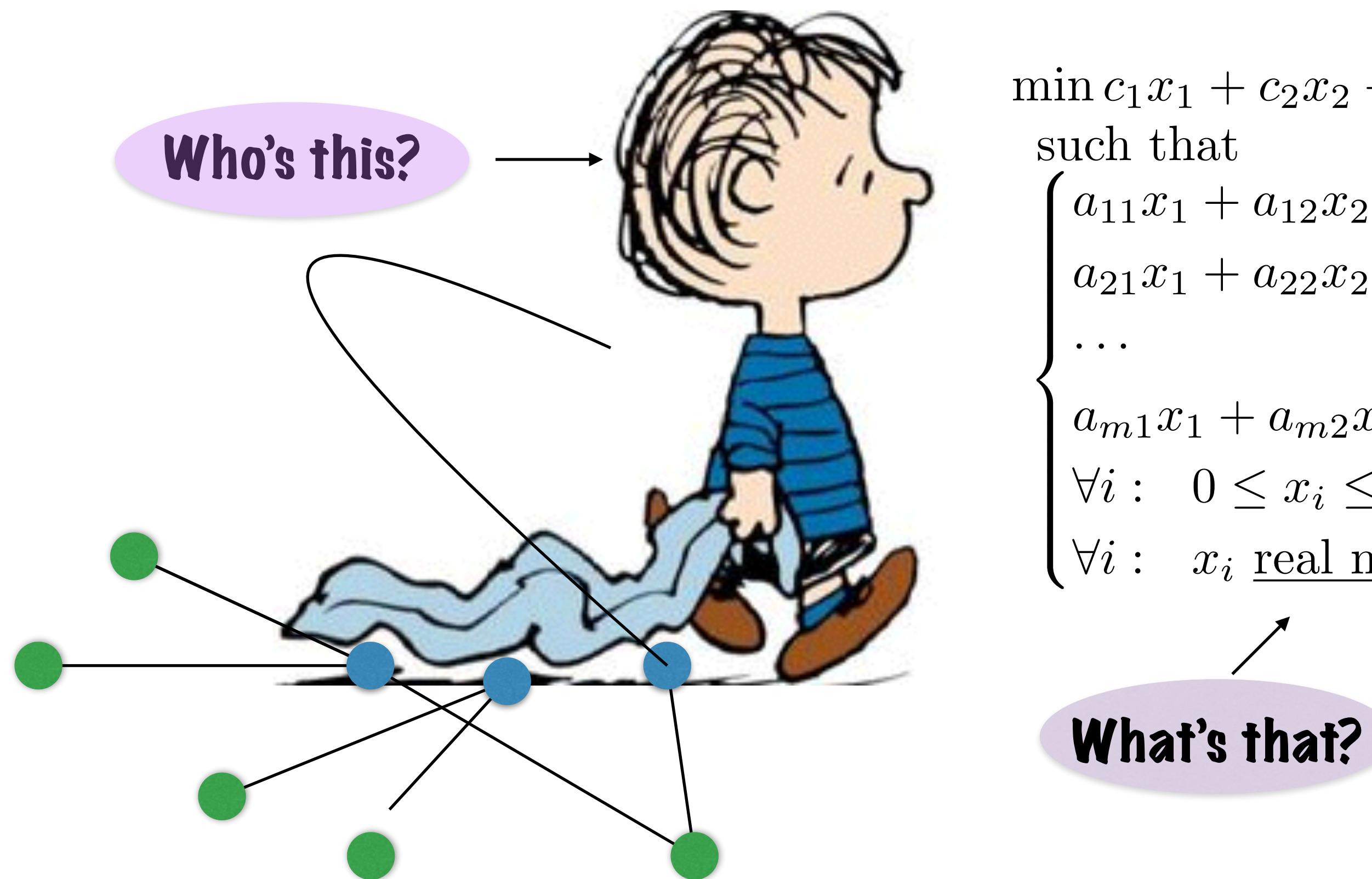
Approximation algorithms, vertex cover, and linear programming



$$\begin{aligned} & \min c_1x_1 + c_2x_2 + \cdots + c_nx_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \end{array} \right. \\ & \forall i : 0 \leq x_i \leq 1 \\ & \forall i : x_i \text{ real number} \end{aligned}$$

What's that?

Approximation algorithms, vertex cover, and linear programming



$$\begin{aligned} & \min c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11}x_1 + a_{12}x_2 + \cdots + a_{1n}x_n & \geq b_1 \\ a_{21}x_1 + a_{22}x_2 + \cdots + a_{2n}x_n & \geq b_2 \\ \dots \\ a_{m1}x_1 + a_{m2}x_2 + \cdots + a_{mn}x_n & \geq b_m \end{array} \right. \\ & \forall i : 0 \leq x_i \leq 1 \\ & \forall i : x_i \text{ real number} \end{aligned}$$

What's that?

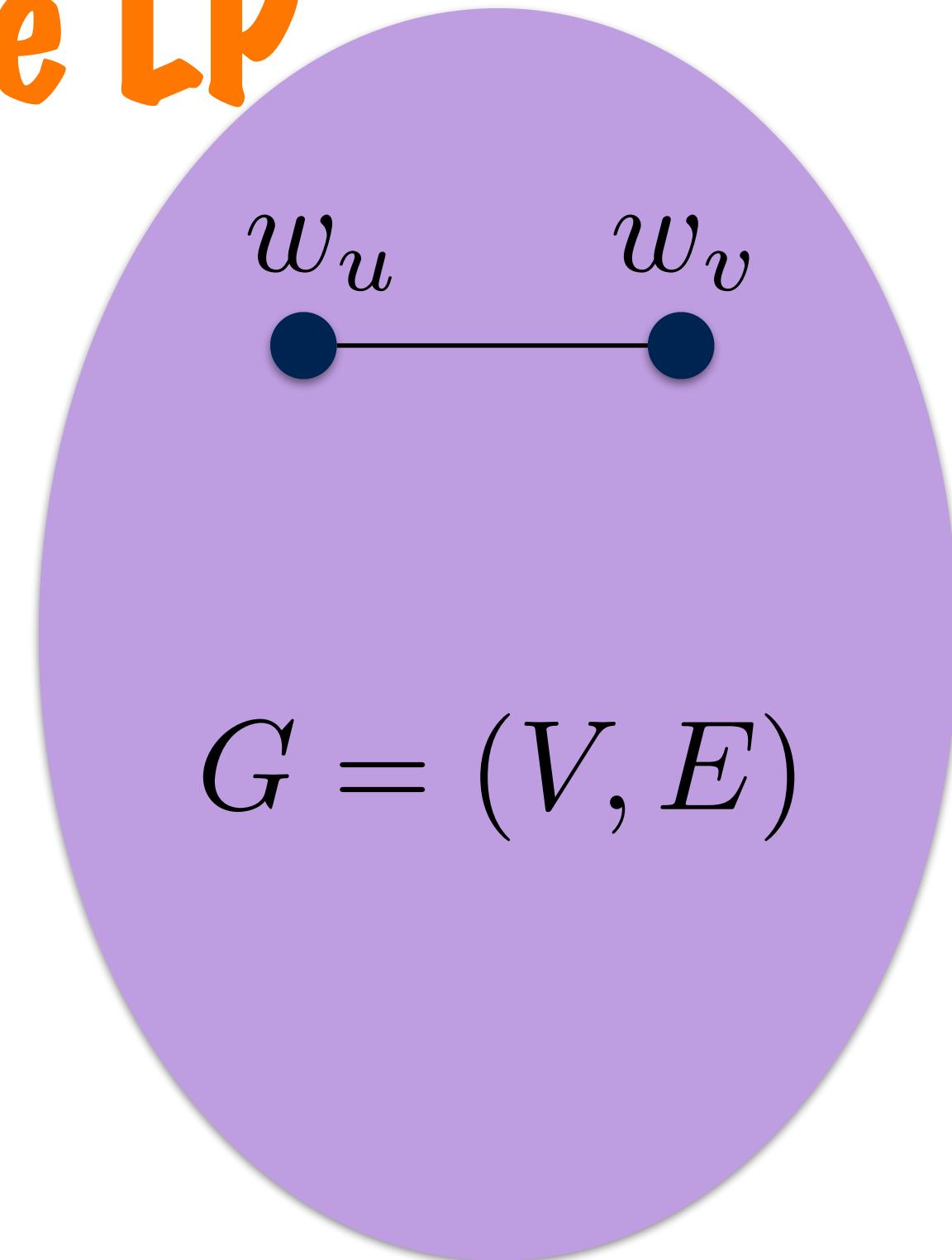
Property of the LP

Constraints:

$$\forall u \in V : 0 \leq x_u \leq 1$$

$$\forall \{u, v\} \in E : x_u + x_v \geq 1$$

Objective: $\min \sum_u w_u x_u$



Theorem:

there exists an optimal solution

s.t. every coordinate is in $\{0, .5, 1\}$

and there is a polynomial-time algorithm to construct it

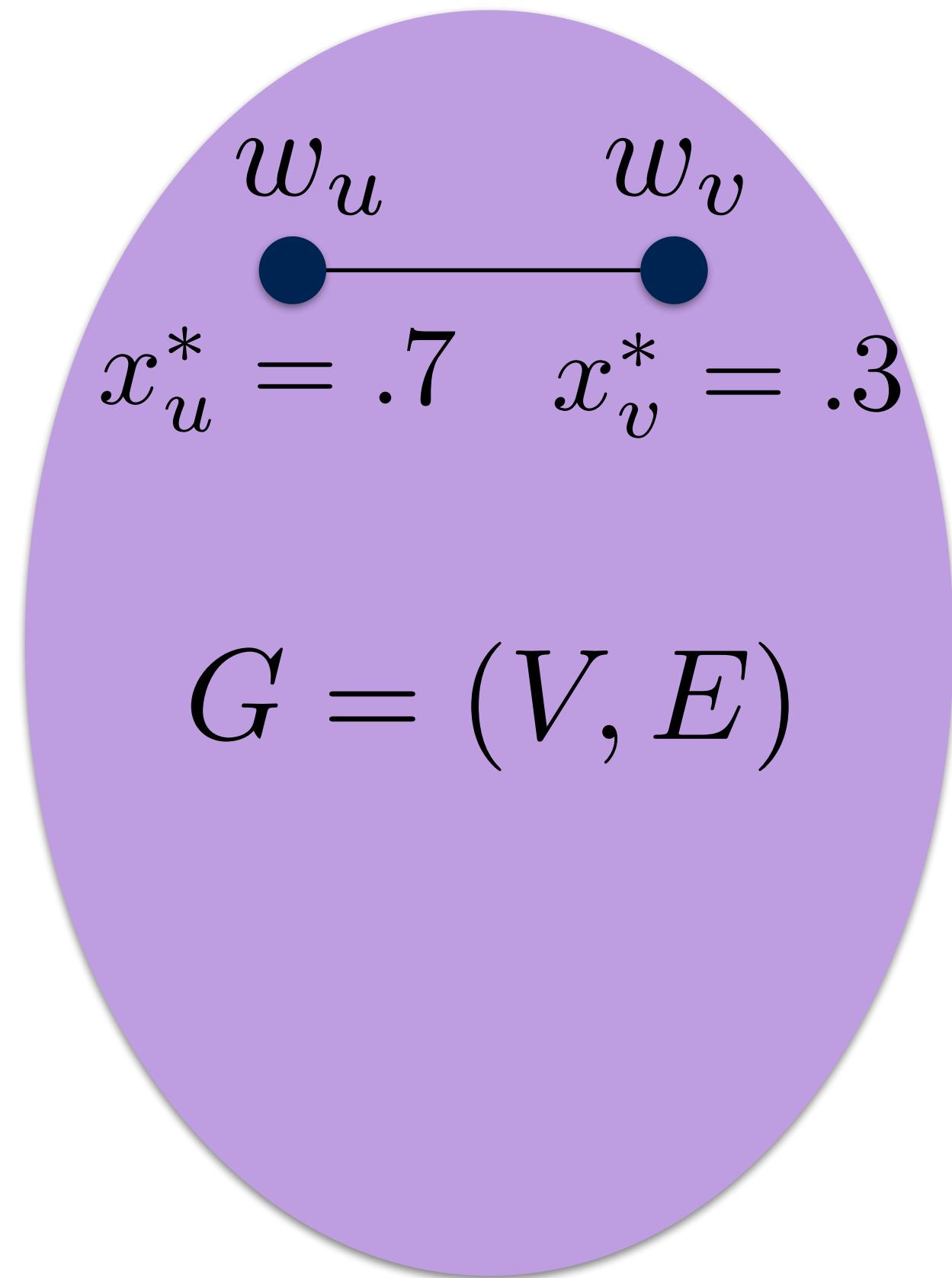
1. Solve the LP

$\implies (x_u^*)_{u \in V}$ such that

$$\forall u \in V : 0 \leq x_u^* \leq 1$$

$$\forall \{u, v\} \in E : x_u^* + x_v^* \geq 1$$

$$\sum_u w_u x_u^* \text{ minimum}$$



2. Freeze all variables with value in {0, .5, 1}

3. While some variables are not frozen

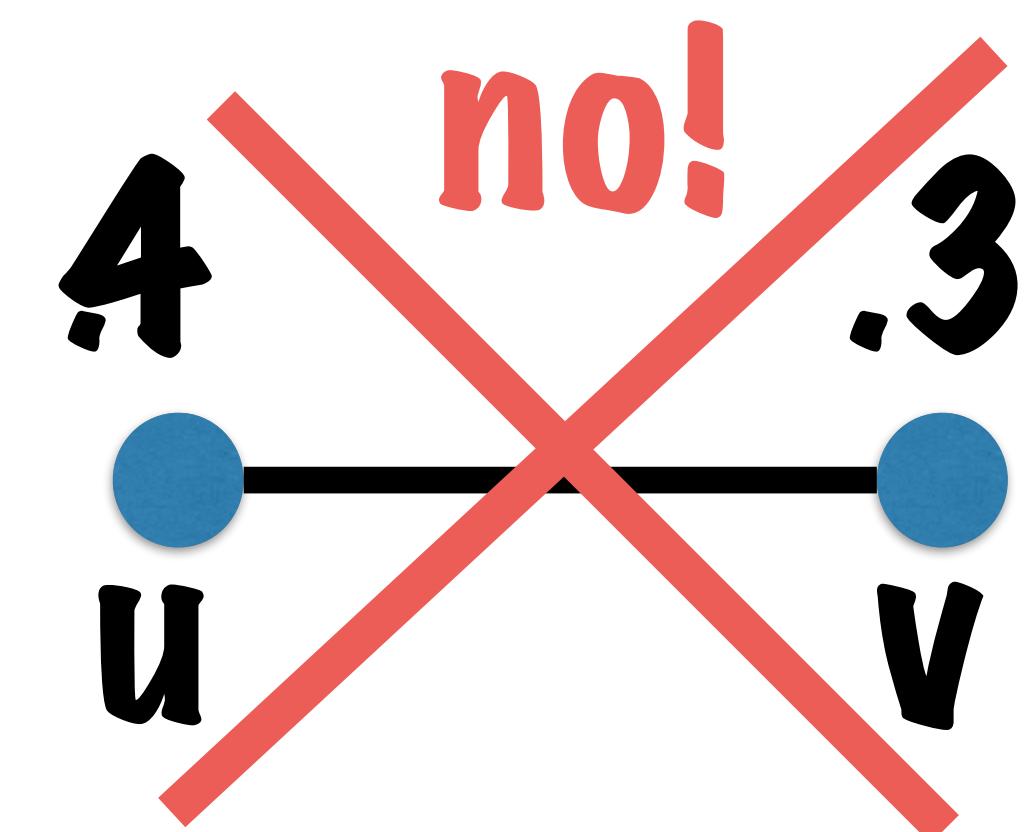
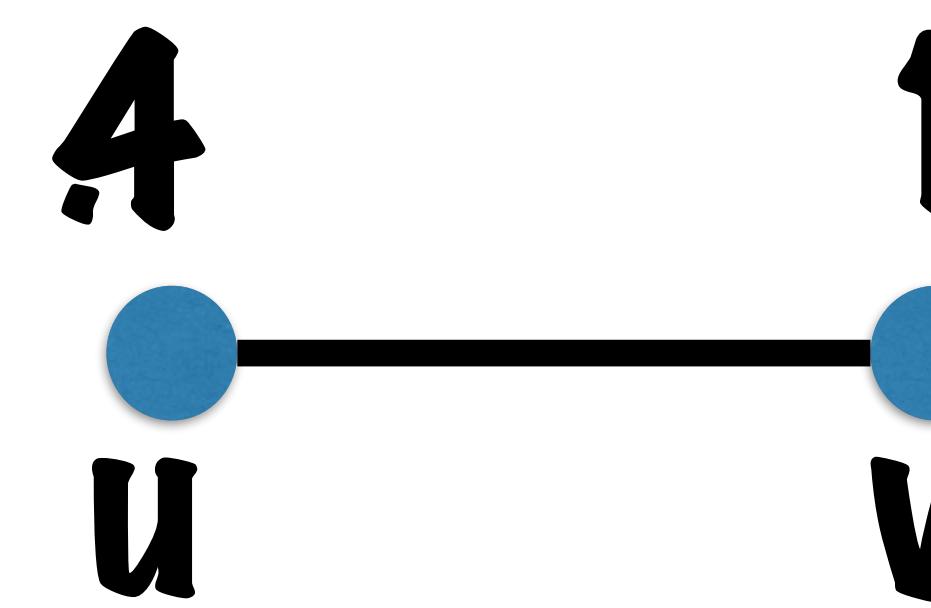
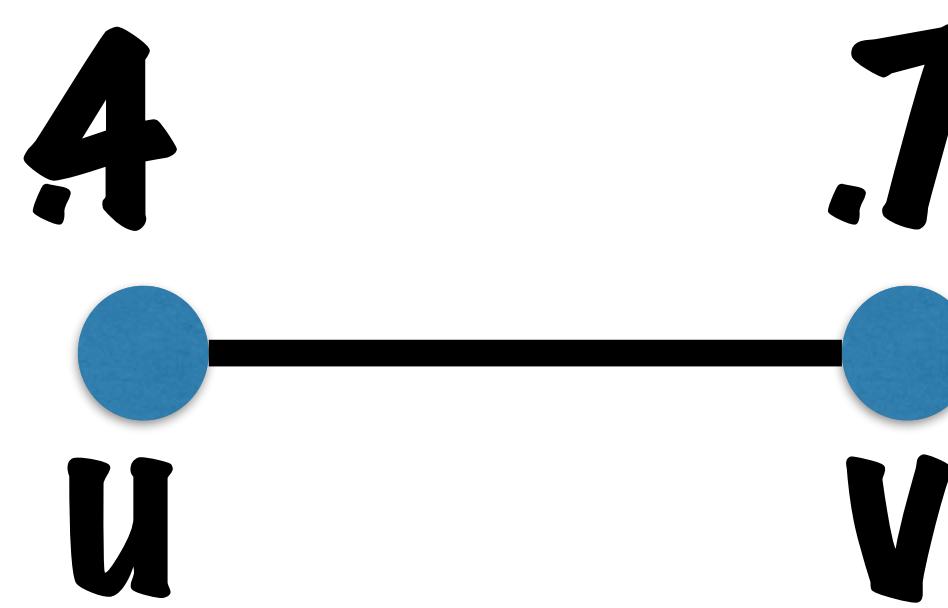
$$L = \{u : .5 < x_u^* < 1\}$$

$$S = \{u : 0 < x_u^* < .5\}$$

Observe:

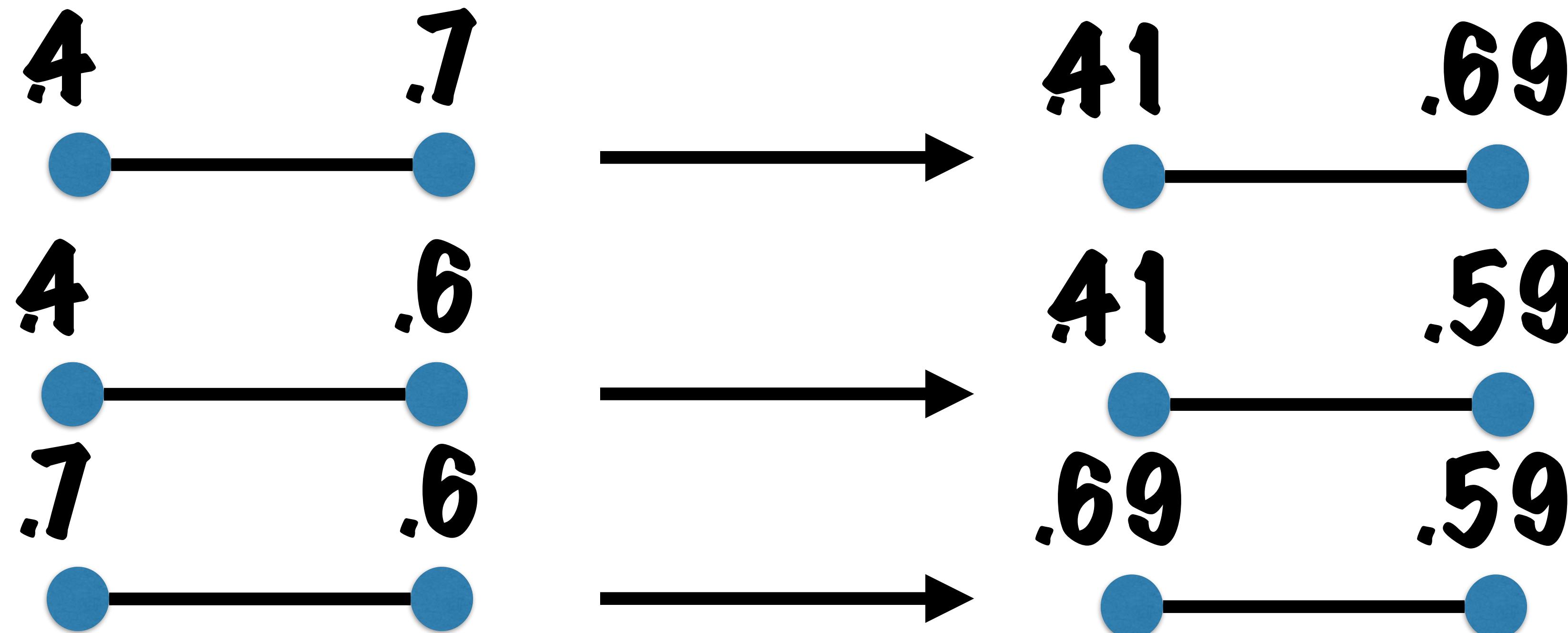
if u is in S and uv is an edge
then

v is in L or $x_v^* = 1$



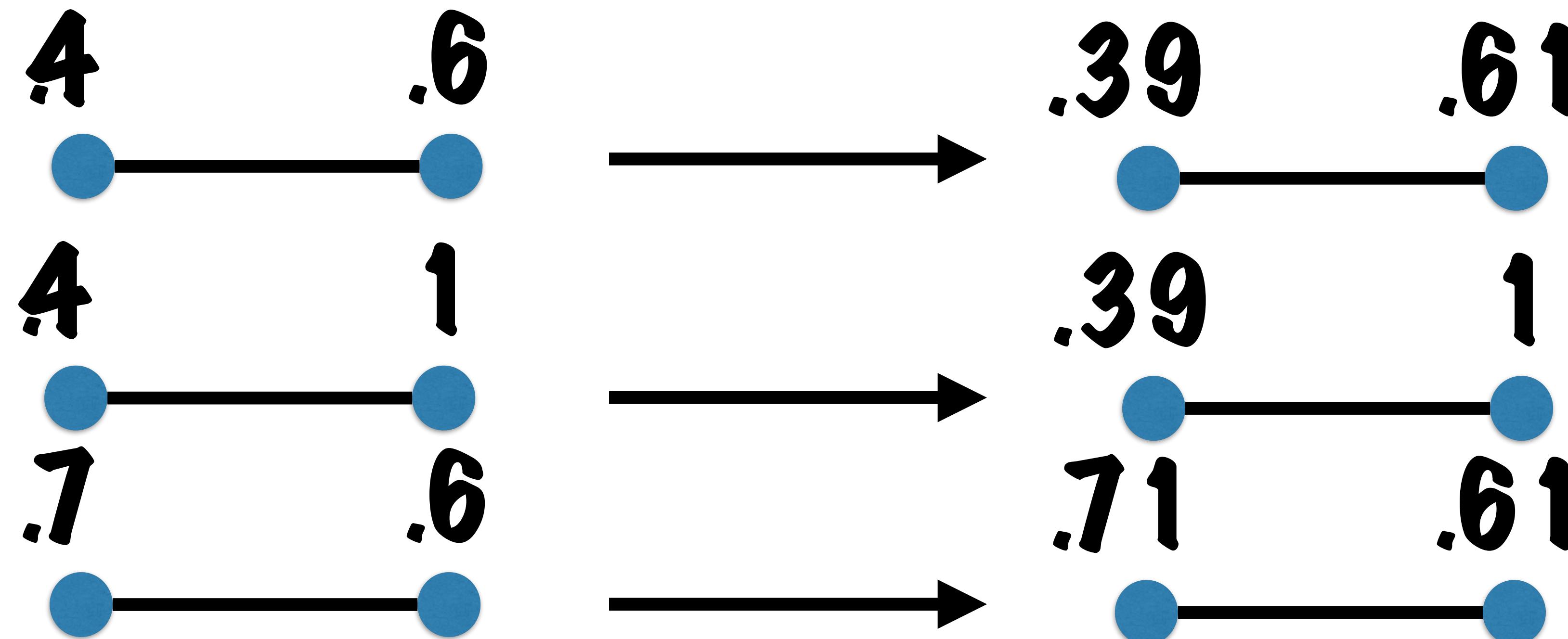
$$y_u = \begin{cases} x_u^* & \text{if } u \text{ frozen} \\ x_u^* + \epsilon & \text{if } u \in S \\ x_u^* - \epsilon & \text{if } u \in L \end{cases}$$

Observe: for ϵ small, it is still feasible.



$$z_u = \begin{cases} x_u^* & \text{if } u \text{ frozen} \\ x_u^* - \epsilon & \text{if } u \in S \\ x_u^* + \epsilon & \text{if } u \in L \end{cases}$$

Observe: for ϵ small, it is still feasible.



$$y_u = \begin{cases} x_u^* & \text{if } u \text{ frozen} \\ x_u^* + \epsilon & \text{if } u \in S \\ x_u^* - \epsilon & \text{if } u \in L \end{cases} \quad z_u = \begin{cases} x_u^* & \text{if } u \text{ frozen} \\ x_u^* - \epsilon & \text{if } u \in S \\ x_u^* + \epsilon & \text{if } u \in L \end{cases}$$

Since y feasible and x^* optimal: $\sum w_u y_u \geq \sum w_u x_u^*$

Since z feasible and x^* optimal: $\sum w_u z_u \geq \sum w_u x_u^*$

But observe: $(\sum w_u y_u + \sum w_u z_u)/2 = \sum w_u x_u^*$

So:

$$\sum_u w_u y_u = \sum_u w_u z_u = \sum_u w_u x_u^*$$

y and z are also optimal solutions

increase ϵ until something happens:

$$y_u = \begin{cases} x_u^* & \text{if } u \text{ frozen} \\ x_u^* + \epsilon & \text{if } u \in S \\ x_v^* - \epsilon & \text{if } u \in L \end{cases}$$

reaches .5

reaches .5

$$x^* \leftarrow y \text{ or } z$$

Repeat...

$$z_u = \begin{cases} x_u^* & \text{if } u \text{ frozen} \\ x_u^* - \epsilon & \text{if } u \in S \\ x_v^* + \epsilon & \text{if } u \in L \end{cases}$$

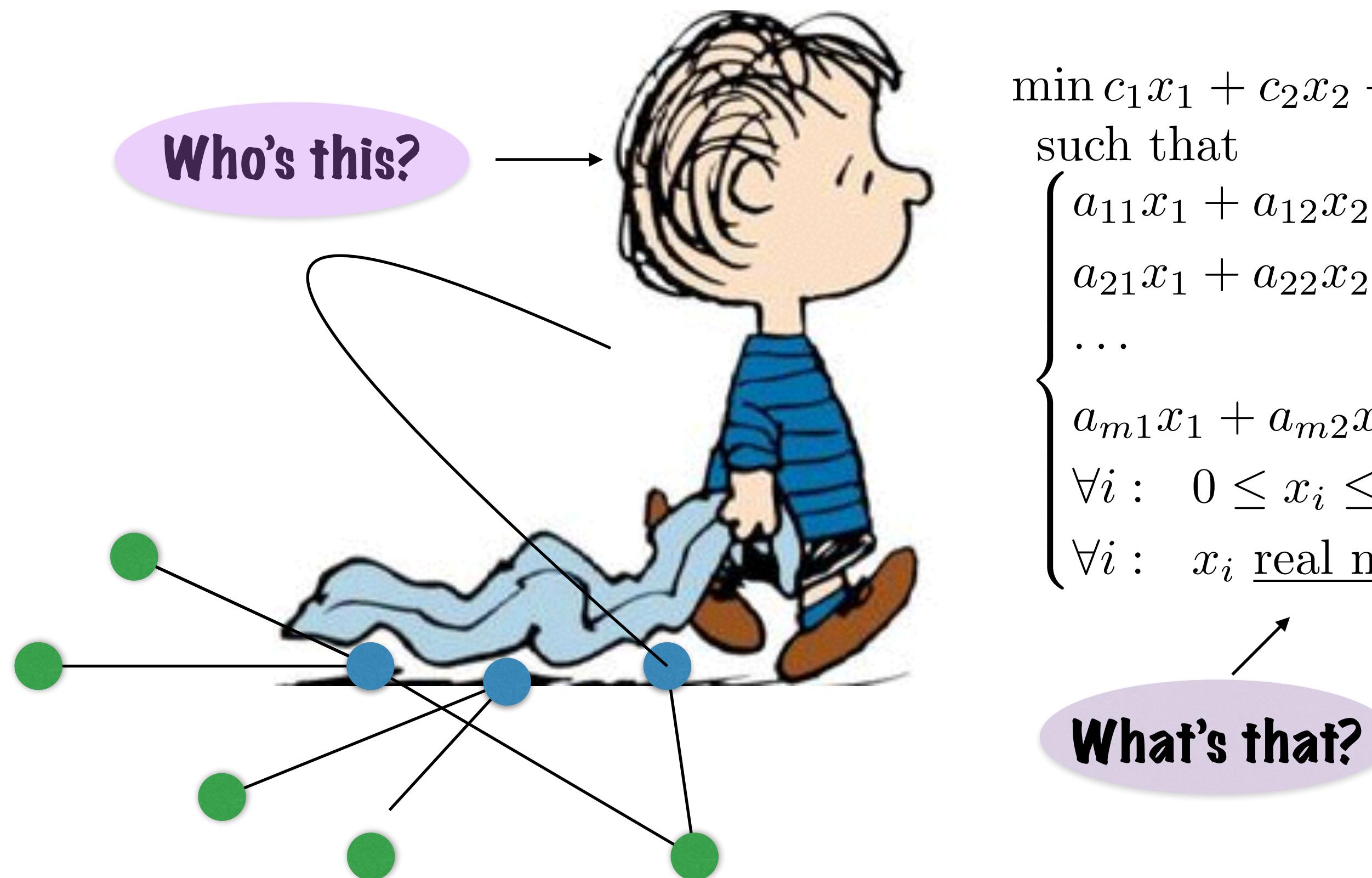
reaches 0

reaches 1

Freeze the variable that reached 0, .5, or 1

QED

Approximation algorithms, vertex cover, and linear programming



$$\begin{aligned} & \min c_1 x_1 + c_2 x_2 + \cdots + c_n x_n \\ \text{such that } & \left\{ \begin{array}{ll} a_{11} x_1 + a_{12} x_2 + \cdots + a_{1n} x_n & \geq b_1 \\ a_{21} x_1 + a_{22} x_2 + \cdots + a_{2n} x_n & \geq b_2 \\ \dots \\ a_{m1} x_1 + a_{m2} x_2 + \cdots + a_{mn} x_n & \geq b_m \\ \forall i : 0 \leq x_i \leq 1 \\ \forall i : x_i \text{ real number} \end{array} \right. \end{aligned}$$



([https://accounts.coursera.org/i zendesk/courserahelp?
return_to=https://learner.coursera.help/hc](https://accounts.coursera.org/i zendesk/courserahelp?return_to=https://learner.coursera.help/hc))

Congratulations! [View Final Grade](#)

 You've completed the course. (/learn/approximation-algorithms-part-1/home/welcome)

Vertex cover and Linear Programming



Claire Mathieu

We introduce the course topic by a typical example of a basic problem, called Vertex Cover, for which we will design and analyze a state-of-the-art approximation algorithm using two basic techniques, called

Introduction: a roadmap to this module and to this course



Lecture:
Introduction 8 min

(/learn/approximation-algorithms-part-1/lecture/oHoDL/lecture-introduction)



Slides

(/learn/approximation-algorithms-part-1/supplement/d7sPw/slides)

Practice Quiz: Quiz 1: P vs.

[Help Center](#)



NP review 3 questions

(/learn/approximation-algorithms-part-1/quiz/tVUPt/quiz-1-p-vs-np-review)



All slides for all chapters of
Approx Algs part 1

(/learn/approximation-algorithms-part-1/supplement/2DS8y/all-slides-for-all-chapters-of-approx-algs-part-1)



Attempt to upload slides
in Keynote format

(/learn/approximation-algorithms-part-1/supplement/TILu1/attempt-to-upload-slides-in-keynote-format)

What is vertex cover? Some examples



Lecture: Definition 5 min

(/learn/approximation-algorithms-part-1/lecture/KR2GV/lecture-definition)



Slides

(/learn/approximation-algorithms-part-1/supplement/6Wf3Q/slides)



Practice Quiz: Quiz

2 4 questions

(/learn/approximation-algorithms-part-1/quiz/x2CUx/quiz-2)

An integer program for vertex cover



Lecture: Integer
program 7 min

(/learn/approximation-algorithms-part-1/lecture/EYIrF/lecture-integer-program)



Slides

(/learn/approximation-algorithms-part-1/supplement/AJTrJ/slides)



Practice Quiz: Quiz

3 2 questions

(/learn/approximation-algorithms-part-1/quiz/52pwn/quiz-3)

Integer programming and linear programming

Lecture: A linear



programming
relaxation 6 min

(/learn/approximation-algorithms-part-1/lecture/jZ93B/lecture-a-linear-programming-relaxation)



Slides

(/learn/approximation-algorithms-part-1/supplement/R9AYt/slides)



Practice Quiz: Quiz

4 3 questions

(/learn/approximation-algorithms-part-1/quiz/nSE9P/quiz-4)

An approximation algorithm for vertex cover



Lecture: Approximation
algorithm 6 min

(/learn/approximation-algorithms-part-1/lecture/76qrk/lecture-approximation-algorithm)



Slides

(/learn/approximation-algorithms-part-1/supplement/SaPYJ/slides)

Practice Quiz: Quiz



5 2 questions

(/learn/approximation-algorithms-part-1/quiz/lOKFi/quiz-5)

Analysis: how good is our algorithm ?



Lecture: Analysis 7 min

(/learn/approximation-algorithms-part-1/lecture/ofAi1/lecture-analysis)



Slides

(/learn/approximation-algorithms-part-1/supplement/4Fu5l/slides)

**Practice Quiz: Quiz**

6 2 questions

(/learn/approximation-algorithms-part-1/quiz/UhCno/quiz-6)

General facts



Lecture: General facts 6 min

(/learn/approximation-algorithms-part-1/lecture/GeC7c/lecture-general-facts)



Slides

(/learn/approximation-algorithms-part-1/supplement/O5FQE/slides)

**Practice Quiz: Quiz**

7 2 questions

(/learn/approximation-algorithms-part-1/quiz/fr8Mj/quiz-7)

Exercises

 Practice Exercises

(/learn/approximation-algorithms-part-1/supplement/Djmm/practice-exercises)

- ✓ **Assignment:** Peer Graded
 - Assignment 1 30 min

(/learn/approximation-algorithms-part-1/peer/gZbt1/peer-graded-assignment-1)

- ✓ **Review Classmates:** Peer
 - Graded Assignment 1 30 min

(/learn/approximation-algorithms-part-1/peer/gZbt1/peer-graded-assignment-1/give-feedback)

 PDF version of the peer-graded assignment

(/learn/approximation-algorithms-part-1/supplement/aZr8k/pdf-version-of-the-peer-graded-assignment)

- ▶ Half integrality (7:35 bug, fixed in pdf slides) 10 min

(/learn/approximation-algorithms-part-1/lecture/FSr9h/half-integrality-7-35-bug-fixed-in-pdf-slides)

 Half-integrality slides

(/learn/approximation-algorithms-part-1/supplement/4oH25/half-integrality-slides)

- All slides together in one file

(/learn/approximation-algorithms-part-1/supplement/tav8Q/all-slides-together-in-one-file)

A $3/2$ -approximation for 4-colourable Graphs. In this exercise, we propose to derive a $3/2$ -approximation algorithm for a more restricted class of graphs. We recall that a graph is said to be *4-colourable* if given a set of 4 colours, it is possible to assign a colour of the set to each vertex in such a way that for each edge (u, v) of the graph, u and v receive different colours. A 4-colouring of G is an assignment of colours to the vertices of G such that for each edge (u, v) of the graph, u and v receive different colours. Consider the Linear Program for Vertex Cover that was described during the lectures and assume that we obtained a solution for this program such that the value of each variable is either 0, $1/2$ or 1. Namely, we have an **optimal** fractional assignment of the variables X such that for each $x_v \in X$, $x_v \in \{0, 1/2, 1\}$. We denote by $\text{val}(X)$ the objective value of the assignment X for the LP.¹

We consider a 4-colouring C of the vertices of G . Let $V^{1/2} = \{v \mid x_v = 1/2\}$, i.e: the set of vertices v such that $x_v = 1/2$ and $V^1 = \{v \mid x_v = 1\}$, i.e the set of vertices v such that $x_v = 1$. Similarly, $V^0 = \{v \mid x_v = 0\}$. Moreover let $V_0^{1/2}, V_1^{1/2}, V_2^{1/2}, V_3^{1/2}$ the set of vertices of V that have colour 0, 1, 2, and 3 respectively in C , i.e: $V_0^{1/2} = \{v \mid x_v = 1/2, C(v) = 0\}$. Finally, we assume that $|V_0^{1/2}| \leq |V_1^{1/2}| \leq |V_2^{1/2}| \leq |V_3^{1/2}|$.

A Rounding Procedure. We propose to define a rounding procedure for this assignment. We build a solution S . For each variable $x_v = 1$, v is added to the solution S . $x_v = 1/2$ and such that $C(v) \neq 3$, v is added to S . Otherwise, v is discarded.

Approximation Ratio.

Question 1. Give a relation between the value $\text{val}(X)$ the cardinality of the sets $V^{1/2}$ and V^1 .

Question 2. Give a tight lower bound on the cardinality of $V_3^{1/2}$ based on the cardinality of $V^{1/2}$.

Question 3. Deduce from Question 2 an upper bound on the cardinality of $|V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}|$ based on the cardinality of $V^{1/2}$.

Question 4. Combine Questions 1 and 3 to give an upper bound on the number of vertices in S based on $\text{val}(X)$.

Question 5. Combine Question 4 and the property of $\text{val}(X)$ to conclude on the approximation ratio of the rounding procedure. Recall that $\text{val}(X)$ is the value of the optimal fractional solution to the LP.

Correctness We now show that S is a correct vertex cover. Namely, we want to show that for each edge (u, v) , u or v (or both of them) are in S . We will proceed by contradiction and assume that neither u or v are in S .

Question 6. Suppose that $u \in V^0$, to which set does v belong? Recall that X is a solution to the LP.

Question 7. Deduce from the previous question to which set do u and v be-

¹such a solution is an extreme point solution of the LP and can be found in polynomial time.

long.

Question 8. If u and v belong to $V^{1/2}$, to which subset of V do they belong if they do not belong to S ?

Question 9. Recall that C is a 4-colouring. Explain the contradiction.

Question 10. Give an example of a well-known class of graph that is 4-colourable.

Course Home (/learn/approximation-algorithms-part-1/home/welcome) > Week 1 (/learn/approximation-algorithms-part-1/home/week/1) > Ex

Assignment: Peer Graded Assignment 1

Review classmates to see your grade

Your grade is ready, but you have not reviewed enough classmates yet.

Review more classmates to see your grade.

[Review a Classmates' Work \(/learn/approximation-algorithms-part-1/peer/gZbt1/peer-graded-assignment-1/give-feedback\)](#)

Instructions (/learn/approximation-algorithms-part-1/peer/gZbt1/peer-graded-assignment-1)

My submission (/learn/approximation-algorithms-part-1/peer/gZbt1/peer-graded-assignment-1/submit)

Discussions (/learn/approximation-algorithms-part-1/peer/gZbt1/peer-graded-assignment-1/discussions)

Thank you for completing a peer reviewed assignment! Please take this survey to help us improve your experience.

(<https://www.surveymonkey.com/r/Z8KFSCC>)

c=assignmentId%3DqFE_lprQEeW3phiGMVrfMw%401&c=courseId%3DbnQLDSclEeWbiBJCM9ziNQ&c=itemId%3DgZbt1&c=submissionId%3D2bUYAJ_hEeWaZwpL

A $\frac{3}{2}$ -Approximation Algorithm for Vertex Cover for the Planar Graphs

December 11, 2015

Shareable Link (https://www.coursera.org/learn/approximation-algorithms-part-1/peer/gZbt1/peer-graded-assignment-1/review/2bUYAJ_hEeWaZwpUNJJ91w)

In this exercise, we propose to derive a $\frac{3}{2}$ -approximation algorithm for a more restricted class of graphs. We recall that a graph is said to be *4-colourable* if given a set of 4 colours, it is possible to assign a colour of the set to each vertex in such a way that for each edge (u, v) of the graph, u and v receive different colours.

A 4-colouring of G is an assignment of colours to the vertices of G such that for each edge (u, v) of the graph, u and v receive different colours.

Consider the Linear Program for Vertex Cover that was described during the lectures and assume that we obtained a solution for this program such that the value of each variable is either 0, $1/2$ or 1. Namely, we have an *optimal/fractional* assignment of the variables X such that for each $x_v \in X$, $x_v \in \{0, 1/2, 1\}$. We denote by $\text{val}(X)$ the objective value of the assignment X for the LP.

We consider a 4-colouring C of the vertices of G .

Let $V^{1/2} = \{v \mid x_v = 1/2\}$, i.e., the set of vertices v such that $x_v = 1/2$ and

$V^1 = \{v \mid x_v = 1\}$, i.e., the set of vertices v such that $x_v = 1$. Similarly,

$V^0 = \{v \mid x_v = 0\}$.

Moreover let $V_0^{1/2}, V_1^{1/2}, V_2^{1/2}, V_3^{1/2}$ the set of vertices of V that have colour 0, 1, 2, and 3 respectively in C , i.e., $V_0^{1/2} = \{v \mid x_v = 1/2, C(v) = 0\}$.

Finally, we assume that $|V_0^{1/2}| \leq |V_1^{1/2}| \leq |V_2^{1/2}| \leq |V_3^{1/2}|$.

A Rounding Procedure. We propose to define a rounding procedure for this assignment. We build a solution S .

For each variable $x_v = 1$, v is added to the solution S .

For each variable $x_v = 1/2$ and such that $C(v) \neq 3$, v is added to S . Otherwise, v is discarded.

Approximation Ratio.

Question 1. Give a relation between the value $\text{val}(X)$ and the cardinality of the sets $V^{1/2}$ and V^1 .

Question 2. Give a tight lower bound on the cardinality of $V_3^{1/2}$ based on the cardinality of $V^{1/2}$.

Question 3. Deduce from Question 2 an upper bound on the cardinality of $|V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}|$ based on the cardinality of $V^{1/2}$.

Question 4. Combine Questions 1 and 3 to give an upper bound on the number of vertices in S based on $\text{val}(X)$.

Question 5. Combine Question 4 and the property of $\text{val}(X)$ to conclude on the approximation ratio of the rounding procedure. Recall that $\text{val}(X)$ is the value of the optimal fractional solution to the LP.

Correctness.

We now show that S is a correct vertex cover. Namely, we want to show that for each edge (u, v) , u or v (or both of them) are in S . We will proceed by contradiction and assume that neither u or v are in S .

Question 6. Suppose that $u \in V^0$, to which set does v belong? Recall that X is a solution to the LP.

Question 7. Deduce from the previous question to which set do u and v belong.

Question 8. If u and v belong to $V^{1/2}$, to which subset of V do they belong if they do not belong to S ?

Question 9. Recall that C is a 4-colouring. Explain the contradiction.

Question 10. Give an example of a well-known class of graphs that is 4-colourable.

Test Exercise 1: Answers to the Questions

Approximation Ratio

Question 1. Give a relation between the value $\text{val}(X)$ and the cardinality of the sets $V^{1/2}$ and V^1 .

Answer: $\text{val}(X) = 1 \cdot |V^1| + 0 \cdot |V^0| + \frac{1}{2} \cdot |V^{1/2}| = |V^1| + \frac{1}{2} \cdot |V^{1/2}|$.

Question 2. Give a tight lower bound on the cardinality of $V_3^{1/2}$ based on the cardinality of $V^{1/2}$.

Answer: $|V^{1/2}| = |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| + |V_3^{1/2}| \leq |V_3^{1/2}| + |V_3^{1/2}| + |V_3^{1/2}| + |V_3^{1/2}| = 4 \cdot |V_3^{1/2}| \Rightarrow |V_3^{1/2}| \geq \frac{1}{4} \cdot |V^{1/2}|$.

Question 3. Deduce from Question 2 an upper bound on the cardinality of $|V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}|$ based on the cardinality of $V^{1/2}$.

Answer: $|V^{1/2}| = |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| + |V_3^{1/2}| \geq |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| + \frac{1}{4} \cdot |V^{1/2}| \Rightarrow |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| \leq \frac{3}{4} \cdot |V^{1/2}|$.

Question 4. Combine Questions 1 and 3 to give an upper bound on the number of vertices in S based on $\text{val}(X)$.

Answer: Number of vertices in $S = |V^1| + |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| \leq |V^1| + \frac{3}{4} \cdot |V^{1/2}| = |V^1| + \frac{3}{2} \cdot (\text{val}(X) - |V^1|) = \frac{3}{2} \cdot \text{val}(X) - \frac{1}{2} \cdot |V^1| \Rightarrow$
Number of vertices in $S \leq \frac{3}{2} \cdot \text{val}(X)$.

Question 5. Combine Question 4 and the property of $\text{val}(X)$ to conclude on the approximation ratio of the rounding procedure. Recall that $\text{val}(X)$ is the value of the optimal fractional solution to the LP.

Answer: Number of Vertices in the Output $S \leq \frac{3}{2} \cdot OPT$, since $OPT = \text{val}(X)$, hence the approximation ratio is $\frac{3}{2}$.

Correctness.

Question 6. Suppose that $u \in V^0$, to which set does v belong? Recall that X is a solution to the LP.

Answer: $u \in V^0 \Leftrightarrow x_u = 0$ but (u, v) being an edge and $x_u, x_v \in X \Rightarrow x_u + x_v \geq 1 \wedge x_u, x_v \in \{0, \frac{1}{2}, 1\} \Rightarrow x_v = 1 \Leftrightarrow v \in V^1$.

Question 7. Deduce from the previous question to which set do u and v belong.

Answer: Since $x_v = 1 \Rightarrow v \in S$, a **contradiction**. Hence, if one of the vertices of an edge belongs to V^0 , the other one must belong to S . (Also, the trivial case: if one of the vertices of an edge belongs to V^1 , it's already in S). Hence if one of the vertices of an edge belongs to V^0 or V^1 , it will be in S , the edge being already covered.

Question 8. If u and v belong to $V^{1/2}$, to which subset of V do they belong if they do not belong to S ?

Answer: If any of u or v belongs to any of $V_0^{1/2}, V_1^{1/2}, V_2^{1/2}$, it will be in S and thus already covered. Only case remains to be considered is when $u, v \in V_3^{1/2}$. Hence, this is the only case when both u and v do not belong to S .

Question 9. Recall that C is a 4-colouring. Explain the **contradiction**.

Answer: $u, v \in V_3^{1/2} \Rightarrow C(u) = C(v) = 3$. Since C is a 4-colouring and u, v are **adjacent vertices**, they can't be colored with the **same color 3**, a **contradiction**. Hence, both the vertices of an edge can't belong to $V_3^{1/2}$, which means that the one that does not belong to it must already have been covered. Hence the proof.

Question 10. Give an example of a well-known class of graphs that is **4-colourable**.

Answer: **Planar graphs** (graphs that can be embedded in the plane, can be drawn in such a way that no edges cross each other) are **4-colourable**.

Answer to Question 1 has to be of the following form: $\text{val}(X) = |V^1| + |V^{1/2}| / 2$.

2 pts
Yes

0 pts
No



Answer to Question 2 has to be of the following form: $V_3^{1/2} \geq |V^{1/2}| / 4$.

2 pts
Yes

0 pts
No



Answer to Question 3 has to be of the following form: $|V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| \leq 3|V^{1/2}|/4$

- 2 pts
Yes
- 0 pts
No

Answer to question 4 has to be of the following form:

$$|S| = |V^1| + |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| \leq |V^1| + 3|V^{1/2}|/4 \leq 3(|V^1| + |V^{1/2}|/2)/2 \leq 3\text{val}(X)/2$$

- 3 pts
Yes
- 0 pts
No

Answer to question 5 has to be of the following form:

$$|S| \leq 3\text{val}(X)/2 \leq 3\text{OPT}/2$$

- 1 pt
Yes
- 0 pts
No

Answer to question 6 has to be of the following form:

v belongs to V^1 otherwise the constraint for edge (u, v) of the LP is not satisfied, i.e., $x_v + x_u < 1$.

- 2 pts
Yes
- 0 pts
No

Answer to question 7 has to be of the following form:

$u, v \in V^{1/2}$ since if one of them is in V^0 the other is in V^1 and so, it belongs to S .

- 2 pts
Yes
- 0 pts
No

Answer to question 8 has to be of the following form:

Since $u, v \notin S$, $u, v \in V_3^{1/2}$.

- 2 pts
Yes
- 0 pts
No

No

Answer to question 9 has to be of the following form:

Since C is a 4-colouring and there is an edge u, v , u and v receive different colour in C so they do not both belong to $V_3^{1/2}$.

- 3 pts
Yes
- 0 pts
No

Answer to question 10 has to be of the following form:

Planar graphs.

- 1 pt
Yes
- 0 pts
No

 Edit submission

Comments

Visible to classmates



share your thoughts...

◀ (/learn/approximation-algorithms-part-
(https://accounts.coursera.org/supplement/Djmn/practice-exercises)-https://learner.coursefiles.supplement/aZr8k/pdf-version-of-the-peer-graded-

▶ (/learn/approximation-algorithms-part-
(https://learner.coursefiles.supplement/aZr8k/pdf-version-of-the-peer-graded-
assignment)

Course Home (/learn/approximation-algorithms-part-1/home/welcome) > Week 1 (/learn/approximation-algorithms-part-1/home/week/1) > Exercise 1 (/learn/approximation-algorithms-part-1/home/week/1/exercise/1)

Review Classmates: Peer Graded Assignment 1

Review by December 16, 11:59 PM PT

Reviews 3 left to complete

Approximation Algorithms Part I Assignment 1

 by Robert Aftias
December 14, 2015

 like  Flag this submission

In this exercise, we propose to derive a 3/2-approximation algorithm for a more restricted class of graphs. We recall that a graph is said to be *4-colourable* if given a set of 4 colours, it is possible to assign a colour of the set to each vertex in such a way that for each edge (u, v) of the graph, u and v receive different colours.

A 4-colouring of G is an assignment of colours to the vertices of G such that for each edge (u, v) of the graph, u and v receive different colours.

Consider the Linear Program for Vertex Cover that was described during the lectures and assume that we obtained a solution for this program such that the value of each variable is either 0, 1/2 or 1. Namely, we have an *optimal* fractional assignment of the variables X such that for each $x_v \in X$, $x_v \in \{0, 1/2, 1\}$. We denote by $\text{val}(X)$ the objective value of the assignment X for the LP.

We consider a 4-colouring C of the vertices of G .

Let $V^{1/2} = \{v \mid x_v = 1/2\}$, i.e., the set of vertices v such that $x_v = 1/2$ and

$V^1 = \{v \mid x_v = 1\}$, i.e., the set of vertices v such that $x_v = 1$. Similarly,

$V^0 = \{v \mid x_v = 0\}$.

Moreover let $V_0^{1/2}, V_1^{1/2}, V_2^{1/2}, V_3^{1/2}$ the set of vertices of V that have colour 0, 1, 2, and 3 respectively in C , i.e., $V_0^{1/2} = \{v \mid x_v = 1/2, C(v) = 0\}$.

Finally, we assume that $|V_0^{1/2}| \leq |V_1^{1/2}| \leq |V_2^{1/2}| \leq |V_3^{1/2}|$.

A Rounding Procedure. We propose to define a rounding procedure for this assignment. We build a solution S .

For each variable $x_v = 1$, v is added to the solution S .

For each variable $x_v = 1/2$ and such that $C(v) \neq 3$, v is added to S . Otherwise, v is discarded.

Approximation Ratio.

Question 1. Give a relation between the value $\text{val}(X)$ and the cardinality of the sets $V^{1/2}$ and V^1 .

Question 2. Give a tight lower bound on the cardinality of $V_3^{1/2}$ based on the cardinality of $V^{1/2}$.

Question 3. Deduce from Question 2 an upper bound on the cardinality of $|V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}|$ based on the cardinality of $V^{1/2}$.

Question 4. Combine Questions 1 and 3 to give an upper bound on the number of vertices in S based on $\text{val}(X)$.

Question 5. Combine Question 4 and the property of $\text{val}(X)$ to conclude on the approximation ratio of the rounding procedure. Recall that $\text{val}(X)$ is the value of the optimal fractional solution to the LP.

Correctness.

We now show that S is a correct vertex cover. Namely, we want to show that for each edge (u, v) , u or v (or both of them) are in S . We will proceed by contradiction and assume that neither u or v are in S .

Question 6. Suppose that $u \in V^0$, to which set does v belong? Recall that X is a solution to the LP.

Question 7. Deduce from the previous question to which set do u and v belong.

Question 8. If u and v belong to $V^{1/2}$, to which subset of V do they belong if they do not belong to S ?

Question 9. Recall that C is a 4-colouring. Explain the contradiction.

Question 10. Give an example of a well-known class of graphs that is 4-colourable.

1)

$\text{val}(X) = 2|V^{1/2}| + |V^1|$

2)

$|V_3^{1/2}| \geq \frac{|V^{1/2}|}{4}$ as it is the largest colour component, and $|V^{1/2}| = |V_1^{1/2}| + |V_2^{1/2}| + |V_3^{1/2}| + |V_4^{1/2}|$

3)

$|V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| \leq \frac{3|V^{1/2}|}{4}$

4)

$|S| \leq |V^1| + 2\left(\frac{3|V^{1/2}|}{4}\right)$ since we know that nothing in $V_3^{1/2}$ is added to S .

From this we can see that $|S| \leq \frac{3}{2} \text{val}(X)$

5)

Since a solution to the ILP formulation of vertex cover satisfies the LP formulation, then the optimal solution is always at most $\text{val}(X)$, which means that

$|S|$ is within a $\frac{3}{2}$ factor of the optimal solution by looking at question 4's inequality.

6)

If $u \in V^0$, then in order to satisfy $x_u + x_v \geq 1$ then $x_v \geq 1$ and thus $x_v \in V^1$.

7)

At most one of u, v belongs to V^0 , and if one does, the other must belong to V^1 .

The possibilities are V^0/V^1 , V^1/V^1 , $V^1/V^{1/2}$, or $V^{1/2}/V^{1/2}$,

of which all but $V^{1/2}/V^{1/2}$ will result in uv being covered by S

even without the $V^{1/2}$ rule in the rounding.

8)

By the rounding procedure definition they must both belong to $V_3^{1/2}$, as all other subsets of $V^{1/2}$ are included in S , so this is the last case we need to check to verify that S is a vertex cover of G .

9)

This is a contradiction as it implies $C(u) = 3 = C(v)$ for some $u, v \in V(G)$ where $uv \in E(G)$, which is not allowed in a proper vertex-colouring of G , thus S is a valid vertex cover of G , as no case that would arise from an LP solution would result in an uncovered edge.

10)

Planar graphs are a well-known and non-trivial class of graphs that are 4-colourable. This was proven last century after being conjectured since before the last century.

Answer to Question 1 has to be of the following form: $\text{val}(X) = |V^1| + |V^{1/2}|/2$.

- 2 pts
Yes
- 0 pts
No

Answer to Question 2 has to be of the following form: $V_3^{1/2} \geq |V^{1/2}|/4$.

- 2 pts
Yes
- 0 pts
No

Answer to Question 3 has to be of the following form: $|V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| \leq 3|V^{1/2}|/4$

- 2 pts
Yes
- 0 pts
No

Answer to question 4 has to be of the following form:

$$|S| = |V^1| + |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| \leq |V^1| + 3|V^{1/2}|/4 \leq 3(|V^1| + |V^{1/2}|/2)/2 \leq 3\text{val}(X)/2$$

- 3 pts
Yes
- 0 pts
No

Help Center

Answer to question 5 has to be of the following form:

$$|S| \leq 3\text{val}(X)/2 \leq 3\text{OPT}/2$$

- 1 pt
Yes
- 0 pts
No

Answer to question 6 has to be of the following form:

v belongs to V^1 otherwise the constraint for edge (u, v) of the LP is not satisfied, i.e., $x_v + x_u < 1$.

- 2 pts
Yes
- 0 pts
No

Answer to question 7 has to be of the following form:

$u, v \in V^{1/2}$ since if one of them is in V^0 the other is in V^1 and so, it belongs to S .

- 2 pts
Yes
- 0 pts
No

Answer to question 8 has to be of the following form:

Since $u, v \notin S$, $u, v \in V_3^{1/2}$.

- 2 pts
Yes
- 0 pts
No

Answer to question 9 has to be of the following form:

Since C is a 4-colouring and there is an edge u, v , u and v receive different colour in C so they do not both belong to $V_3^{1/2}$.

- 3 pts
Yes
- 0 pts
No

Answer to question 10 has to be of the following form:

Planar graphs.

- 1 pt
Yes
- 0 pts
No

[Submit Review](#)

Comments

Visible to classmates



share your thoughts...

◀ (/learn/approximation-algorithms-part-
(https://accounts.coursera.org/supplement/Djinn/practice-exercises)-https://learner.courses/145/supplement/aZr8k/pdf-version-of-the-peer-graded-assignment)

▶ (/learn/approximation-algorithms-part-
(https://learner.courses/145/supplement/aZr8k/pdf-version-of-the-peer-graded-assignment)

Course Home (/learn/approximation-algorithms-part-1/home/welcome) > Week 1 (/learn/approximation-algorithms-part-1/home/week/1) > Exercise 1 (/learn/approximation-algorithms-part-1/home/week/1/exercise/1)

Review Classmates: Peer Graded Assignment 1

Review by December 16, 11:59 PM PT

Reviews 1 left to complete

Assignment 1



by Weiran Huang
December 14, 2015

like Flag this submission

In this exercise, we propose to derive a 3/2-approximation algorithm for a more restricted class of graphs. We recall that a graph is said to be *4-colourable* if given a set of 4 colours, it is possible to assign a colour of the set to each vertex in such a way that for each edge (u, v) of the graph, u and v receive different colours.

A 4-colouring of G is an assignment of colours to the vertices of G such that for each edge (u, v) of the graph, u and v receive different colours.

Consider the Linear Program for Vertex Cover that was described during the lectures and assume that we obtained a solution for this program such that the value of each variable is either 0, 1/2 or 1. Namely, we have an *optimal* fractional assignment of the variables X such that for each $x_v \in X$, $x_v \in \{0, 1/2, 1\}$. We denote by $\text{val}(X)$ the objective value of the assignment X for the LP.

We consider a 4-colouring C of the vertices of G .

Let $V^{1/2} = \{v \mid x_v = 1/2\}$, i.e., the set of vertices v such that $x_v = 1/2$ and

$V^1 = \{v \mid x_v = 1\}$, i.e., the set of vertices v such that $x_v = 1$. Similarly,

$V^0 = \{v \mid x_v = 0\}$.

Moreover let $V_0^{1/2}, V_1^{1/2}, V_2^{1/2}, V_3^{1/2}$ the set of vertices of V that have colour 0, 1, 2, and 3 respectively in C , i.e., $V_0^{1/2} = \{v \mid x_v = 1/2, C(v) = 0\}$.

Finally, we assume that $|V_0^{1/2}| \leq |V_1^{1/2}| \leq |V_2^{1/2}| \leq |V_3^{1/2}|$.

A Rounding Procedure. We propose to define a rounding procedure for this assignment. We build a solution S .

For each variable $x_v = 1$, v is added to the solution S .

For each variable $x_v = 1/2$ and such that $C(v) \neq 3$, v is added to S . Otherwise, v is discarded.

Approximation Ratio.

Question 1. Give a relation between the value $\text{val}(X)$ and the cardinality of the sets $V^{1/2}$ and V^1 .

Question 2. Give a tight lower bound on the cardinality of $V_3^{1/2}$ based on the cardinality of $V^{1/2}$.

Question 3. Deduce from Question 2 an upper bound on the cardinality of $|V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}|$ based on the cardinality of $V^{1/2}$.

Question 4. Combine Questions 1 and 3 to give an upper bound on the number of vertices in S based on $\text{val}(X)$.

Question 5. Combine Question 4 and the property of $\text{val}(X)$ to conclude on the approximation ratio of the rounding procedure. Recall that $\text{val}(X)$ is the value of the optimal fractional solution to the LP.

Correctness.

We now show that S is a correct vertex cover. Namely, we want to show that for each edge (u, v) , u or v (or both of them) are in S . We will proceed by contradiction and assume that neither u or v are in S .

Question 6. Suppose that $u \in V^0$, to which set does v belong? Recall that X is a solution to the LP.

Question 7. Deduce from the previous question to which set do u and v belong.

Question 8. If u and v belong to $V^{1/2}$, to which subset of V do they belong if they do not belong to S ?

Question 9. Recall that C is a 4-colouring. Explain the contradiction.

Question 10. Give an example of a well-known class of graphs that is 4-colourable.

$$\text{Q1. } \text{val}(X) = 0.5|V^{1/2}| + |V^1|.$$

$$\text{Q2. } |V^{1/2}| = |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| + |V_3^{1/2}| \leq 4|V_3^{1/2}|. \text{ Thus } |V_3^{1/2}| \geq |V^{1/2}|/4.$$

$$\text{Q3. } |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| = |V^{1/2}| - |V_3^{1/2}| \leq |V^{1/2}| - |V^{1/2}|/4 = \frac{3}{4}|V^{1/2}|.$$

Q4. $|S| = |V^1| + |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| \leq |V^1| + \frac{3}{4}|V^{1/2}| \leq \frac{3}{2}(0.5|V^{1/2}| + |V^1|) - 0.5|V^1| \leq \frac{3}{2}val(X)$.

Q5. $val(X)$ is the optimal of LP, which should not be greater than the optimal of half-IP. Thus $|S| \leq val(X) \leq OPT_{half-IP}$.

Help Center

Q6. v must belong to V^1 , otherwise $x_u + x_v \geq 1$ violates.

Q7. u and v belong to S .

Q8. They belong to $V_3^{1/2}$ if they do not belong to S .

Q9. If u and v both not belong to S , they are both belong to $V_3^{1/2}$, which means that they can both be colored by color 3. However, they cannot be colored by the same color since for each edge (u,v) of the graph, u and v receive different colours.

Q10. planar graph.

Answer to Question 1 has to be of the following form: $val(X) = |V^1| + |V^{1/2}|/2$.

- 2 pts
Yes
- 0 pts
No

Answer to Question 2 has to be of the following form: $V_3^{1/2} \geq |V^{1/2}|/4$.

- 2 pts
Yes
- 0 pts
No

Answer to Question 3 has to be of the following form: $|V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| \leq 3|V^{1/2}|/4$

- 2 pts
Yes
- 0 pts
No

Answer to question 4 has to be of the following form:

$$|S| = |V^1| + |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| \leq |V^1| + 3|V^{1/2}|/4 \leq$$

$$3(|V^1| + |V^{1/2}|/2)/2 \leq 3val(X)/2$$

- 3 pts
Yes
- 0 pts
No

Answer to question 5 has to be of the following form:

$$|S| \leq 3val(X)/2 \leq 3OPT/2$$

- 1 pt
Yes
- 0 pts
No

Answer to question 6 has to be of the following form:

v belongs to V^1 otherwise the constraint for edge (u,v) of the LP is not satisfied, i.e., $x_v + x_u < 1$.

- 2 pts
Yes
- 0 pts
No

Answer to question 7 has to be of the following form:

$u, v \in V^{1/2}$ since if one of them is in V^0 the other is in V^1 and so, it belongs to S .

- 2 pts
Yes
- 0 pts
No

Answer to question 8 has to be of the following form:

Since $u, v \notin S$, $u, v \in V_3^{1/2}$.

- 2 pts
Yes
- 0 pts
No

Answer to question 9 has to be of the following form:

Since C is a 4-colouring and there is an edge u, v , u and v receive different colour in C so they do not both belong to $V_3^{1/2}$.

- 3 pts
Yes
- 0 pts
No

Answer to question 10 has to be of the following form:

Planar graphs.

- 1 pt
Yes
- 0 pts
No

[Submit Review](#)

Comments

Visible to classmates



share your thoughts...

◀ (/learn/approximation-algorithms-part-
(https://account.coursera.org/supplement/Djinn/practice-exercises)=https://learner.coursefiles.supplement/aZr8k/pdf-version-of-the-peer-graded-

▶ (/learn/approximation-algorithms-part-
(https://learner.coursefiles.supplement/aZr8k/pdf-version-of-the-peer-graded-
assignment)

Question 1:

$$val(X) = \frac{1}{2} |V^{1/2}| + |V^1|$$

Question 2:

Because

$$|V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| + |V_3^{1/2}| = 1$$

And

$$|V_0^{1/2}| \leq |V_1^{1/2}| \leq |V_2^{1/2}| \leq |V_3^{1/2}|$$

We have

$$4 |V_3^{1/2}| \geq |V^{1/2}|$$

$$|V_3^{1/2}| \geq \frac{1}{4} |V^{1/2}|$$

Question 3:

Because

$$|V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| + |V_3^{1/2}| = |V^{1/2}|$$

We have

$$|V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| \leq \frac{3}{4} |V^{1/2}|$$

Question 4:

$$val(X) = \frac{1}{2} |V^{1/2}| + |V^1| \geq \frac{1}{2} * \frac{4}{3} * (|V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}|) + |V^1| \geq \frac{2}{3} |S|$$

$$|S| \leq 1.5 val(X)$$

Question 5:

Because $val(X)$ is obtained with the real number constraint,

$$val(X) \leq Optimal\ solution$$

We have

$$|S| \leq 1.5 Optimal\ solution$$

Question 6:

$$v \in V^1$$

Question 7:

$$u \in V - S, v \in S$$

Question 8:

$$u \in V_3^{1/2} \text{ and } v \in V_3^{1/2}$$

Question 9:

u and v cannot be painted with the same color which is Color 3.

If $u \in V^{1/2}$ and $v \in V^{1/2}$, at most one of u and v can be painted with Color 3, that is to say, at least one of them is added to S.

Question 10:

A pentagon, that is a single cycle with 5 vertices.

Test Exercise 1: Answers to the Questions

In this exercise, we propose to derive a $3/2$ -approximation algorithm for a more restricted class of graphs. We recall that a graph is said to be *4-colourable* if given a set of 4 colours, it is possible to assign a colour of the set to each vertex in such a way that for each edge (u,v) of the graph, u and v receive different colours.

A *4-colouring* of G is an assignment of colours to the vertices of G such that for each edge (u,v) of the graph, u and v receive different colours.

Consider the Linear Program for Vertex Cover that was described during the lectures and assume that we obtained a solution for this program such that the value of each variable is either 0 , $\frac{1}{2}$ or 1 . Namely, we have an *optimal/fractional* assignment of the variables X such that for each $x_v \in X$, $x_v \in \{0, \frac{1}{2}, 1\}$. We denote by $val(X)$ the objective value of the assignment X for the LP.

We consider a *4-colouring* C of the vertices of G .

Let $V^{1/2} = \{v | x_v = \frac{1}{2}\}$, i.e., the set of vertices v such that $x_v = \frac{1}{2}$ and

$V^1 = \{v | x_v = 1\}$, i.e., the set of vertices v such that $x_v = 1$. Similarly,

$V^0 = \{v | x_v = 0\}$.

Moreover let $V_0^{1/2}, V_1^{1/2}, V_2^{1/2}, V_3^{1/2}$ the set of vertices of V that have colour 0, 1, 2, and 3 respectively in C , i.e., $V_0^{1/2} = \{v | x_v = \frac{1}{2}, C(v) = 0\}$.

Finally, we assume that $|V_0^{1/2}| \leq |V_1^{1/2}| \leq |V_2^{1/2}| \leq |V_3^{1/2}|$.

A Rounding Procedure. We propose to define a rounding procedure for this assignment. We build a solution S .

For each variable $x_v = 1$, v is added to the solution S .

For each variable $x_v = \frac{1}{2}$ and such that $C(v) \neq 3$, v is added to S . Otherwise, v is discarded.

Approximation Ratio.

- Question 1. Give a relation between the value $val(X)$ and the cardinality of the sets $V^{1/2}$ and V^1 .
 - $val(X) = 1 \cdot |V^1| + 0 \cdot |V^0| + \frac{1}{2} \cdot |V^{1/2}| = |V^1| + \frac{1}{2} \cdot |V^{1/2}|$.
- Question 2. Give a tight lower bound on the cardinality of $V_3^{1/2}$ based on the cardinality of $V^{1/2}$.
 - $|V^{1/2}| = |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| + |V_3^{1/2}| \leq |V_3^{1/2}| + |V_3^{1/2}| + |V_3^{1/2}| + |V_3^{1/2}| = 4 \cdot |V_3^{1/2}|$.
 $\Rightarrow |V_3^{1/2}| \geq \frac{1}{4} \cdot |V^{1/2}|$.
- Question 3. Deduce from Question 2 an upper bound on the cardinality of $|V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}|$ based on the cardinality of $V^{1/2}$.
 - $|V^{1/2}| = |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| + |V_3^{1/2}| \geq |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| + \frac{1}{4} \cdot |V^{1/2}|$.
 $\Rightarrow |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| \leq \frac{3}{4} \cdot |V^{1/2}|$.
- Question 4. Combine Questions 1 and 3 to give an upper bound on the number of vertices in S based on $val(X)$.
 - Number of vertices in S =
 $|V^1| + |V_0^{1/2}| + |V_1^{1/2}| + |V_2^{1/2}| \leq |V^1| + \frac{3}{4} \cdot |V^{1/2}| = |V^1| + \frac{3}{2} \cdot (val(X) - |V^1|) = \frac{3}{2} \cdot val(X) - \frac{1}{2} \cdot |V^1|$.
 \Rightarrow Number of vertices in $S \leq \frac{3}{2} \cdot val(X)$.

- Question 5. Combine Question 4 and the property of $\text{val}(X)$ to conclude on the approximation ratio of the rounding procedure. Recall that $\text{val}(X)$ is the value of the optimal fractional solution to the LP.

- Number of Vertices in the Output $S \leq \frac{3}{2} \cdot OPT$, since $OPT = \text{val}(X)$, hence the approximation ratio is $\frac{3}{2}$.

Correctness.

We now show that S is a correct vertex cover. Namely, we want to show that for each edge (u, v) , u or v (or both of them) are in S . We will proceed by contradiction and assume that neither u nor v are in S .

- Question 6. Suppose that $u \in V^0$, to which set does v belong? Recall that X is a solution to the LP.

- $u \in V^0 \Leftrightarrow x_u = 0$ but (u, v) being an edge and $x_u, x_v \in X \Rightarrow x_u + x_v \geq 1 \wedge x_u, x_v \in \{0, \frac{1}{2}, 1\} \Rightarrow x_v = 1 \Leftrightarrow v \in V^1$.

- Question 7. Deduce from the previous question to which set do u and v belong.

- Since $x_v = 1 \Rightarrow v \in S$, a contradiction. Hence, if one of the vertices of an edge belongs to V^0 , the other one must belong to S . (Also, the trivial case: if one of the vertices of an edge belongs to V^1 , it's already in S). Hence if one of the vertices of an edge belongs to V^0 or V^1 , it will be in S , the edge being already covered.

- Question 8. If u and v belong to $V^{1/2}$, to which subset of V do they belong if they do not belong to S ?

- Again if any of u or v belongs to any of $V_0^{1/2}, V_1^{1/2}, V_2^{1/2}$, it will be in S and covered. Only case remains to be considered is when $u, v \in V_3^{1/2}$. Hence, this is the only case when both u and v do not belong to S .

- Question 9. Recall that C is a 4-colouring. Explain the contradiction.

- $u, v \in V_3^{1/2} \Rightarrow C(u) = C(v) = 3$ Since C is a 4-colouring and u, v are adjacent vertices, they can't be colored with the same color 3, a contradiction. Hence both the vertices of an edge can't belong to $V_3^{1/2}$, which means that the one that does not belong to must already have been covered. Hence the proof.

- Question 10. Give an example of a well-known class of graphs that is 4-colourable.

- *Planar graphs* (graphs that can be embedded in the plane, can be drawn in such a way that no edges cross each other) are 4-colourable.

Knapsack and rounding



The knapsack problem: what
should you put in your
knapsack?

The knapsack problem

Given: capacity B knapsack, n items,
item i has size s_i and value v_i

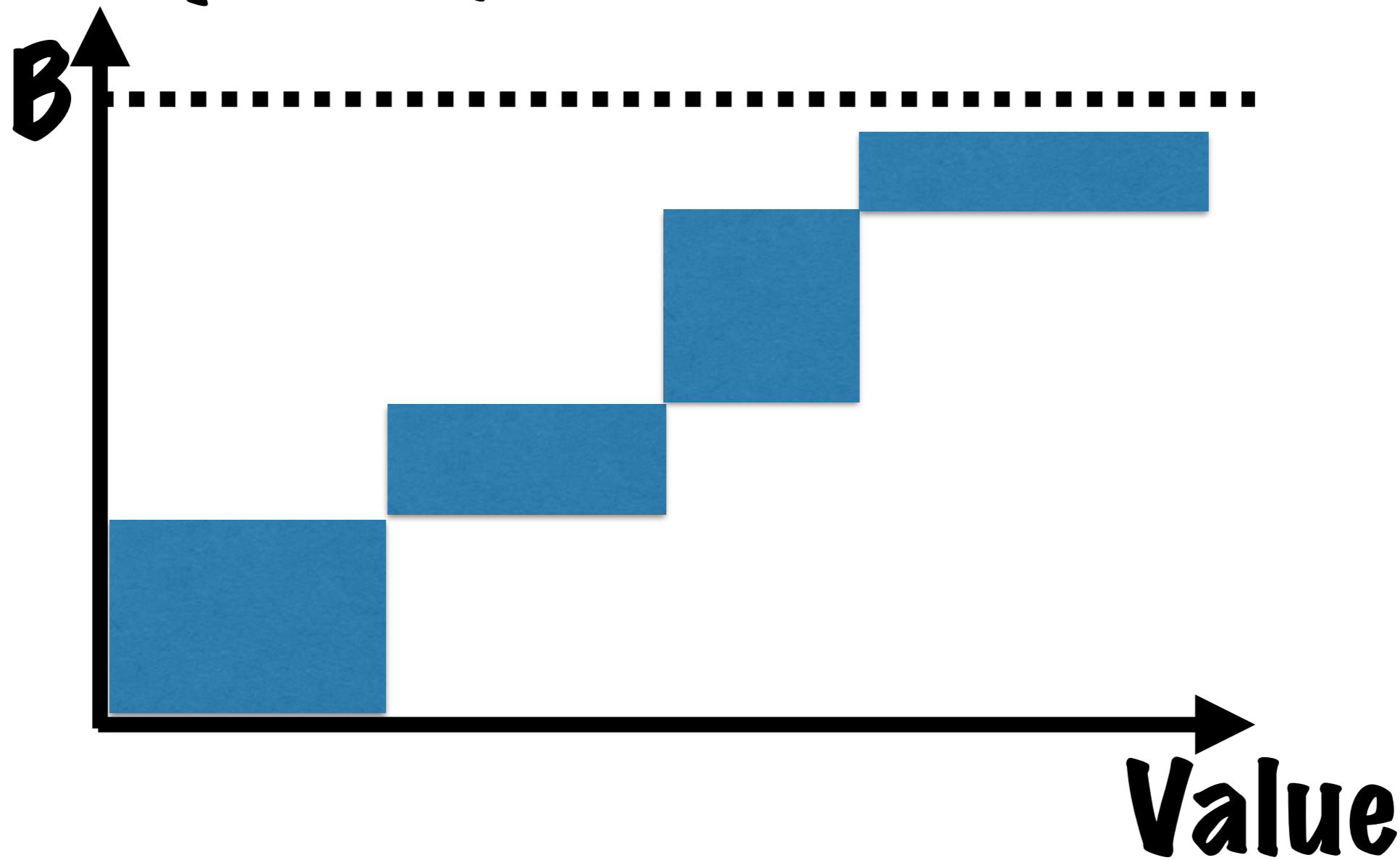
Place some items in knapsack
Maximize value

NP-hard



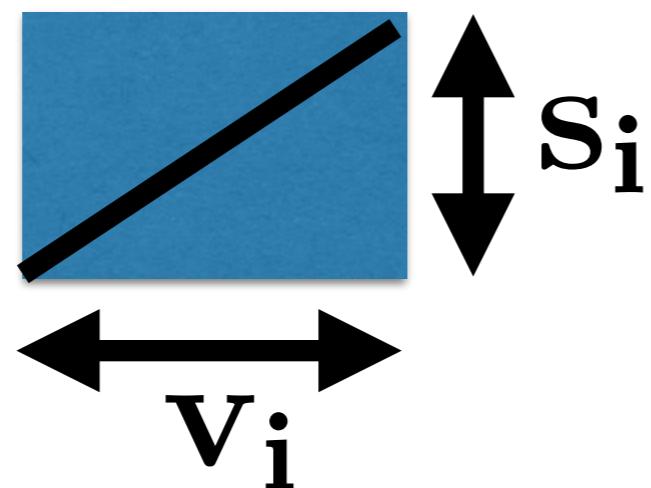
**A general recipe:
for intuition,
graphical representation**

Size, capacity



Desire: small size, high value

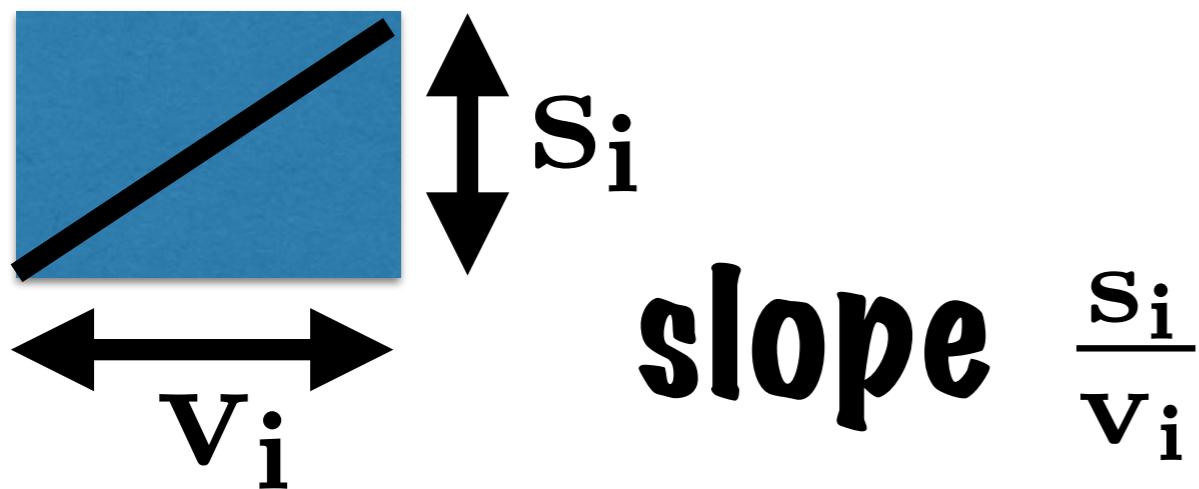
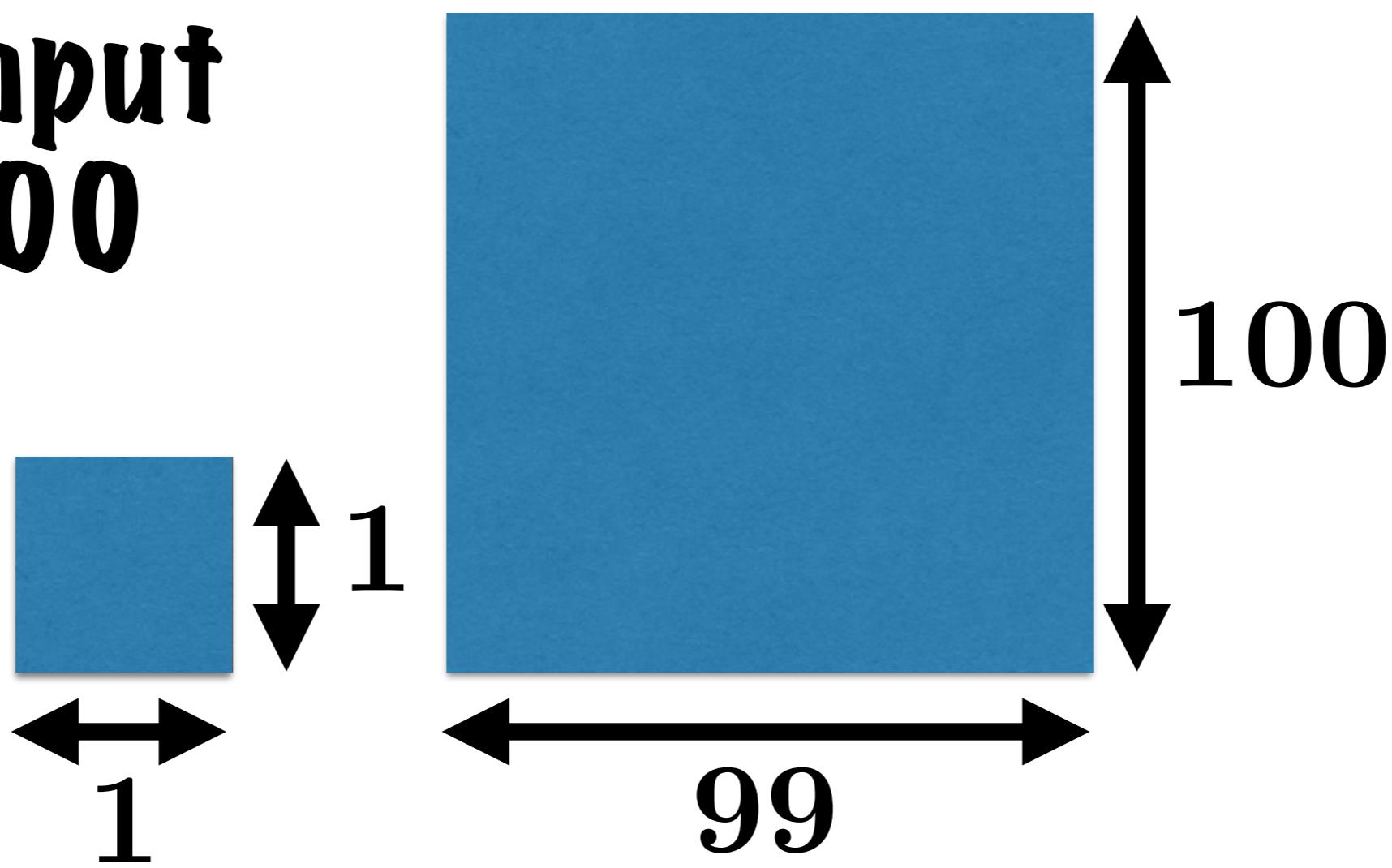
Naive greedy algorithm:
take by order of
increasing size/value



slope $\frac{s_i}{v_i}$

How good is Greedy?

Bad input
 $B=1\ 00$



Greedy is bad

**Another general recipe:
for intuition,
try special cases**

If all items have the same size...

Greedy is good.

If all items have the same value...

Greedy is good.

If all items have size=value...

Knapsack and rounding



Knapsack and rounding

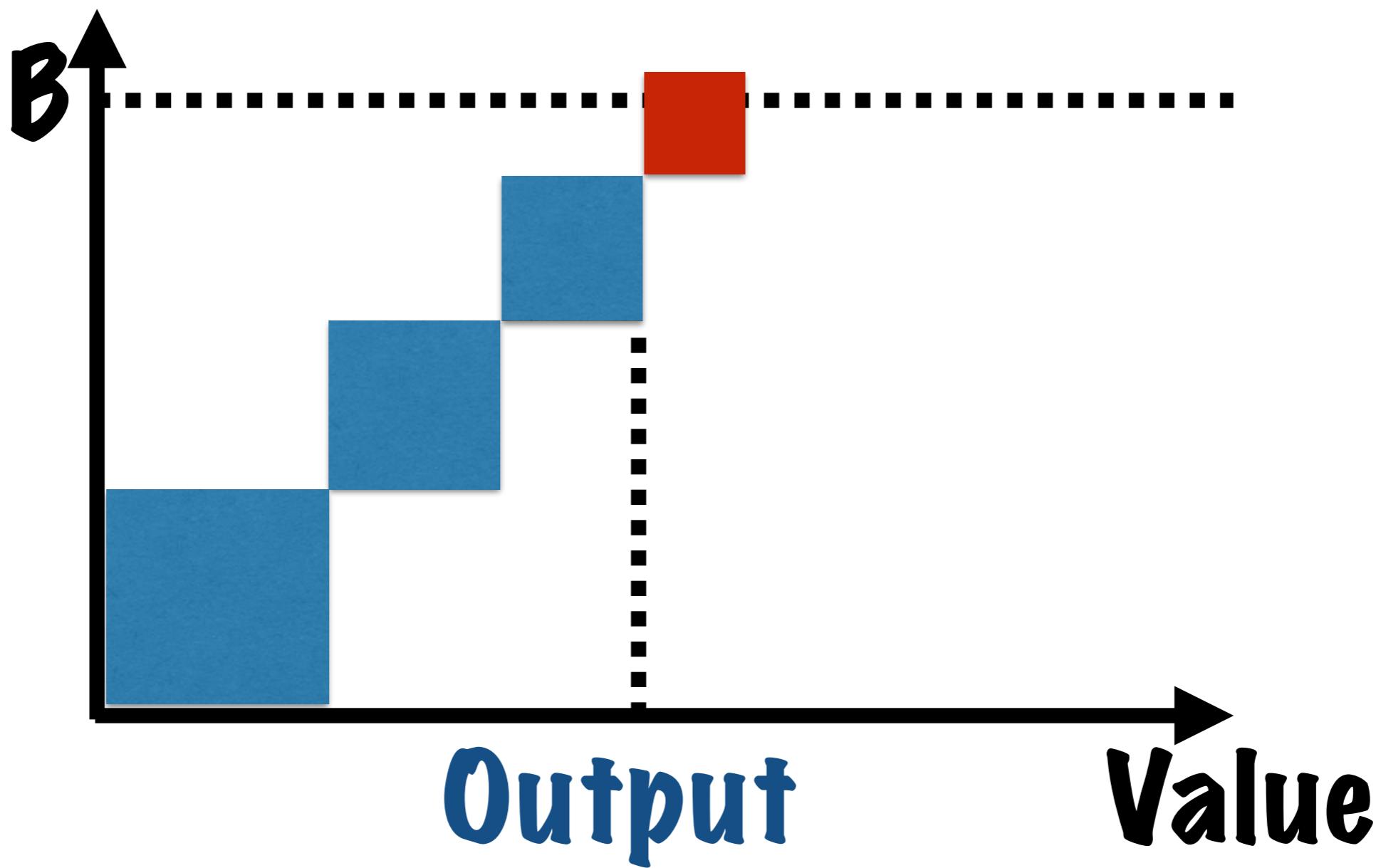


A greedy algorithm for special case size=value

Order items by decreasing value.

How good is that?

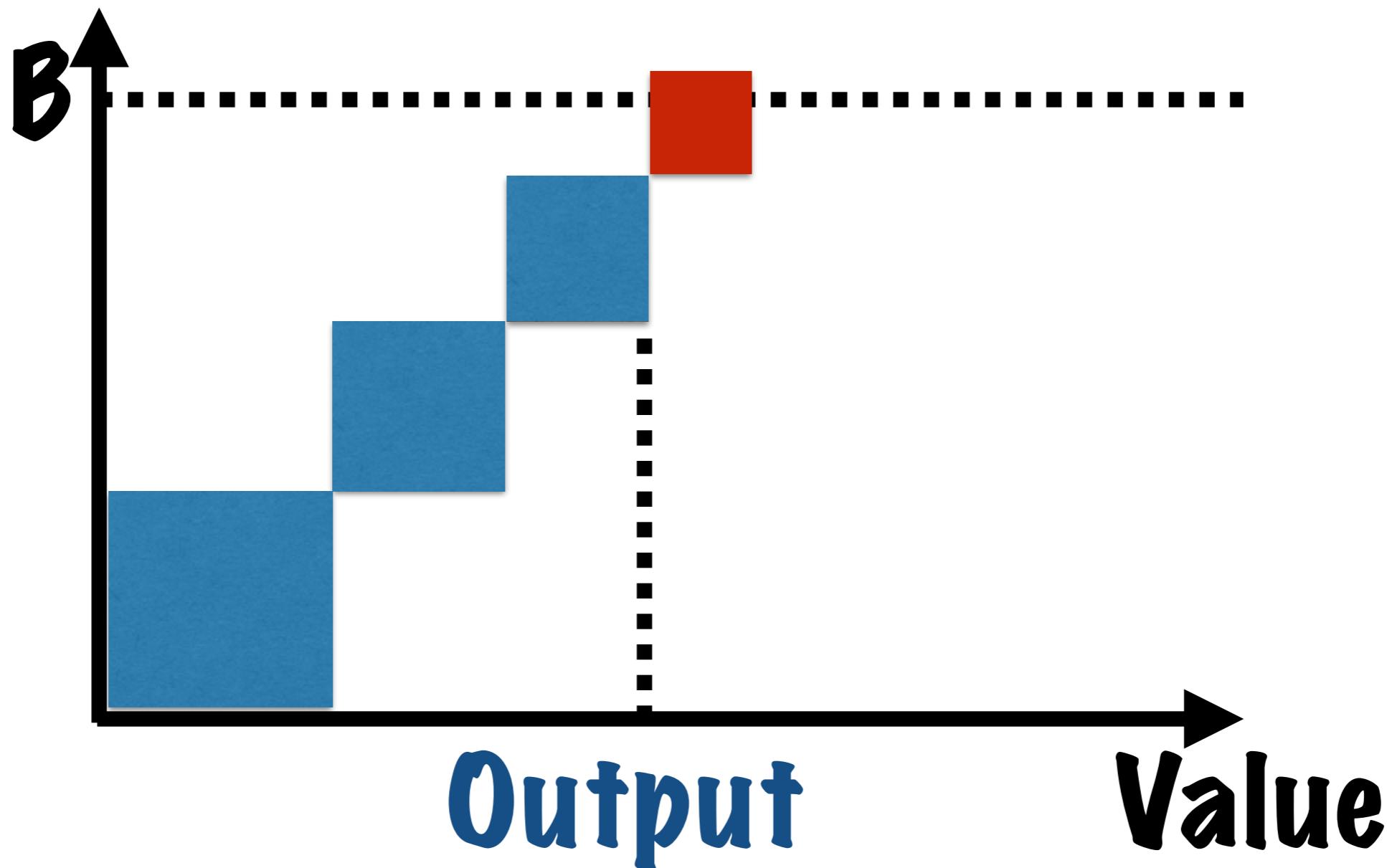
Observe: $\text{OPT} \leq B$



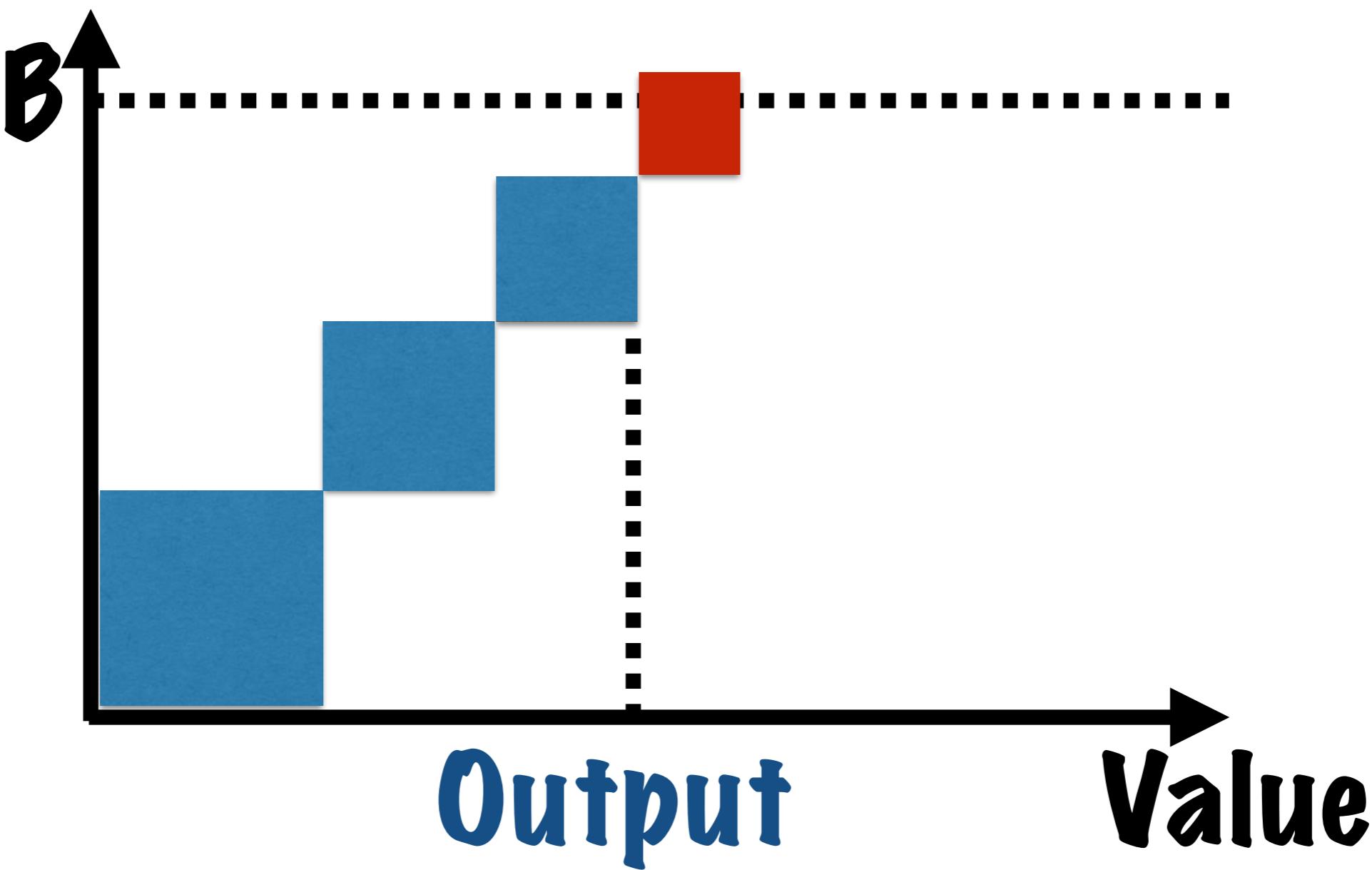
Observe: Output+1 item > B

Can assume:

Output has at least 1 item

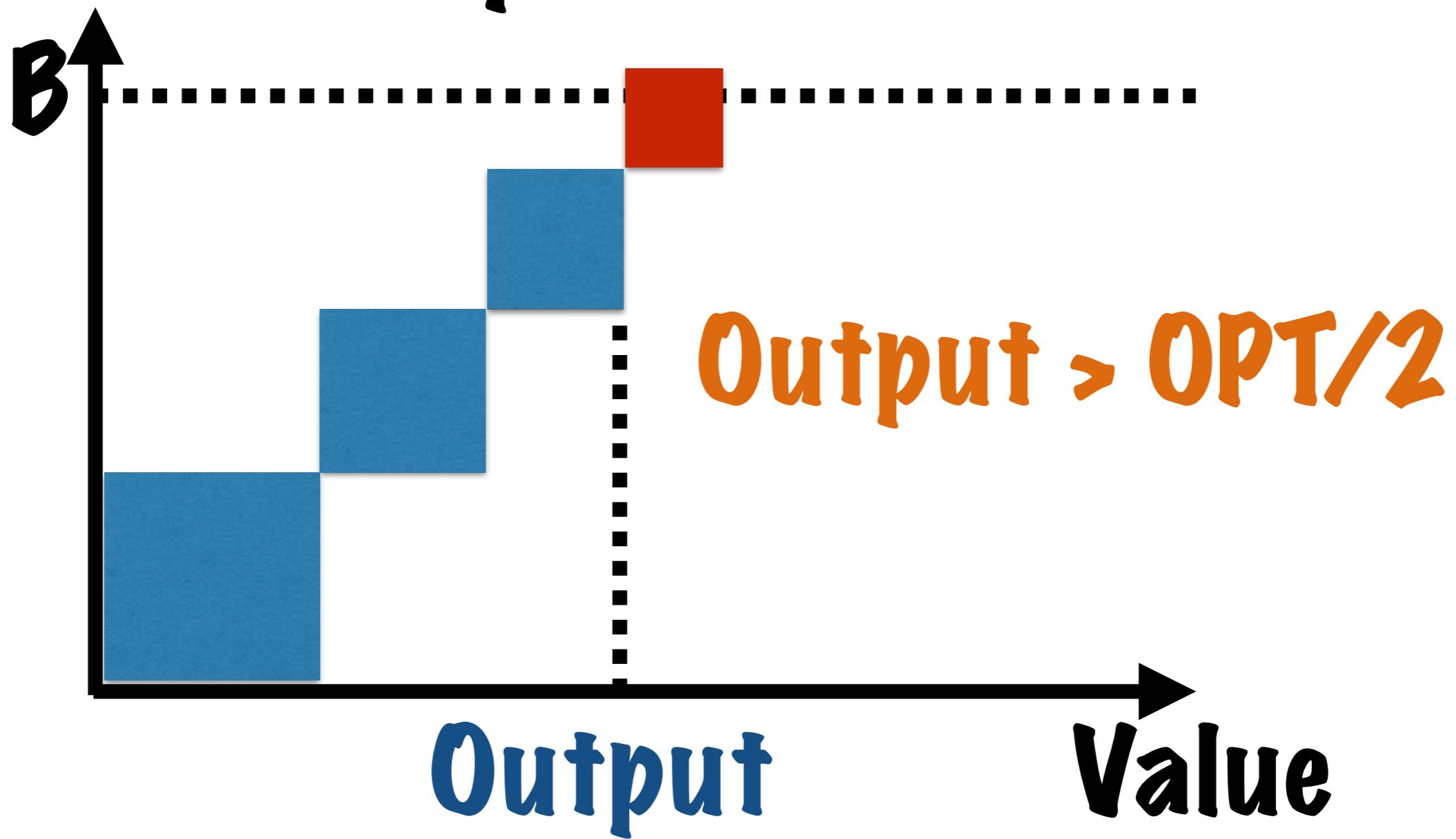


**Observe: first item in output
is better than
item not in output**



Combine: Output + red item > B
First output item > red item

Output > $B/2$



Theorem:
in special case size=value,
greedy is a 2-approximation.

Can we do better?

Knapsack and rounding



Knapsack and rounding



Knapsack special special case

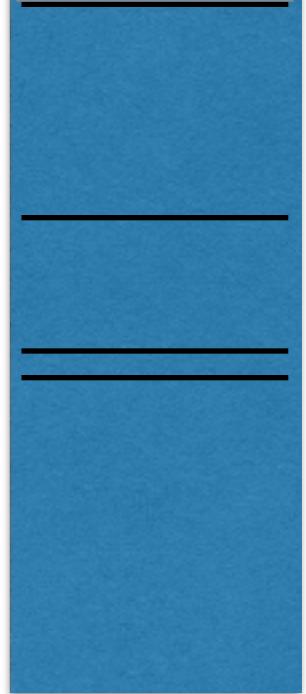
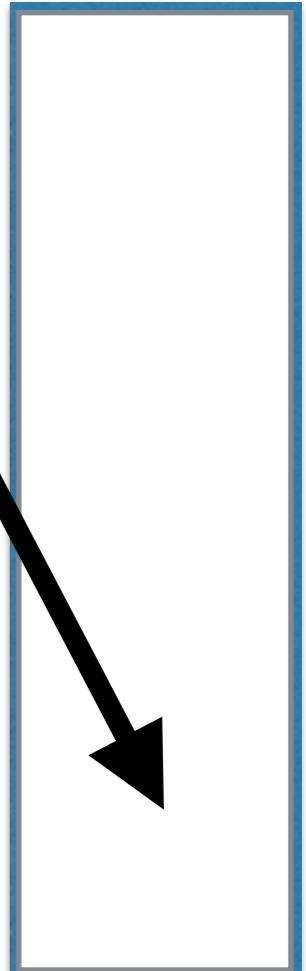
All items have size = value
 $\in \{1, 2, \dots, B\}$
and in addition
B is a “small” integer

Dynamic programming

interface

Given partial solution
for first i items,
what to remember
to complete solution optimally?

add
stuff
here



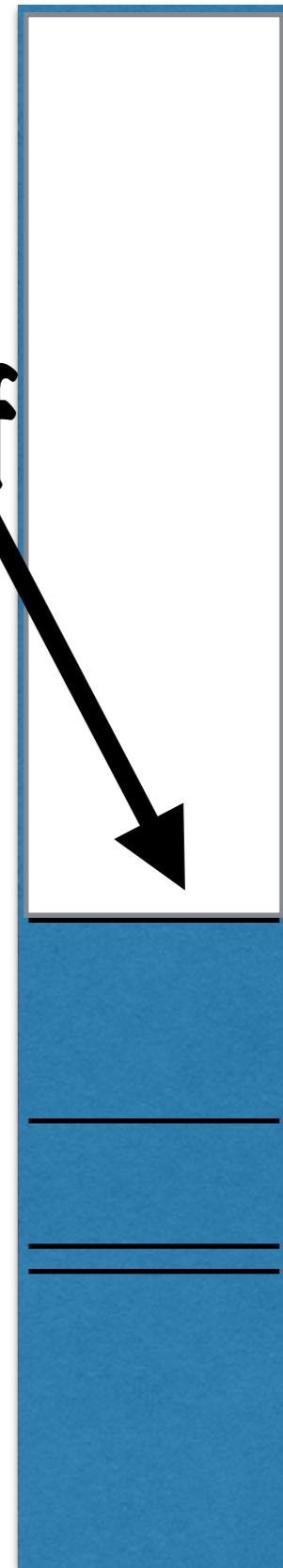
Dynamic programming

Q: What to remember?

A: remember
total value so far

$A[i, v]$ =whether
 v achievable with
subset of first i items

add
stuff
here



Q: v achievable with
subset of first i items iff...

A: ...it depends on
whether subset contains i

If not:

v must be reached
with first $i-1$ items

If yes:

$v - v_i$ must be reached
with first $i-1$ items

**A[i,v]=whether
v achievable with
subset of first i items**

**A[i, v] =
A[i - 1, v] or
((v ≥ v_i) and A[i - 1, v - v_i])**

Algorithm

```
For  $v = 0 \dots B$  :  $A[1, v] \leftarrow \text{false}$ 
 $A[1, v_1] \leftarrow \text{true}, A[1, 0] \leftarrow \text{true}$ 
For  $i = 2 \dots n$ ,
    For  $v = 0 \dots v_i - 1$  :  $A[i, v] \leftarrow A[i - 1, v]$ 
    For  $v = v_i \dots B$  :
         $A[i, v] \leftarrow A[i - 1, v] \text{ or } A[i - 1, v - v_i]$ 
Output  $\max\{v : A[n, v] \text{ is true}\}$ 
```

Knapsack and rounding



Knapsack and rounding



**Less special special case:
values are small integers**

**All values
 $\in \{1, 2, \dots, N\}$**
N: "small" integer

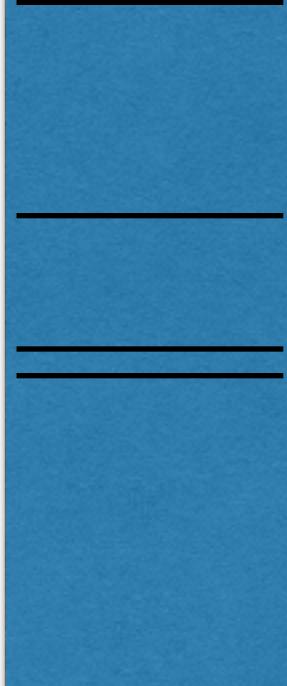
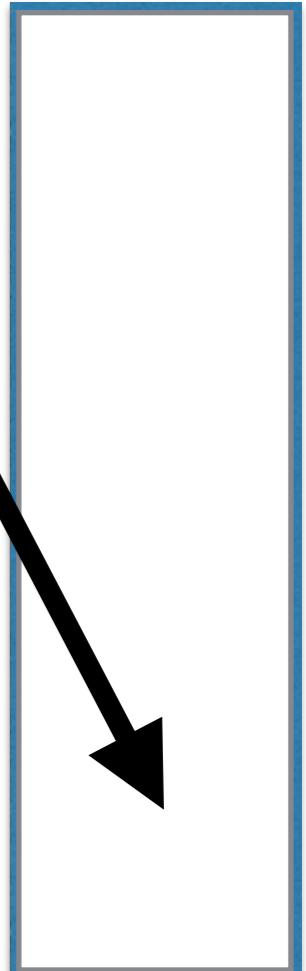
Extend previous ideas

Dynamic programming

interface

Given partial solution
for first i items,
what to remember
to complete solution optimally?

add
stuff
here



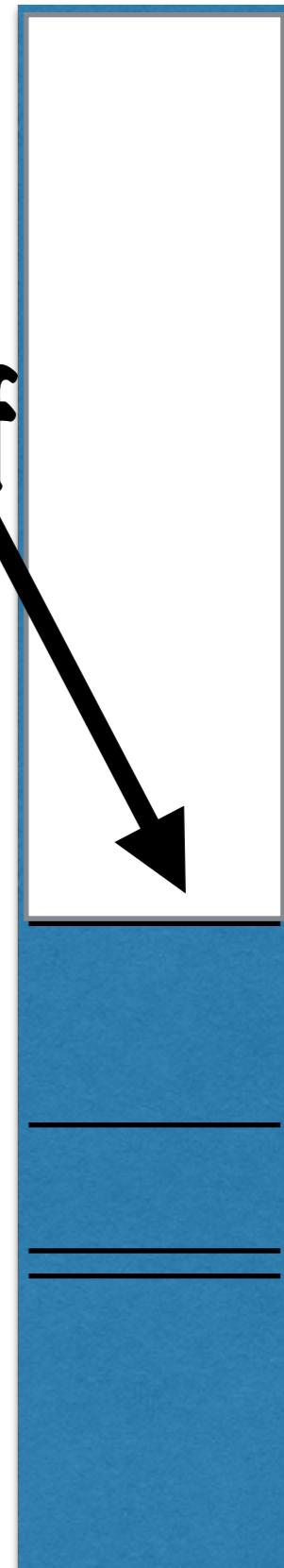
Dynamic programming

Q: What to remember?

$A[i, v]$ =whether
~~v achievable with~~
subset of first i items

$A[i, v]$ =must
remember size

add
stuff
here

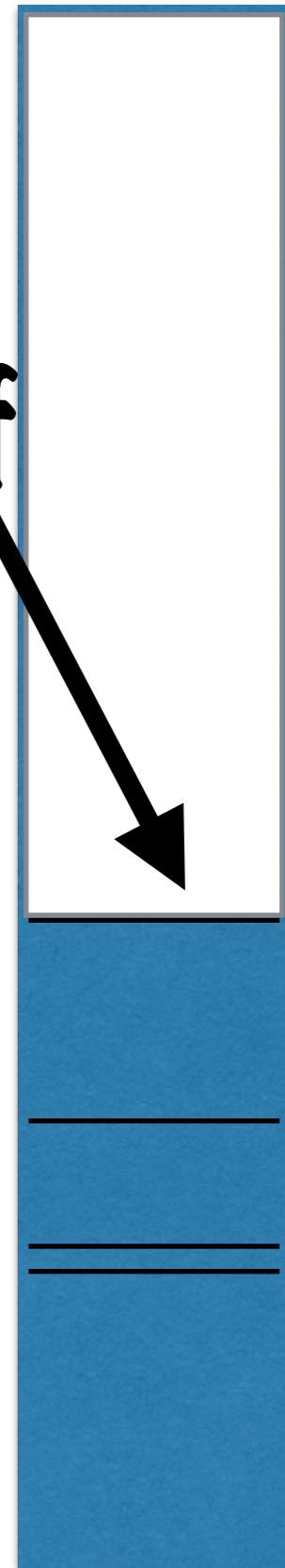


Dynamic programming

Q: What to remember?

add
stuff
here

$A[i, v]$ =min size achievable
for subset of first i items
of value v



Q: v, s achievable with
subset of first i items iff...

A: ...it depends on
whether subset contains i

If not:

v, s reached
with first $i-1$ items

If yes:

$v - v_i, s - s_i$ reached
with first $i-1$ items

Dynamic program

$A[i, v]$ =min size achievable
for subset of first i items
of value v

If $v \geq v_i$
then $A[i, v] =$
 $\min(A[i - 1, v],$
 $A[i - 1, v - v_i] + s_i)$
else ...

For $v = 0 \dots nN$: $A[1, v] \leftarrow B + 1$

$A[1, v_1] \leftarrow s_1$, $A[1, 0] \leftarrow 0$

For $i = 2 \dots n$,

For $v = 0 \dots v_i - 1$: $A[i, v] \leftarrow A[i - 1, v]$

For $v = v_i, v_i + 1, \dots, nN$:

$A[i, v] \leftarrow \min(A[i - 1, v], A[i - 1, v - v_i] + s_i)$

Output $\max\{v : A[n, v] \leq B\}$

Runtime: $O(n^2N)$

Q: What's the main idea?

For $v = 0 \dots nN$: $A[1, v] \leftarrow B + 1$

$A[1, v_1] \leftarrow s_1, A[1, 0] \leftarrow 0$

For $i = 2 \dots n$,

For $v = 0 \dots v_i - 1$: $A[i, v] \leftarrow A[i - 1, v]$

For $v = v_i, v_i + 1, \dots, nN$:

$A[i, v] \leftarrow \min(A[i - 1, v], A[i - 1, v - v_i] + s_i)$ **Stop and admire!**

Output $\max\{v : A[n, v] \leq B\}$



A: The definition of $A[i, v]$

dynamic program key step =
DP table definition

Knapsack and rounding

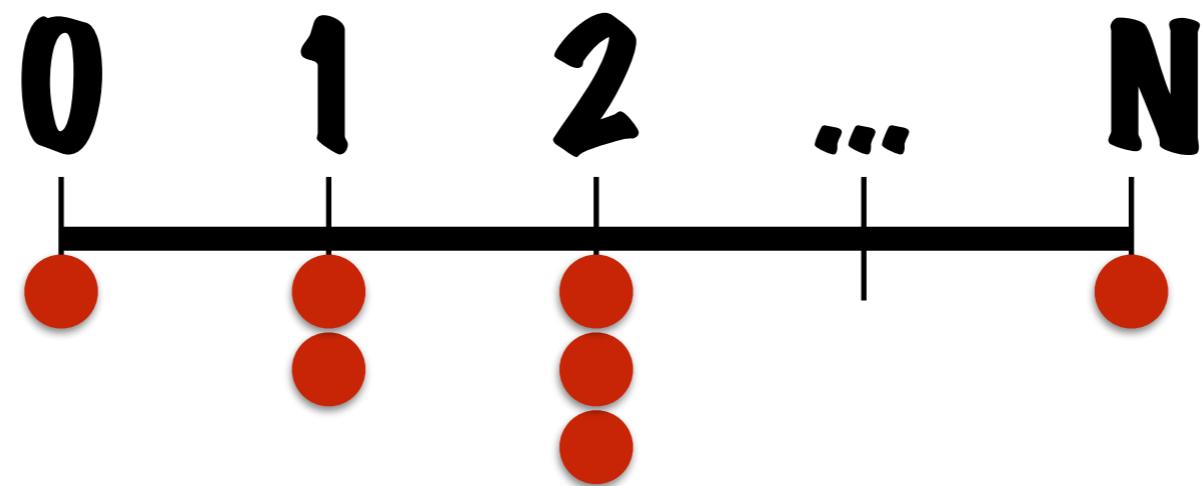


Knapsack and rounding



General case: algorithm

We already have an algorithm
when values
are small integers



Idea:
modify input
so that values are
small integers

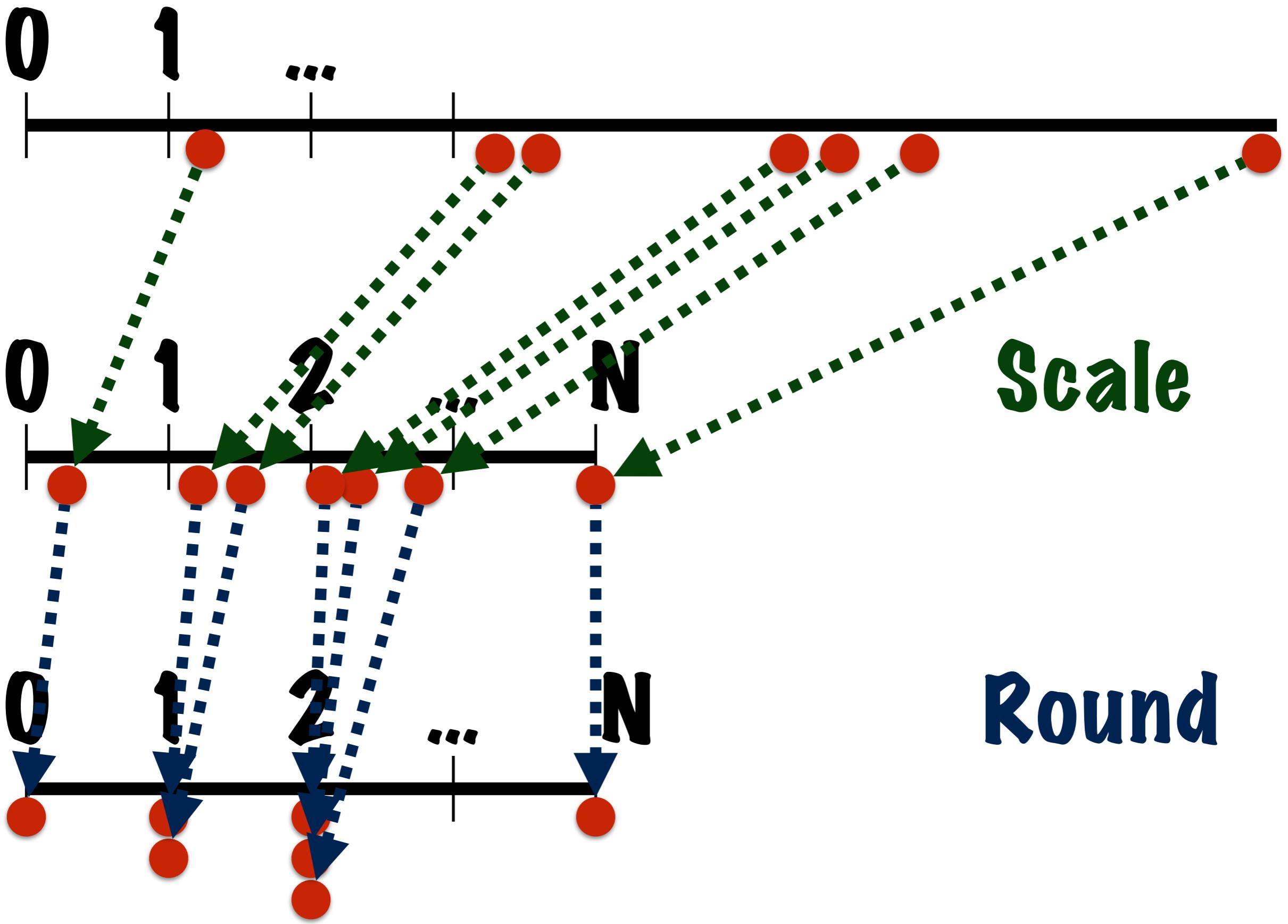
**Q : How to modify input
so that values are
small integers?**

A : Scale and round!

Discard items that don't fit

1. **Multiply each value by something so that values are small**
2. **Round values to integers**

Dynamic program for the scaled and rounded problem



**Effect of scaling:
Max scaled value = N**

Multiply by $\alpha = \frac{N}{\max v_i}$

How do we pick N?

- small enough that the runtime is polynomial
- large enough that the resulting set of items has value close to the optimum

Algorithm Scale

1. Discard items not fitting
2. Let $N=100n$
3. Let $v'_i \leftarrow \lfloor v_i \times \frac{N}{\max_j v_j} \rfloor$
4. Apply DP to $B, (s_i, v'_i)_i$
5. Output corresponding items

Runtime

```
For v = 0 . . . nN : A[1, v] ← B + 1  
A[1, v1] ← s1, A[1, 0] ← 0  
For i = 2 . . . n,  
    For v = 0 . . . vi − 1 : A[i, v] ← A[i − 1, v]  
    For v = vi, vi + 1, . . . , nN :  
        A[i, v] ← min(A[i − 1, v], A[i − 1, v − vi] + si)  
Output max{v : A[n, v] ≤ B}
```

$$N = 100 \times n \implies n^2 N = 100 \times n^3$$

Knapsack and rounding



Knapsack and rounding



Analysis

- S : output items
- DP: S optimal for scaled
rounded input
- S^* : optimal items
- Scaling: S^* optimal for scaled
unrounded input

Relate S for original and modified input value

output value

$$\text{Value}(S) = \sum_S v_i$$

$$= \frac{1}{\alpha} \sum_S (\alpha v_i)$$

$$\geq \frac{1}{\alpha} \sum_S v'_i$$

value for scaled&rounded input

Relate S to S^* on modified input

$$\sum_S v'_i \geq \sum_{S^*} v'_i$$



S optimal for scaled&rounded

Relate S^* for original and modified input value

$$v'_i > \alpha v_i - 1$$



effect of rounding

Sum:

$$\sum_{S^*} v'_i > \alpha \sum_{S^*} v_i - n$$

Combine and substitute:

$$\text{Value}(S) \geq \frac{1}{\alpha} \sum_S v'_i$$

$$\geq \frac{1}{\alpha} \sum_{S^*} v'_i$$

$$\geq \frac{1}{\alpha} [\alpha \sum_{S^*} v_i - n]$$

$$= \text{OPT} - \frac{n}{\alpha}$$

$$= \text{OPT} - \frac{n \times \max v_i}{N}$$

Lower bound OPT

Discarded items that don't fit:

$$OPT \geq \max v_i$$

Wrapping up

$$\text{Value}(S) \geq \text{OPT} - \frac{n}{N} \text{OPT}$$

$$N = 100 \times n$$

$$\text{Value}(S) \geq .99 \times \text{OPT}$$

**Theorem: Solution to knapsack
with value at least .99 OPT
and runtime $O(\text{poly}(n))$**

Knapsack and rounding



Knapsack and rounding



Approximation schemes

The general algorithm

1. Let $N=1000 n$
2. Let $v'_i \leftarrow \lfloor v_i \times \frac{N}{\max_j v_j} \rfloor$
3. Apply DP to $B, (s_i, v'_i)_i$
4. Output corresponding items

Theorem: this gives a solution
to knapsack
with value at least .999 OPT
and with runtime $O(\text{poly}(n))$

**Approximation scheme : family of
algorithms:
One for each $\epsilon > 0$
runtime polynomial in input
output value is near-optimal:**

$$|\text{Value}(\text{Output}) - \text{OPT}| \leq \epsilon \times \text{OPT}$$

**Theorem: knapsack
has an approximation scheme**

How ϵ comes in

Q: The smaller ϵ , the closer to OPT. Why not let it go to 0 and find the exact OPT in $O(\text{poly}(n))$?

A: Runtime increases as
 ϵ goes to zero.

Runtime has N with $N=n/\epsilon$ so
it goes to infinity.

Method:

- 1. Simplify the input**
- 2. Design algorithm for “simple” inputs**

Knapsack and rounding



Question 1

We could choose $p_1 = 0$, $p_2 = 1$, and $p_3 = 0.25$.

- The total value is $\sum p_i s_i = 1 \times 2 + 0.25 \times 4.4 = 3.1$.
- The total size is $\sum p_i v_i = 1 \times 0.5 + 0.25 \times 2 = 1 \leq B$.

Question 2

First let us sort the items by decreasing value of v_i/s_i . Then we go over each item i one after another and choose for p_i the maximum fraction that we can take so that it fits in the remaining space in the bag. That is if r is the remaining capacity when considering item i , then we want $p_i s_i \leq r$ and so choose the largest p_i lower than 1 and r/s_i .

This is implemented by the following pseudo-code where r represents the remaining space in the bag.

```

sort  $(v_i, s_i)$  by decreasing  $v_i/s_i$ 
 $r \leftarrow B$ 
 $v \leftarrow 0$ 
for  $i$  in  $\{1..n\}$ 
     $p_i \leftarrow \min(1, r/s_i)$ 
     $r \leftarrow r - p_i s_i$ 
     $v \leftarrow v + p_i v_i$ 
return  $v$ 

```

At the beginning of the k -th step of the for loop, $r = B - \sum_{i=1}^k p_i s_i$. Note that $p_i \leq r/s_i$ enforces that r is still positive after having been updated. So we finally get that $\sum_{i=1}^n p_i s_i \leq B$, the algorithm return a proper solution.

Question 3

Let us consider that the items are sorted by decreasing v_i/s_i . Let p_i be the optimal solution, and p'_i be the solution produced by the algorithm above.

Assume that the algorithm did not produce the optimal solution. Then let k be the smallest index such that $p_k \neq p'_k$.

If $p_k > p'_k$ then $\sum_{i=1}^k p_i s_i > \sum_{i=1}^k p'_i s_i$. Moreover $p_k \leq 1$ so $p'_k < 1$ and so $p_k = r/s_k$ where r is the remaining capacity at the beginning of the k -th iteration of the for loop, hence $r = B - \sum_{i=1}^{k-1} p'_i s_i$. And so we obtain $\sum_{i=1}^k p'_i s_i = B < \sum_{i=1}^k p_i s_i$, p_i is not a correct solution which is a contradiction.

If $p_k < p'_k$ then we can modify p_i by setting p_k to p'_k and decreasing the values of p_i for $i > k$ so that $\sum_i p_i v_i \leq B$. Then as $v_i/s_i < v_k/s_k$ for any $i > k$ the resulting total value will be strictly greater which is in contradiction with the optimality of p_i .

Question 4

The complexity of the sort is $O(n \cdot \log(n))$. The complexity of the for loop is $O(n)$. So the overall complexity is $O(n \cdot \log(n))$.

[Course Home \(/learn/approximation-algorithms-part-1/home/welcome\)](#) ➤ [Week 2 \(/learn/appro...](#)

Review Classmates: Peer Assignment Knapsack

Review by December 23, 11:59 PM PT

Reviews 3 left to complete

Fractional Knapsack



by Johnson Jia

December 22, 2015

like Flag this submission

Consider the following fractional variant of knapsack packing which we call Fractional Knapsack. For every item you can pack an arbitrary fraction of that item, i.e., if you pack a fraction $p_i \in [0, 1]$ of item $i \in \{1, \dots, n\}$ then it takes $p_i \cdot s_i$ space and has a value $p_i \cdot v_i$.

For simplicity, assume that v_i / s_i are different for all i .

1. Let $B = 1$. Given three items with $s_1 = 1, v_1 = 2.1, s_2 = 0.5, v_2 = 2, s_3 = 2$, and $v_3 = 4.4$. How should p_1, p_2 , and p_3 be chosen to reach a total value of 3.1?
2. Design a combinatorial algorithm (no LP allowed) which achieves the optimal value.
3. Argue the correctness.
4. Give the time complexity of your algorithm.

[Fractional Knapsack](https://s3.amazonaws.com/coursera-uploads/peer-review/bnQLDSciEeWbiBJCM9ziNQ/e9830e9431dd2ac79be240dd9af1c6ff/FracKS.pdf) (<https://s3.amazonaws.com/coursera-uploads/peer-review/bnQLDSciEeWbiBJCM9ziNQ/e9830e9431dd2ac79be240dd9af1c6ff/FracKS.pdf>)

Typeset using Latex2e

Help Center

One possible solution:

1. Set $p_1 = 0, p_2 = 1$, and $p_3 = 1/4$.

2. Sort items in decreasing order of v_i / s_i (or equivalently in increasing order of s_i / v_i).

Pack items in this order greedily until the bin is full or no items are left.



3. Correctness: Compare the greedy solution (ALG) with the optimal solution (OPT). We show by contradiction that the solutions are identical.

Suppose the solutions were not identical.

Let i be the first item where the solutions differ, i.e., $p_i(OPT) \neq p_i(ALG)$. The greedy algorithm ensures that we have $p_i(OPT) < p_i(ALG)$.

Since we assumed that v_i/s_i are different for all i , this means that the optimal solution can be improved (by taking more of item i , i.e., by setting $p_i(OPT) = p_i(ALG)$). Which is a contradiction, since OPT is the optimal solution. Hence, the greedy algorithm is optimal.

4. The time complexity is dominated by the time it takes to sort n item. Hence the time complexity is $O(n \log n)$.

1. The values for the fractions are set to $p_1 = 0$, $p_2 = 1$, and $p_3 = 1/4$.

- 5 pts
Yes
- 0 pts
No

2. The items packed are chosen are selected by decreasing value of v_i/s_i (or equivalently in increasing value of s_i/v_i)

- 5 pts
Yes
- 0 pts
No

3. It is argued that the optimal solution must have the same choice of p_i for all items i

- 7 pts
Yes
- 0 pts
No

4. The time complexity is computed correctly.

- 3 pts
Yes

0 pts
No

[Submit Review](#)

Comments

Visible to classmates



share your thoughts...

◀ (/learn/approximation-algorithms-part-1/supplement/x3O2m/practise-exercises)
(https://accounts.coursera.org/zendesk/courserahelp?return_to=https://learner.coursera.help/hc/1/supplement/x3O2m/practise-exercises)

▶ (/learn/approximation-algorithms-part-1/supplement/reals/all-slides-together-in-one-file)
([https://www.coursera.org/learn/approximation-algorithms-part-1/peer/4mFQh/peer-assignment-knapsack/review-next](#))

Pracise Exercise on Knapsack

The solutions will be available in 1-3 weeks in one of the following modules.

Question 1

Consider the following greedy algorithm for knapsack packing.

- (a) Sort items in non-increasing order of v_i/s_i
- (b) Greedily add items until we hit an item a_L that is too big ($\sum_{k=1}^L s_k > B$)
- (c) Pick the better of $\{a_1, a_2, \dots, a_{L-1}\}$ and $\{a_L\}$.

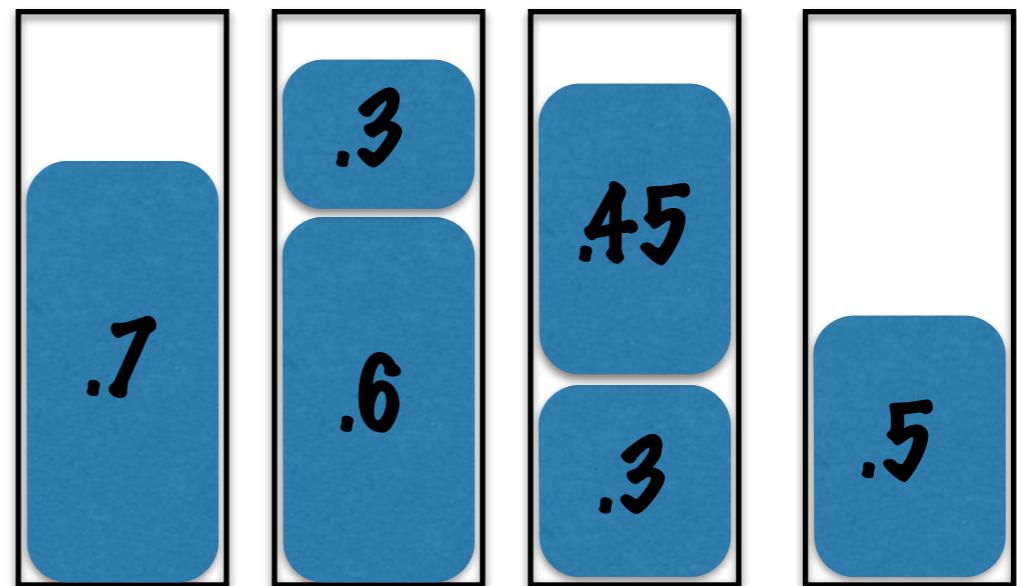
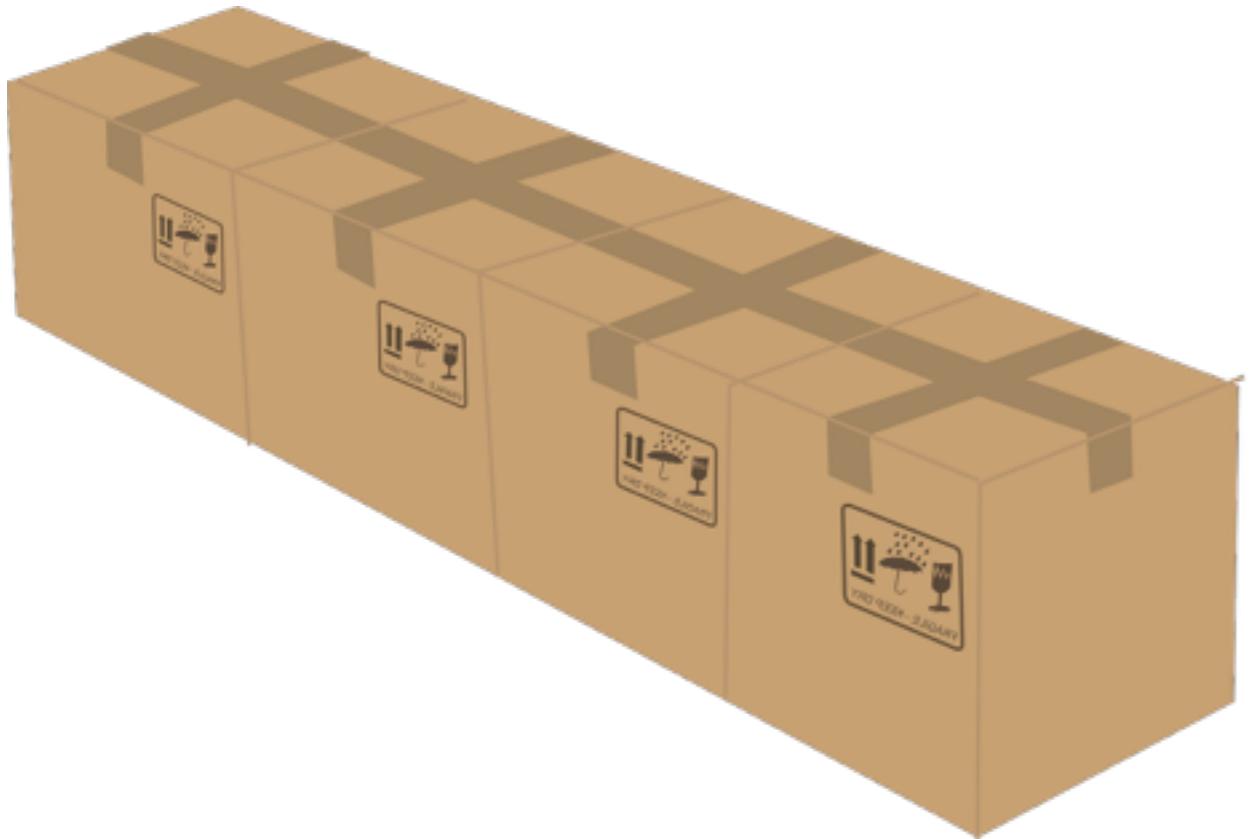
Your task is the following.

- (a) Show that the value of the solution found by the greedy algorithm is at least half of the (unknown) optimal value as the number of items n tends to infinity. (2-Approximation)
- (b) There exists an instance (set of items) such that the optimal value reached by the greedy algorithm is half of the value reached by the optimal algorithm as the size of knapsack goes to infinity. (tightness)

Question 2

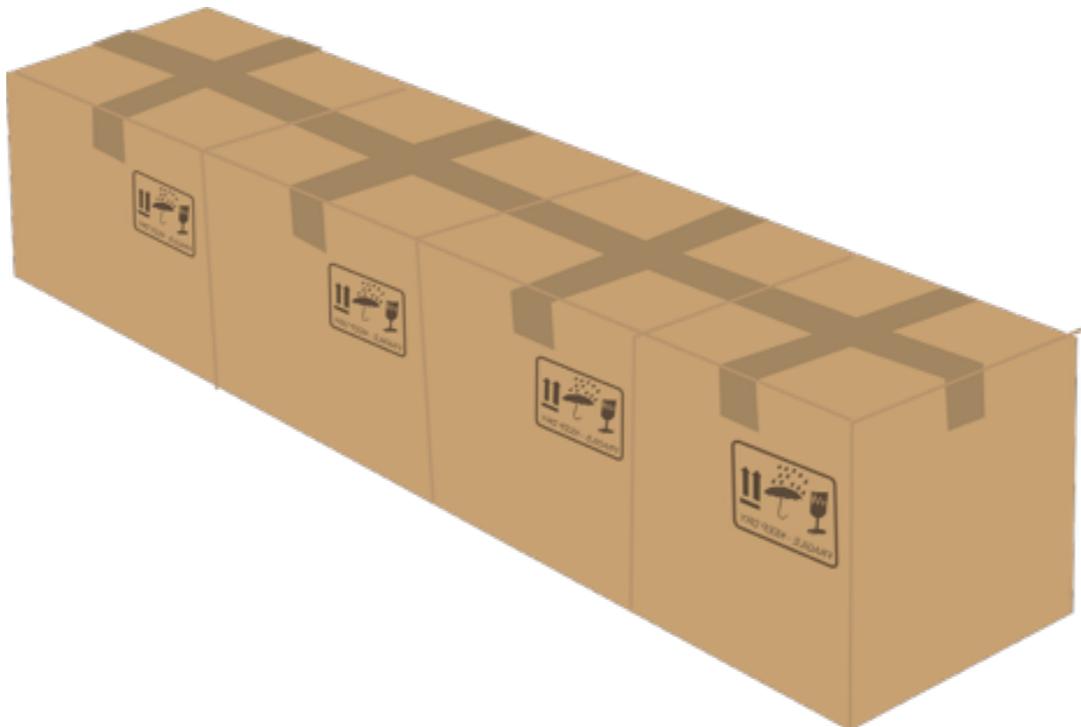
You are doing the Tour de France and you are given a map with all the n places where you can refill your water bottles. Your bottles fit 2 liters of water with which are enough for m kilometres. Your goal is to stop as few times as possible to refill your bottles, since you care about a good time for the Tour de France. Design an efficient algorithm using dynamic programming, prove its correctness and analysis its run time. Give the one-dimensional array A . And the time complexity of filling your array.

Bin packing, linear programming and rounding



The bin packing problem

Pack your items
using
as few bins as possible



Given n items
item i has size $s_i < 1$
pack items into the fewest
unit capacity bins

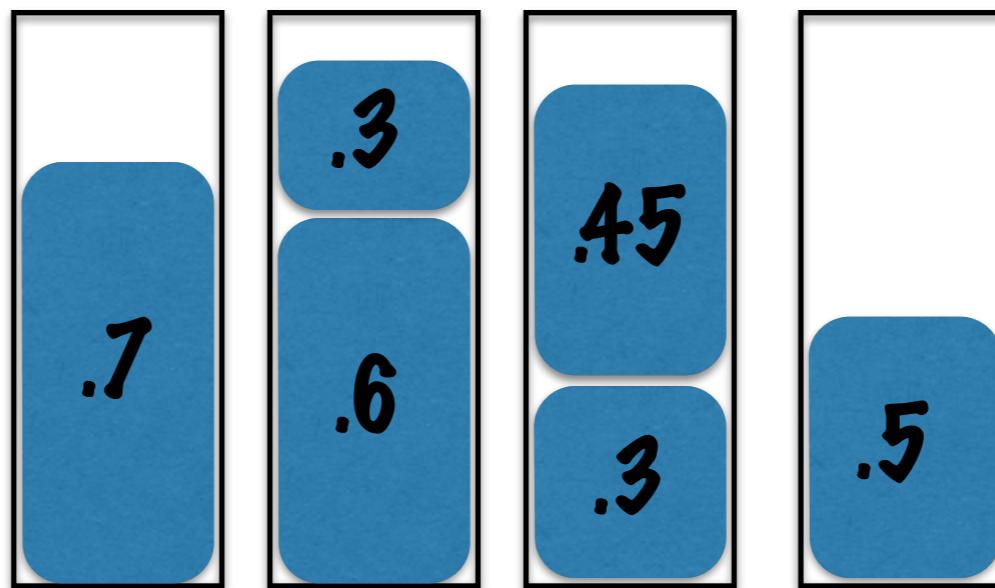


The Next Fit algorithm

One bin at a time:
If next item does not fit,
close the bin and
open a new bin

"Next Fit" algorithm

$s1=.7$
 $s2=.6$
 $s3=.4$
 $s4=.3$
 $s5=.45$
 $s6=.5$



can this instance
be packed better?

How good is Next Fit?

Capacity 100. Items:

| | | | | | | | | | | | | | |
|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 91 | 50 | 21 | 90 | 39 | 43 | 54 | 95 | 25 | 14 | 25 | 61 | 35 | |
| 28 | 97 | 16 | 18 | 42 | 99 | 95 | 12 | 7 | 21 | 23 | 42 | 50 | 77 |
| 100 | 79 | 36 | 31 | 71 | 35 | 1 | 63 | 6 | 13 | 92 | 58 | 73 | 72 |
| 32 | 37 | 62 | 54 | 18 | 25 | 9 | 52 | 93 | | | | | |

Next Fit.



used 31 bins

| | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 91 | 50 | 21 | 90 | 39 | 43 | 54 | 95 | 25 | 14 | 25 | 61 | 35 | 28 | 97 | 16 | 18 | 42 | 99 | 95 | 12 | 7 | 21 | 23 | |
| 42 | 50 | 77 | 100 | 79 | 36 | 31 | 71 | 35 | 1 | 63 | 6 | 13 | 92 | 58 | 73 | 72 | 32 | 37 | 62 | 54 | 18 | 25 | 9 | 52 |
| 93 | | | | | | | | | | | | | | | | | | | | | | | | |

bin 7: $(25+14+25)$
 next item: 61, but
 $(25+14+25)+61 > 100$
 so, close bin 7, open bin 8,
 put item 61 in bin 8.



| | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 91 | 50 | 21 | 90 | 39 | 43 | 54 | 95 | 25 | 14 | 25 | 61 | 35 | 28 | 97 | 16 | 18 | 42 | 99 | 95 | 12 | 7 | 21 | 23 | |
| 42 | 50 | 77 | 100 | 79 | 36 | 31 | 71 | 35 | 1 | 63 | 6 | 13 | 92 | 58 | 73 | 72 | 32 | 37 | 62 | 54 | 18 | 25 | 9 | 52 |
| 93 | | | | | | | | | | | | | | | | | | | | | | | | |

In general:

(items in bin $2i-1$) + next item > 100

(items in bins $2i-1$ or $2i$) > 100

31 bins: total item sizes > $15 * 100$



| | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 91 | 50 | 21 | 90 | 39 | 43 | 54 | 95 | 25 | 14 | 25 | 61 | 35 | 28 | 97 | 16 | 18 | 42 | 99 | 95 | 12 | 7 | 21 | 23 | |
| 42 | 50 | 77 | 100 | 79 | 36 | 31 | 71 | 35 | 1 | 63 | 6 | 13 | 92 | 58 | 73 | 72 | 32 | 37 | 62 | 54 | 18 | 25 | 9 | 52 |
| 93 | | | | | | | | | | | | | | | | | | | | | | | | |

In general:
k bins by Next Fit
Total item sizes > $(k-1)/2 * 100$



| | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 91 | 50 | 21 | 90 | 39 | 43 | 54 | 95 | 25 | 14 | 25 | 61 | 35 | 28 | 97 | 16 | 18 | 42 | 99 | 95 | 12 | 7 | 21 | 23 | |
| 42 | 50 | 77 | 100 | 79 | 36 | 31 | 71 | 35 | 1 | 63 | 6 | 13 | 92 | 58 | 73 | 72 | 32 | 37 | 62 | 54 | 18 | 25 | 9 | 52 |
| 93 | | | | | | | | | | | | | | | | | | | | | | | | |

What about OPT?

Total item sizes < OPT * 100



| | | | | | | | | | | | | | | | | | | | | | | | | |
|----|----|----|-----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|----|
| 91 | 50 | 21 | 90 | 39 | 43 | 54 | 95 | 25 | 14 | 25 | 61 | 35 | 28 | 97 | 16 | 18 | 42 | 99 | 95 | 12 | 7 | 21 | 23 | |
| 42 | 50 | 77 | 100 | 79 | 36 | 31 | 71 | 35 | 1 | 63 | 6 | 13 | 92 | 58 | 73 | 72 | 32 | 37 | 62 | 54 | 18 | 25 | 9 | 52 |
| 93 | | | | | | | | | | | | | | | | | | | | | | | | |

Combining:
k bins by Next Fit
 $\text{OPT} * 100 > (k-1)/2 * 100$
 $\#(\text{bins of Next Fit}) < 2 * \text{OPT} + 1$



Asymptotic 2 approximation

Is this tight?

Example with
 $\text{OPT}=501$ bins,
 $\text{Next Fit}=1000$ bins

What about non-asymptotic?

Distinguishing between
 $\text{OPT}=2$ and $\text{OPT}=3$
is NP-hard

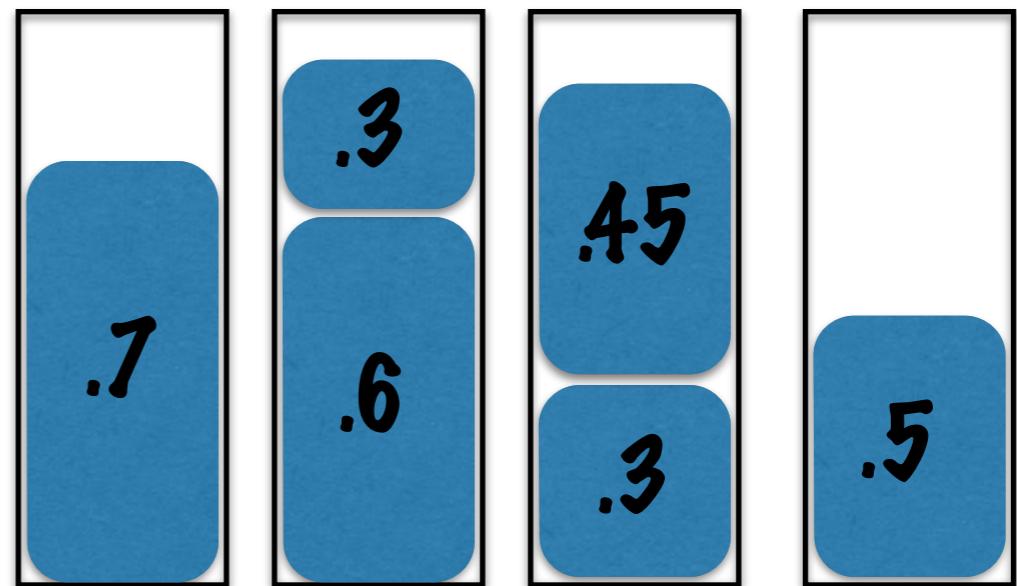
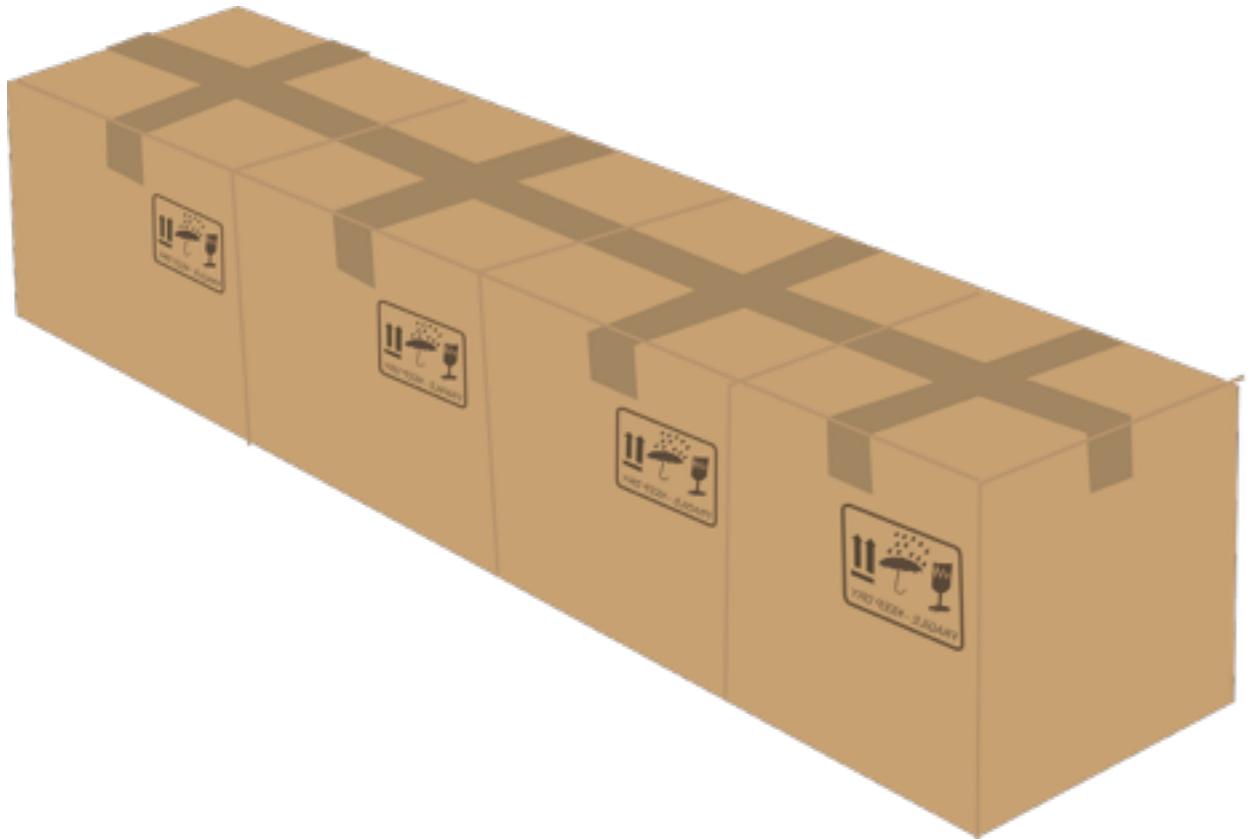
What have we learned?

1. Crude algorithms can give good bounds

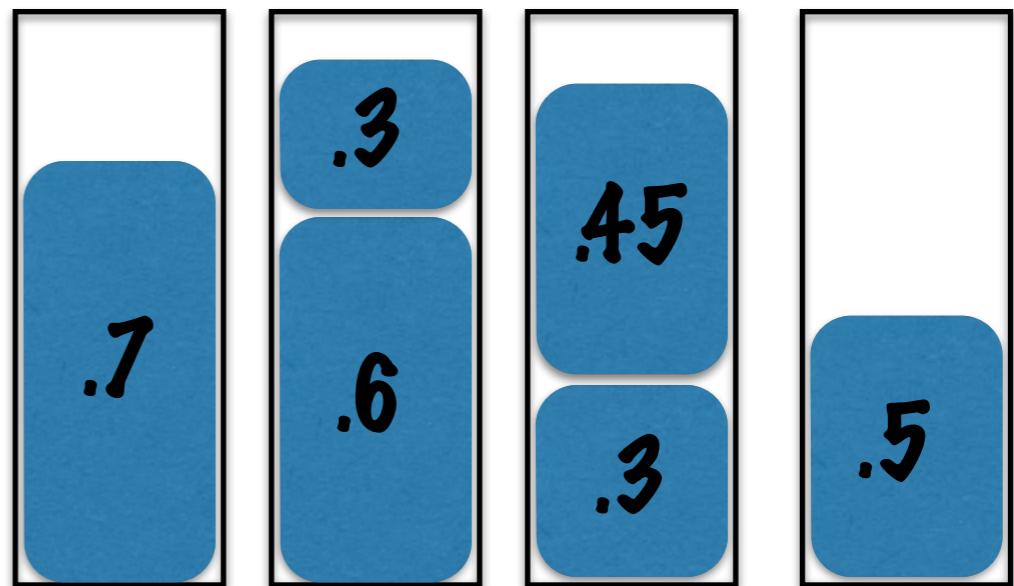
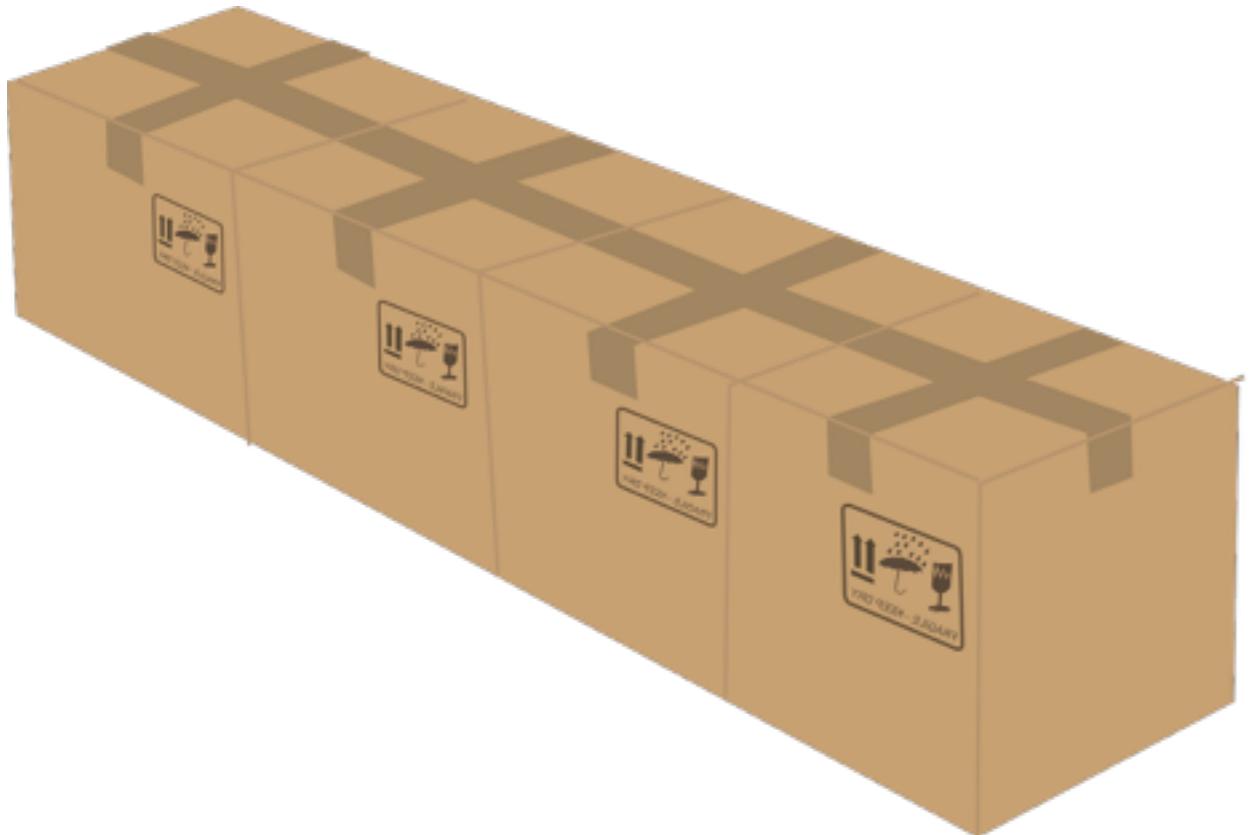
Message: first try the simplest algorithm

2. Analysis: for intuition, first execute it on some concrete examples

Bin packing, linear programming and rounding



Bin packing, linear programming and rounding



Can we do better than
Next Fit?

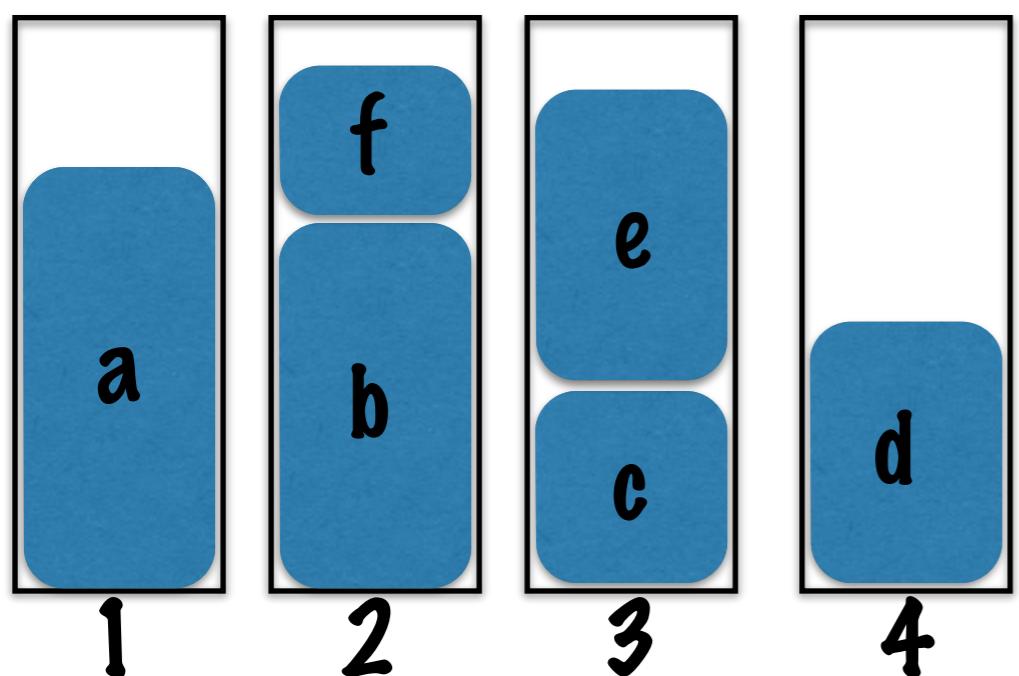
First tool: linear programming
relaxation

An integer program

Given n items and K unit bins,
is there a packing?

Variables: $x_{ij} \in \{0, 1\}$

$x_{ij} = 1$ iff item i is placed in bin j



$$x_{a1} = x_{b2} =$$

$$x_{f2} = x_{c3} =$$

$$x_{e3} = x_{d4} = 1,$$

$$x_{ij} = 0 \text{ otherwise}$$

An integer program

Constraint: every item
must go somewhere

Item b must go into bin 1,2,3 or 4

$$x_{b1} + x_{b2} + x_{b3} + x_{b4} = 1$$

An integer program

Constraint: must not exceed bin capacity

Item sizes in bin j sum to at most 1.

$$x_{aj}s_a + x_{bj}s_b + \dots + x_{fj}s_f \leq 1$$

Integer program

n items, K bins

x_{ij} = whether item i goes into bin j

$$\forall i : \sum_j x_{ij} = 1$$

$$\forall j : \sum_i x_{ij} s_i \leq 1$$

$$\forall i, j : x_{ij} \in \{0, 1\}$$

feasible iff items can be packed into K bins

Linear programming relaxation

n items, K bins

$$\forall i : \sum_j x_{ij} = 1$$

$$\forall j : \sum_i x_{ij} s_i \leq 1$$

$$\forall i, j : 0 \leq x_{ij} \leq 1$$

Algorithm: use the LP relaxation
to pack items (somehow)

How good is the relaxation?

A bad example

n items, K bins

$$\forall i : \sum_j x_{ij} = 1$$

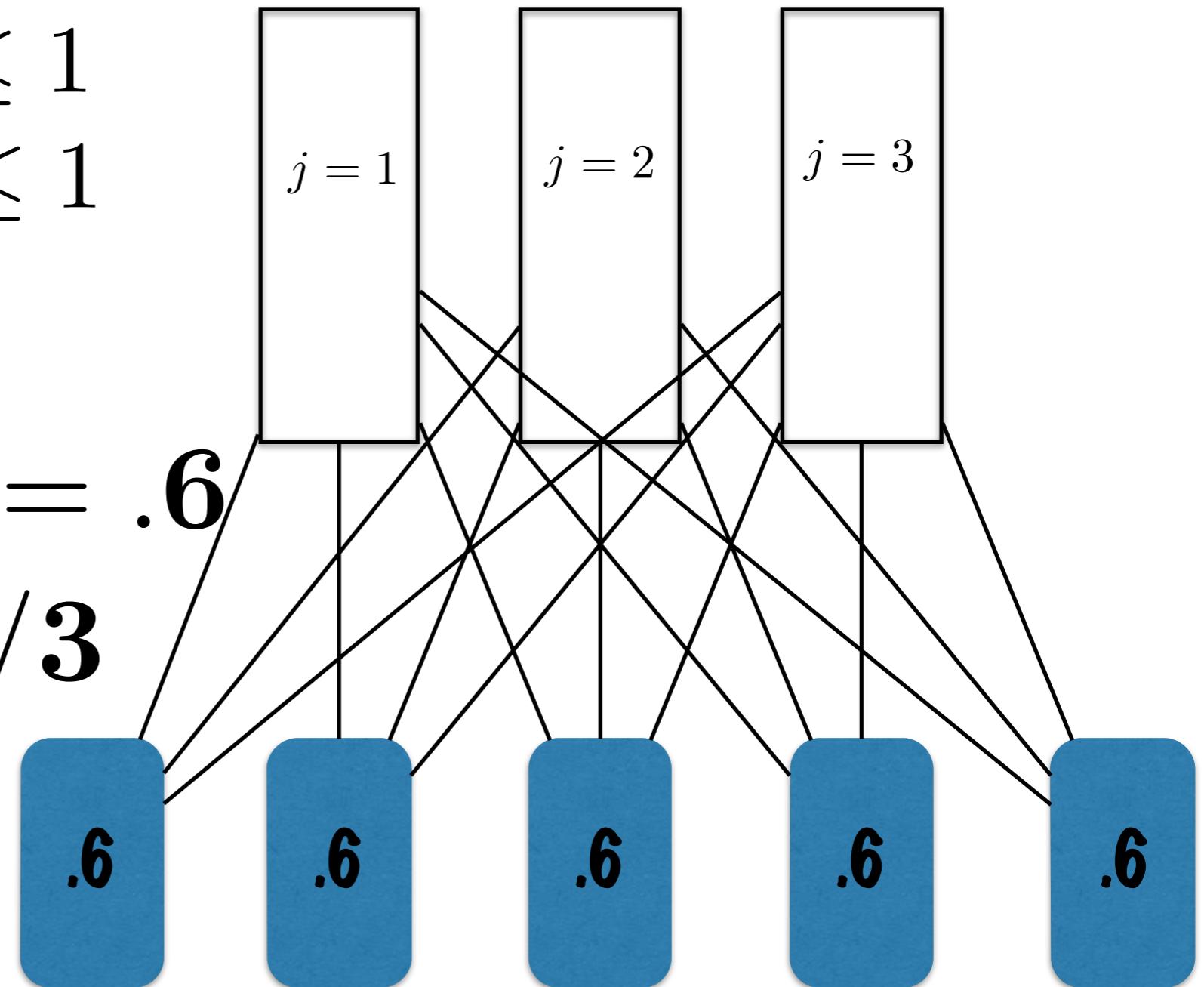
$$\forall j : \sum_i x_{ij} s_i \leq 1$$

$$\forall i, j : 0 \leq x_{ij} \leq 1$$

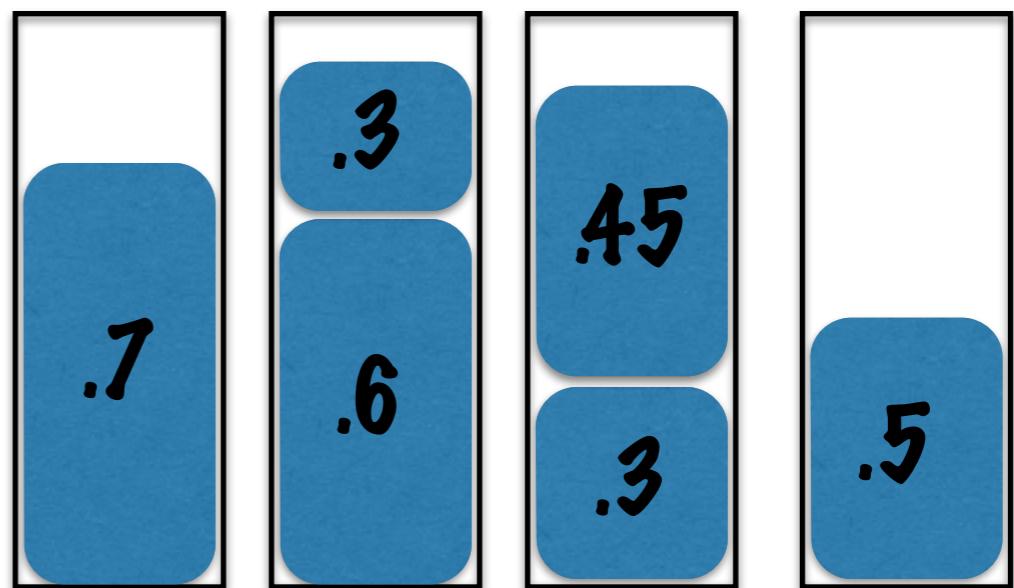
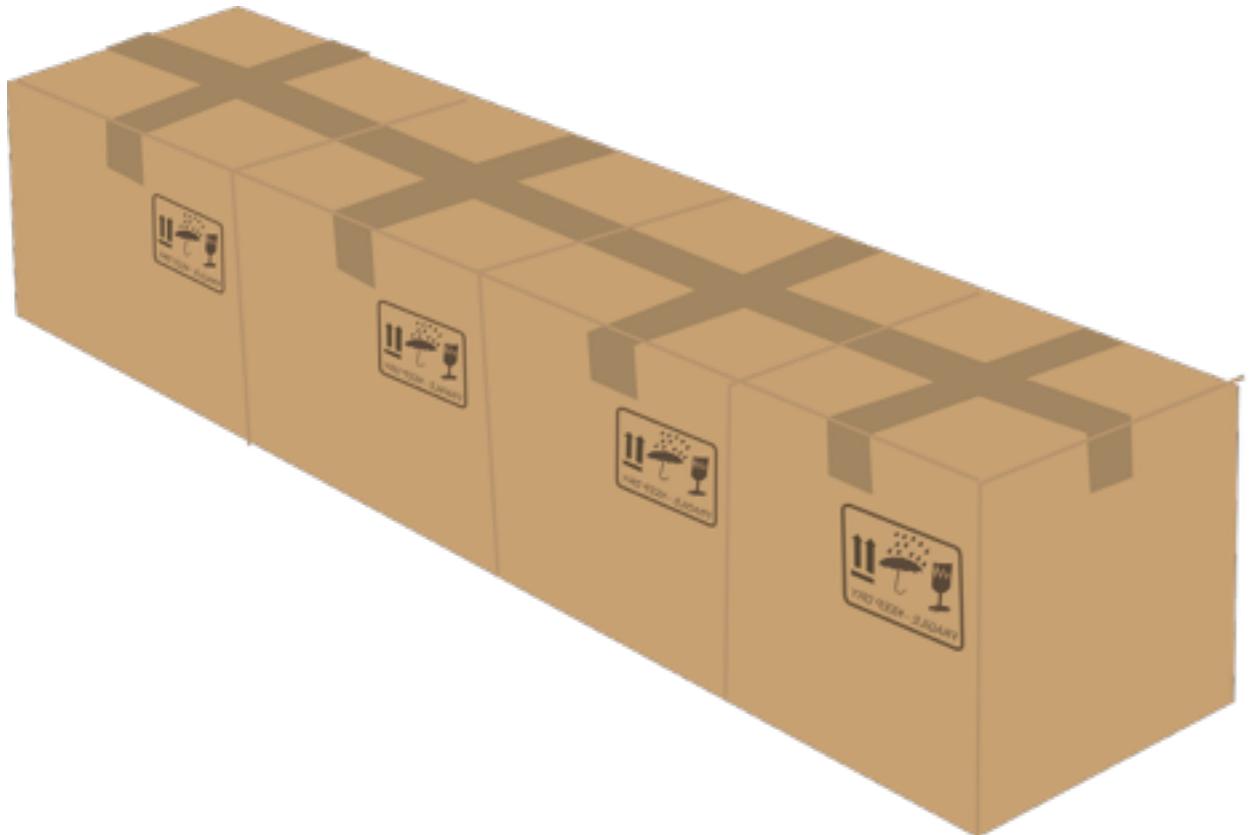
$$s_1 = \dots = s_5 = .6$$

$$\forall i, j \quad x_{ij} = 1/3$$

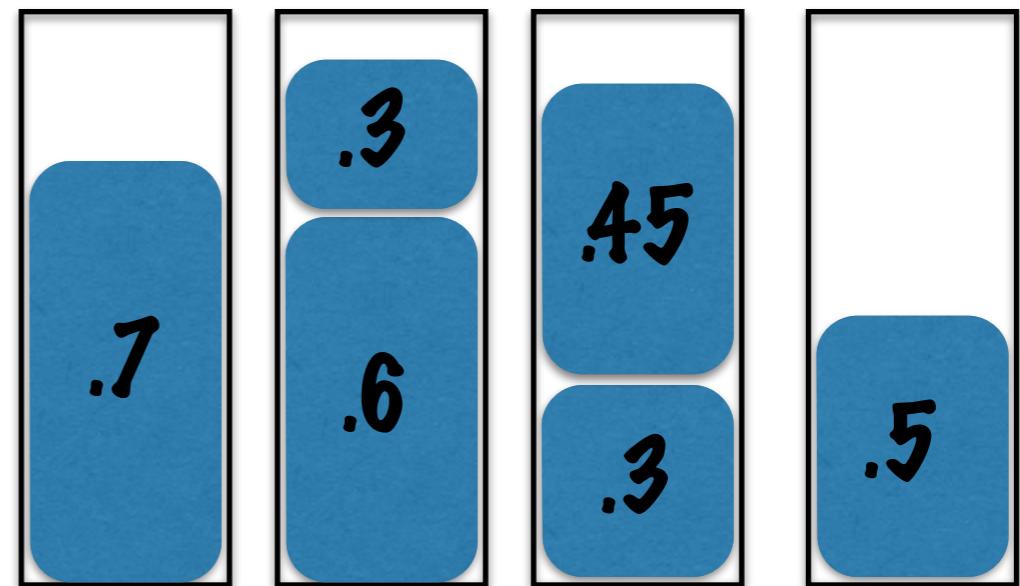
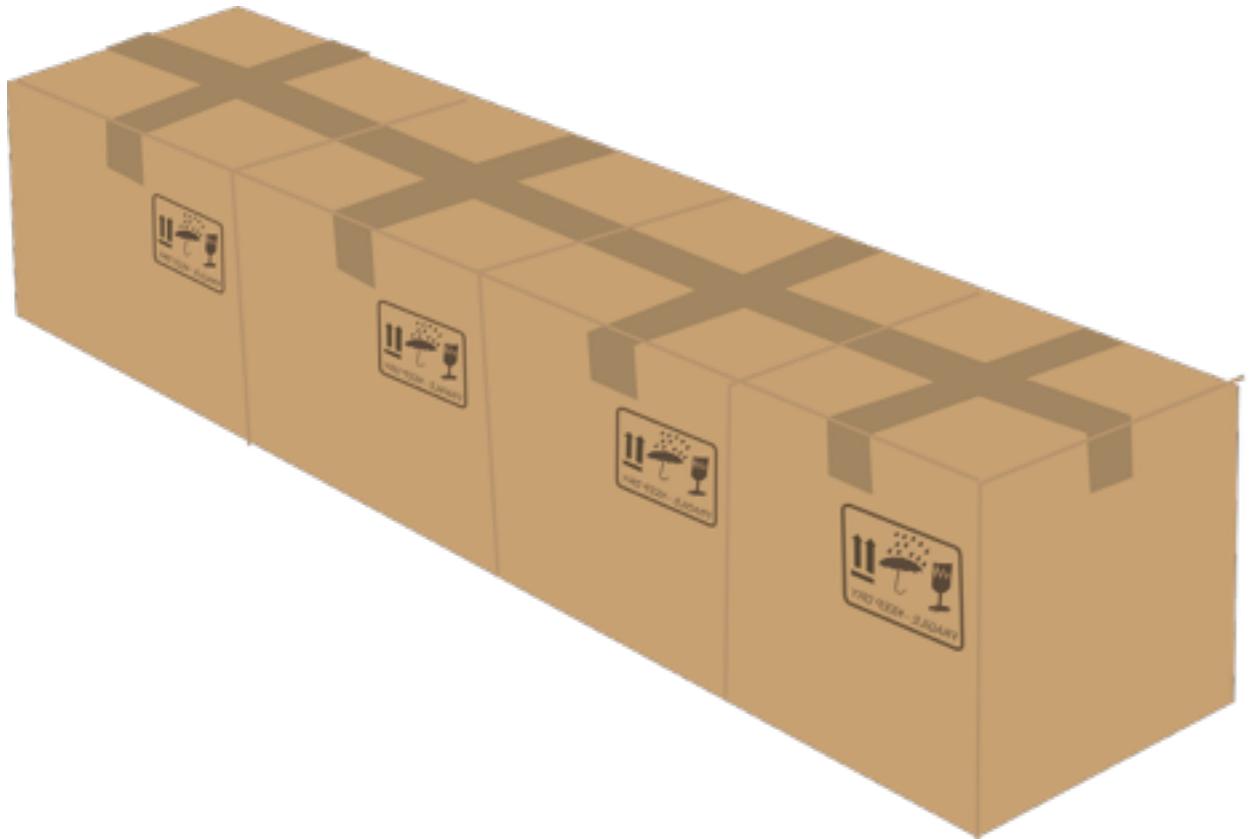
LP: 3 bins
OPT: 5 bins



Bin packing, linear programming and rounding



Bin packing, linear programming and rounding



Remember:

To analyze output vs. OPT,
focus on LP value...

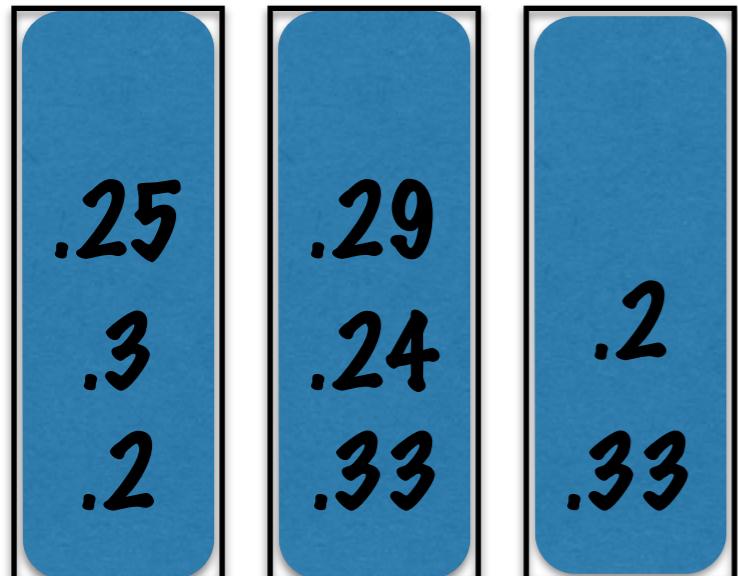


Try next meta-tool:
special cases

What if items are smaller than
1/3*capacity?

.2,.3,.25,.33,.24,.29,.33,.2,...

What does Next Fit do?



$$.2 + .3 + .25 = .75$$

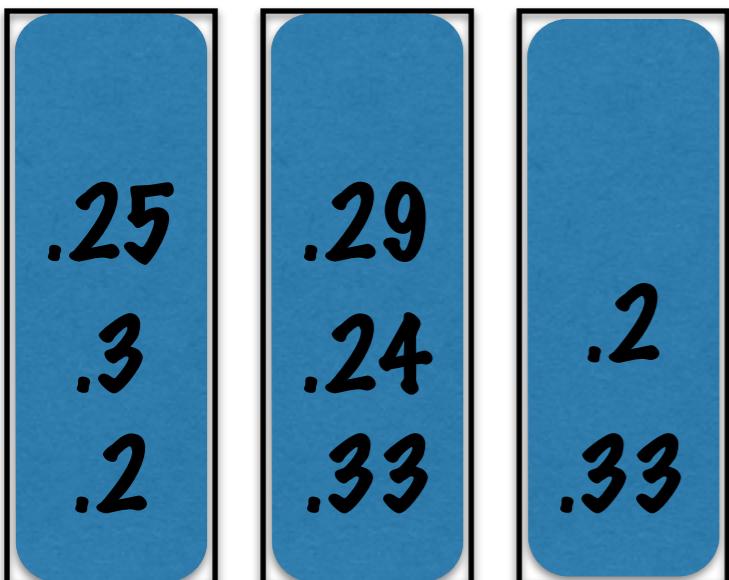
$$.33 + .24 + .29 = .86$$

$$.33 + .2 = .53$$

The next item will fit in bin 3:
 $.53 + (\text{something less than } 1/3) < 1$

Next Fit when items are smaller than
 $1/3 * \text{capacity}$

Bin filled to $< 2/3$: next item fits
so:
only close bin when filled to $> 2/3$



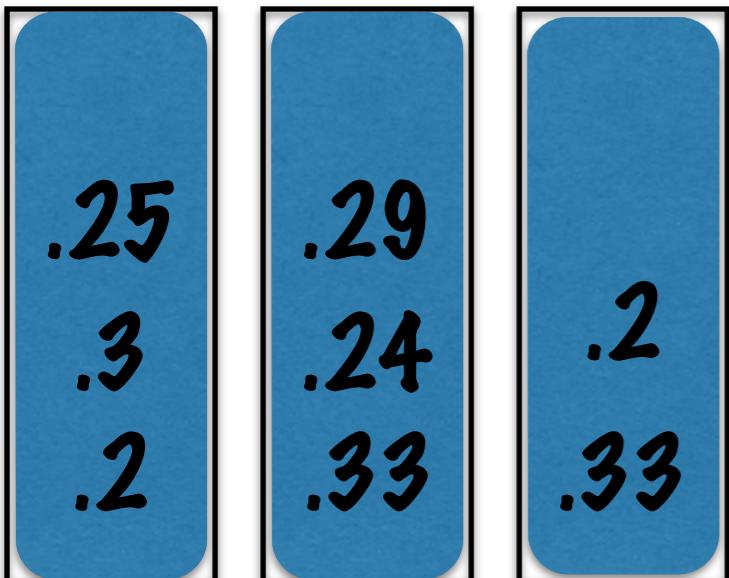
All bins except last
are filled to $> 2/3$

Next Fit when items are smaller than
 $1/3 * \text{capacity}$

All bins except last
are filled to $> 2/3$

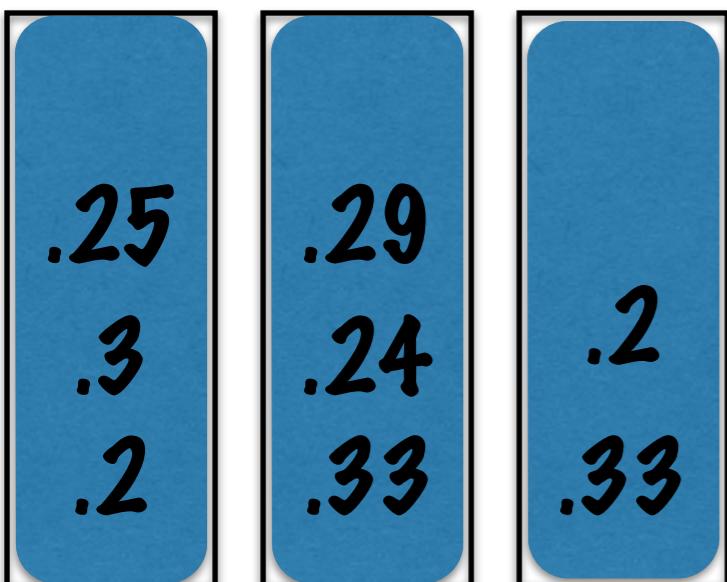
Total size $> 2/3 * (\#\text{bins} - 1)$

But Total size $< \text{OPT}$



Combine:
 $\#\text{bins} < 3/2 * \text{OPT} + 1$

Theorem:
when items are smaller than
 $1/3 * \text{capacity}$,
Next Fit uses at most
 $1 + (3/2)$ OPT bins



Message

1. From example to structural observation
2. With observation, upper bound algorithm
3. With different argument, lower bound OPT
4. Combine

.25

.3

.2

.29

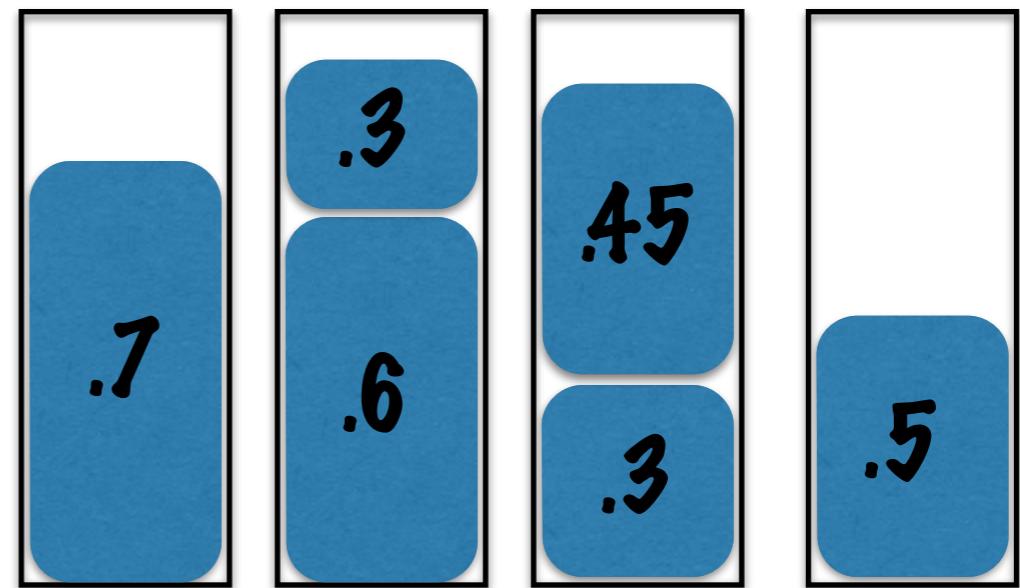
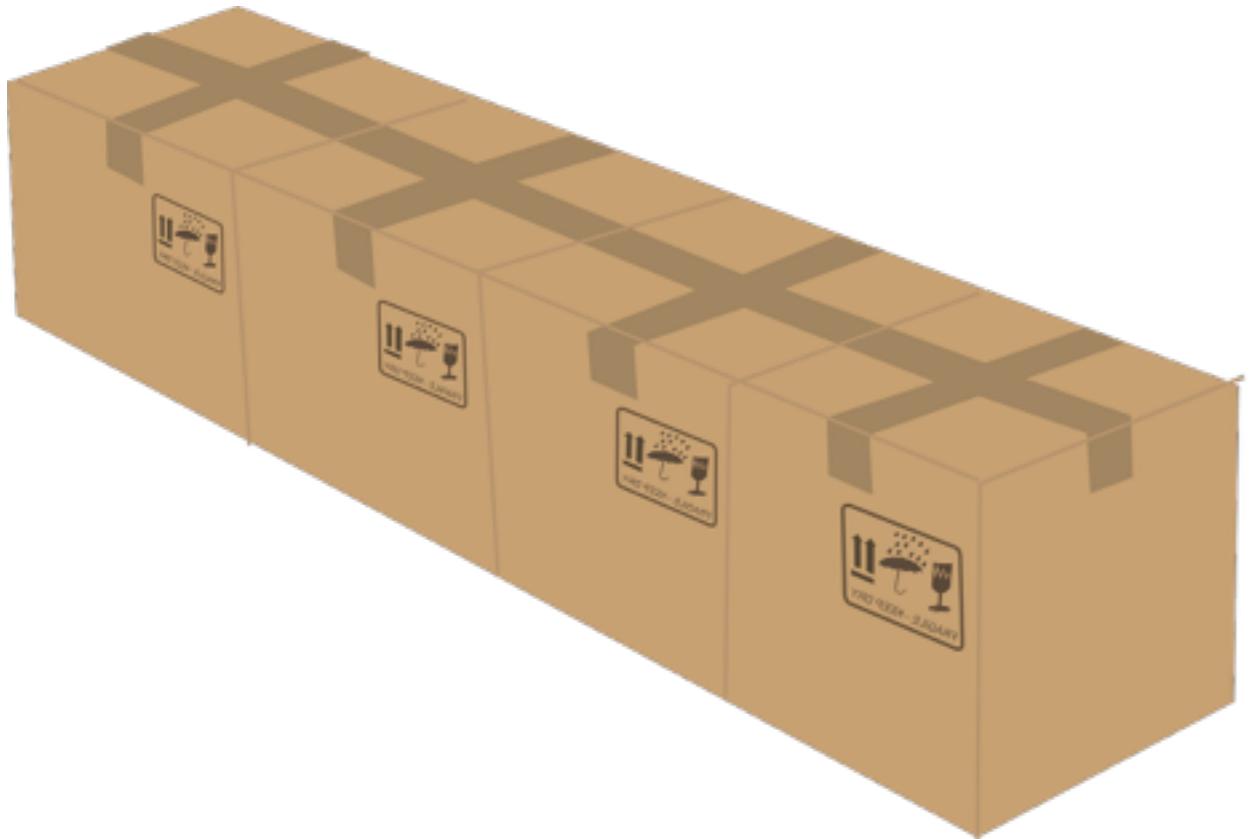
.24

.33

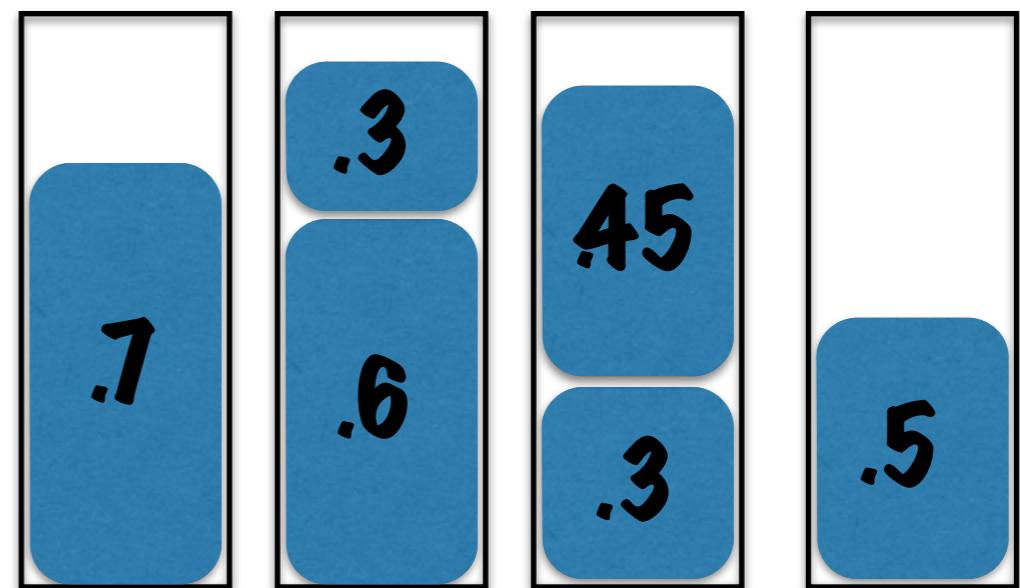
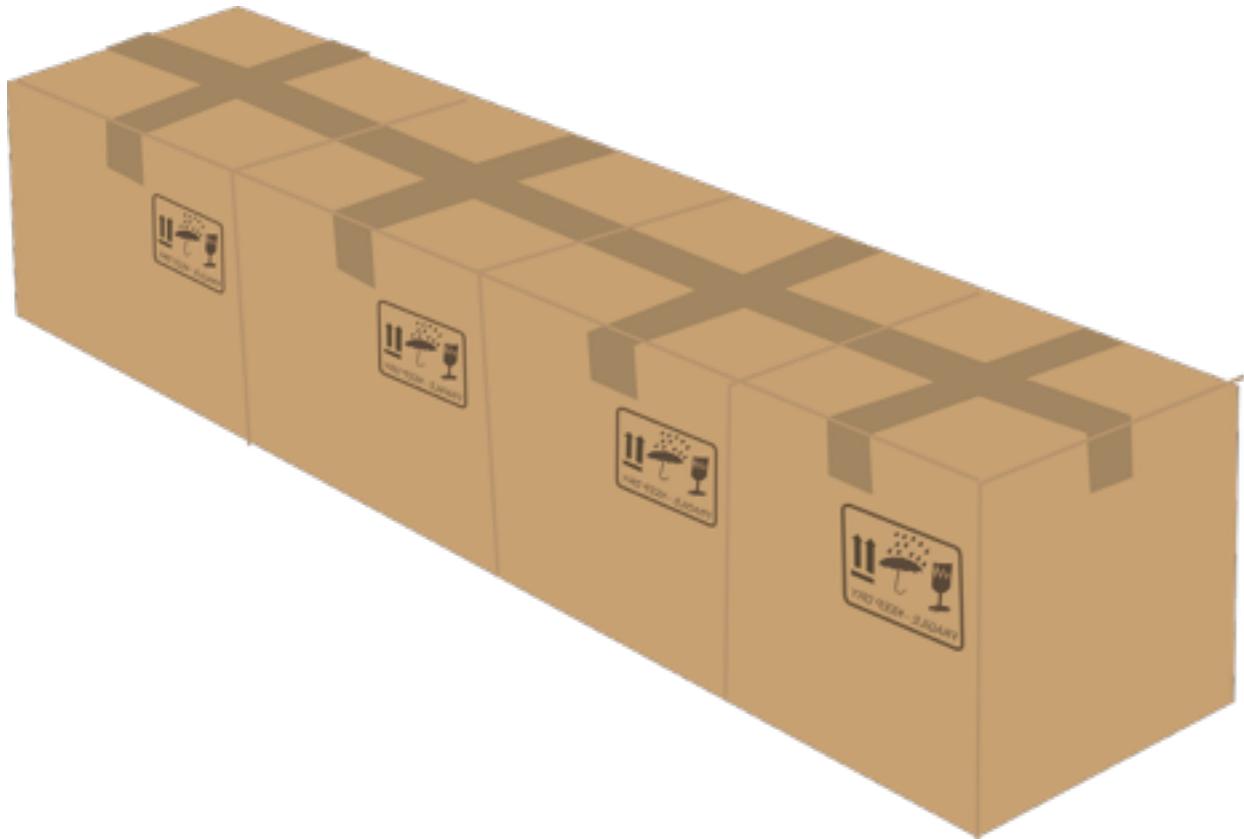
.2

.33

Bin packing, linear programming and rounding



Bin packing, linear programming and rounding



Meta-tool: special cases

Large items

Special special case

Large items,
few distinct sizes

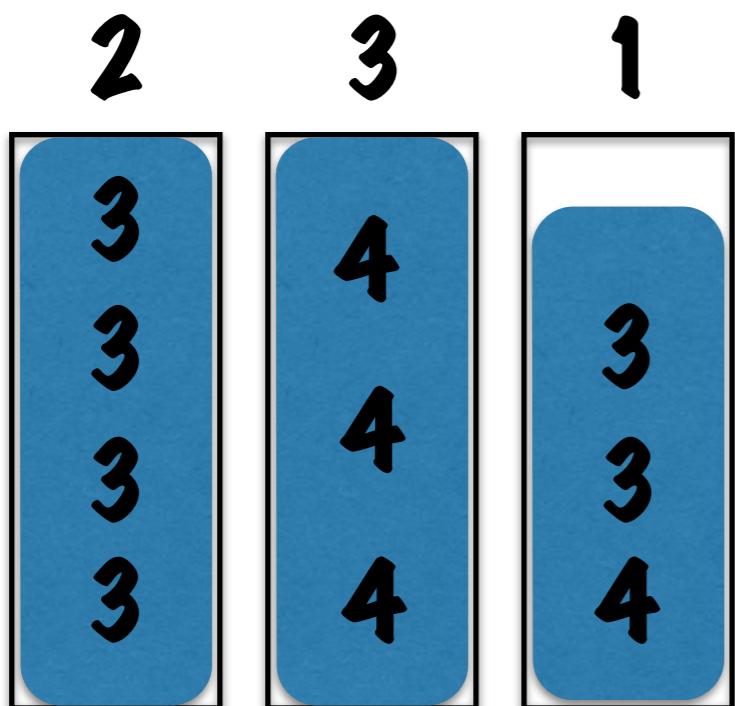
Example

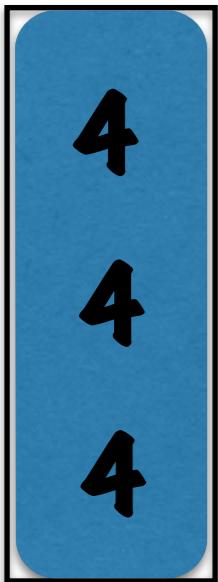
Bin capacity 12
sizes: { 3, 4 }
10 items of size 3,
10 items of size 4.

Bin capacity 12

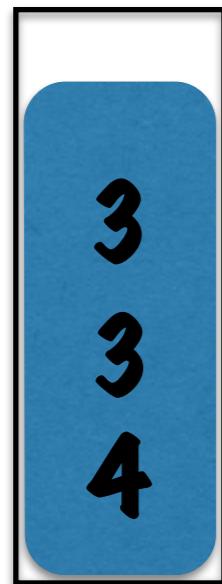
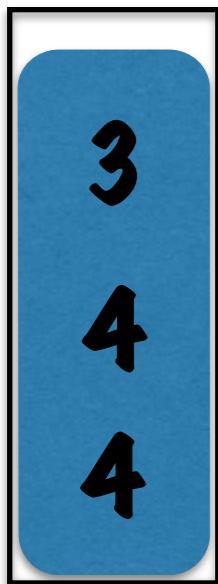
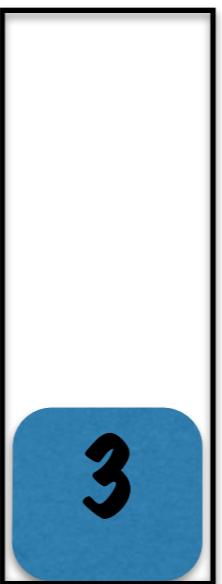
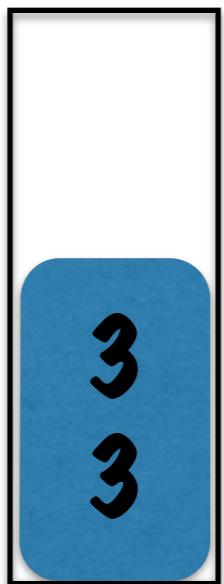
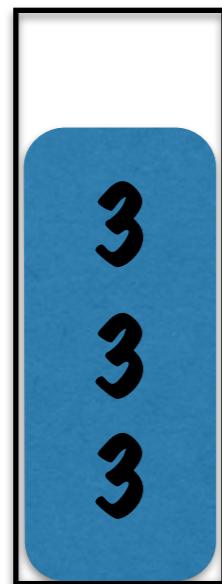
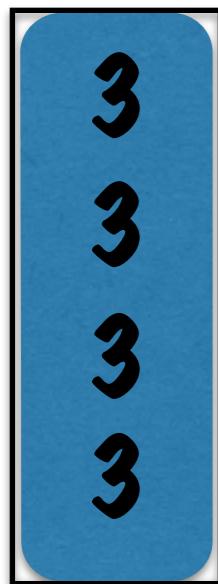
sizes: { 3, 4 }

**10 items of size 3,
10 items of size 4.**

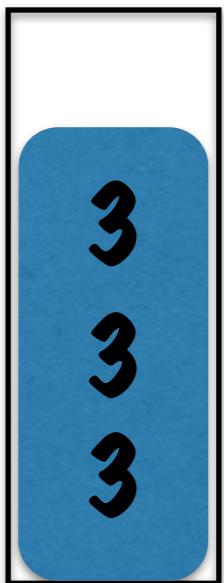
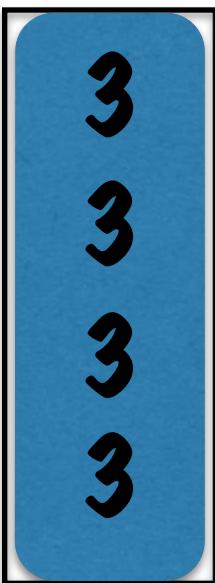




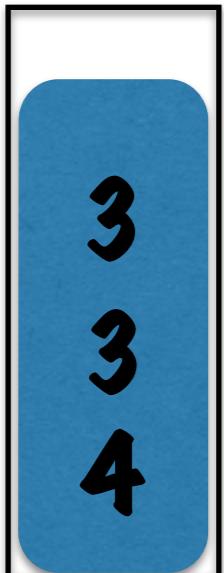
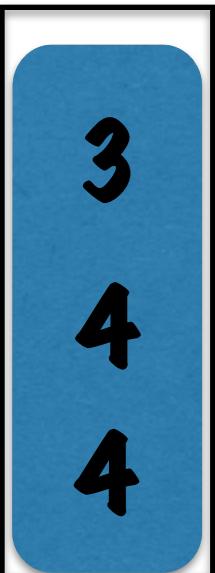
Observe:
few configurations



**Large items,
few distinct sizes
 $C=\{\text{configurations}\}$**



**In configuration c
size s occurs
 $a_{s,c}$ times**



Integer program

Input: $S=\{\text{size}\}$

number of items of size s : n_s

Output: $C=\{\text{configurations}\}$

number of bins in configuration c : x_c

Constraints: $\sum_c a_{s,c} x_c \geq n_s$

Number of bins: $\sum_c x_c$

integer

If size > capacity/10 then:
< 10 items per configuration

If < 10 sizes then:
< 10^{10} configurations

Solve LP relaxation
10 constraints, 10^{10} variables
Round up to nearest integer

#bins < OPT + 10^{10}

(Exhaustive search also ok)

For every $(x_c)_{c \in C} \in \{0, 1, \dots, n\}^{|C|}$

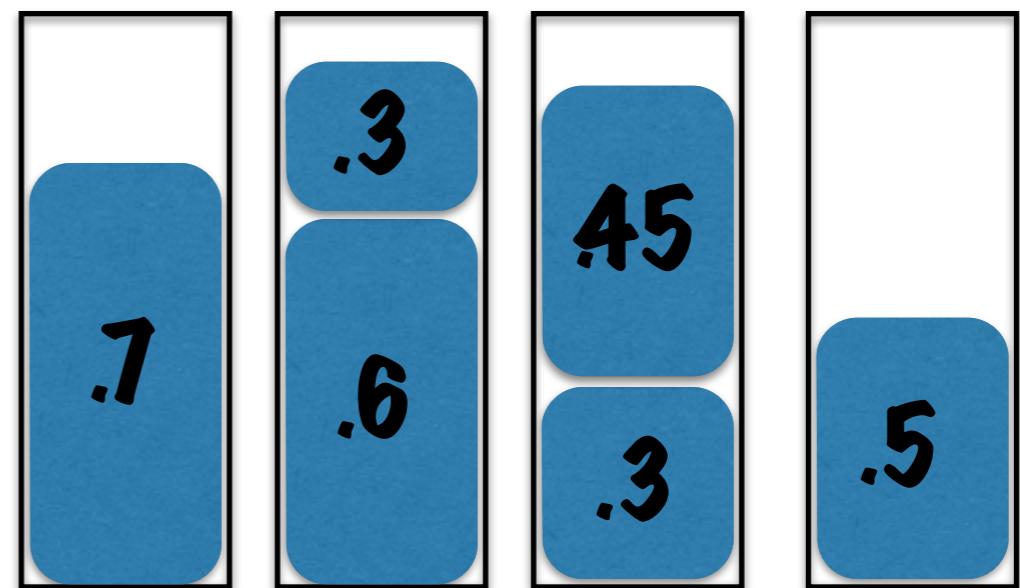
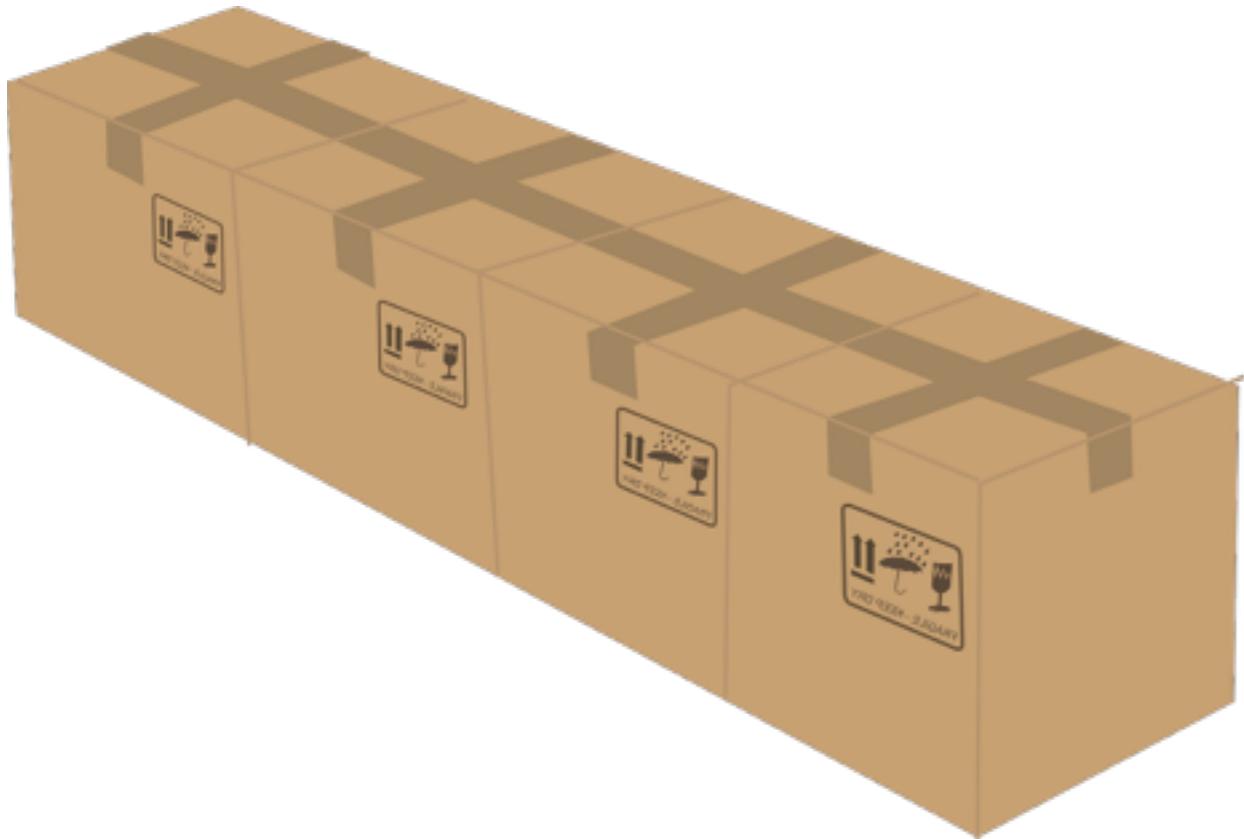
Check whether, for every size s,
enough slots for items of size s

Output solution with min #bins

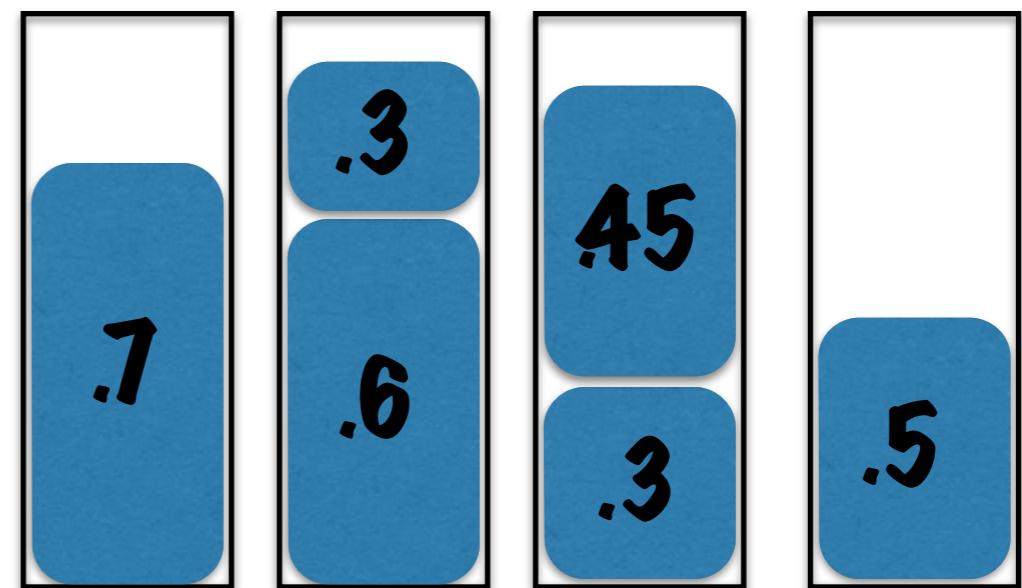
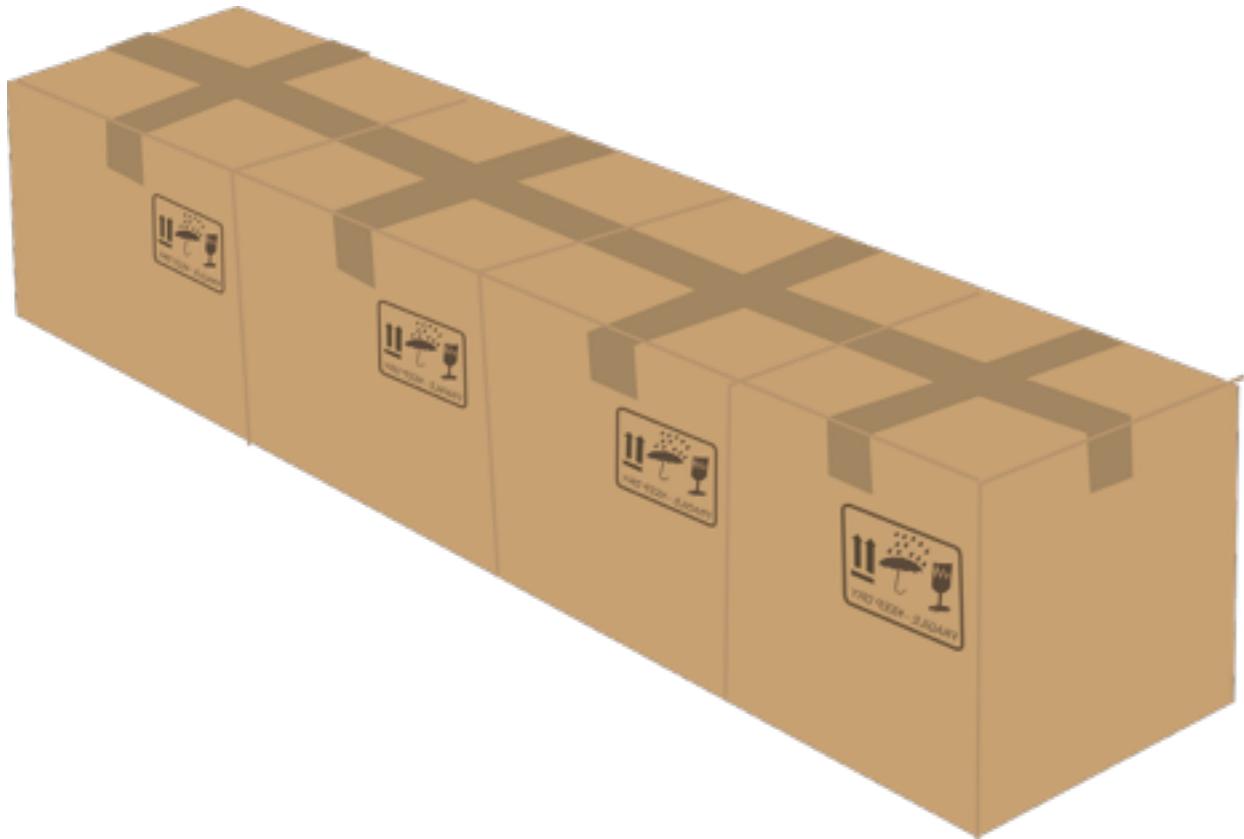
Runtime if size>capacity/10:

$$|S| \times n^{|S|^{10}}$$

Bin packing, linear programming and rounding



Bin packing, linear programming and rounding



Less special special case

**Large items,
many sizes**

Idea: Rounding

Round sizes

Reduce to special case

How to round?

Capacity = 100 & Min size > 10:

Attempt:

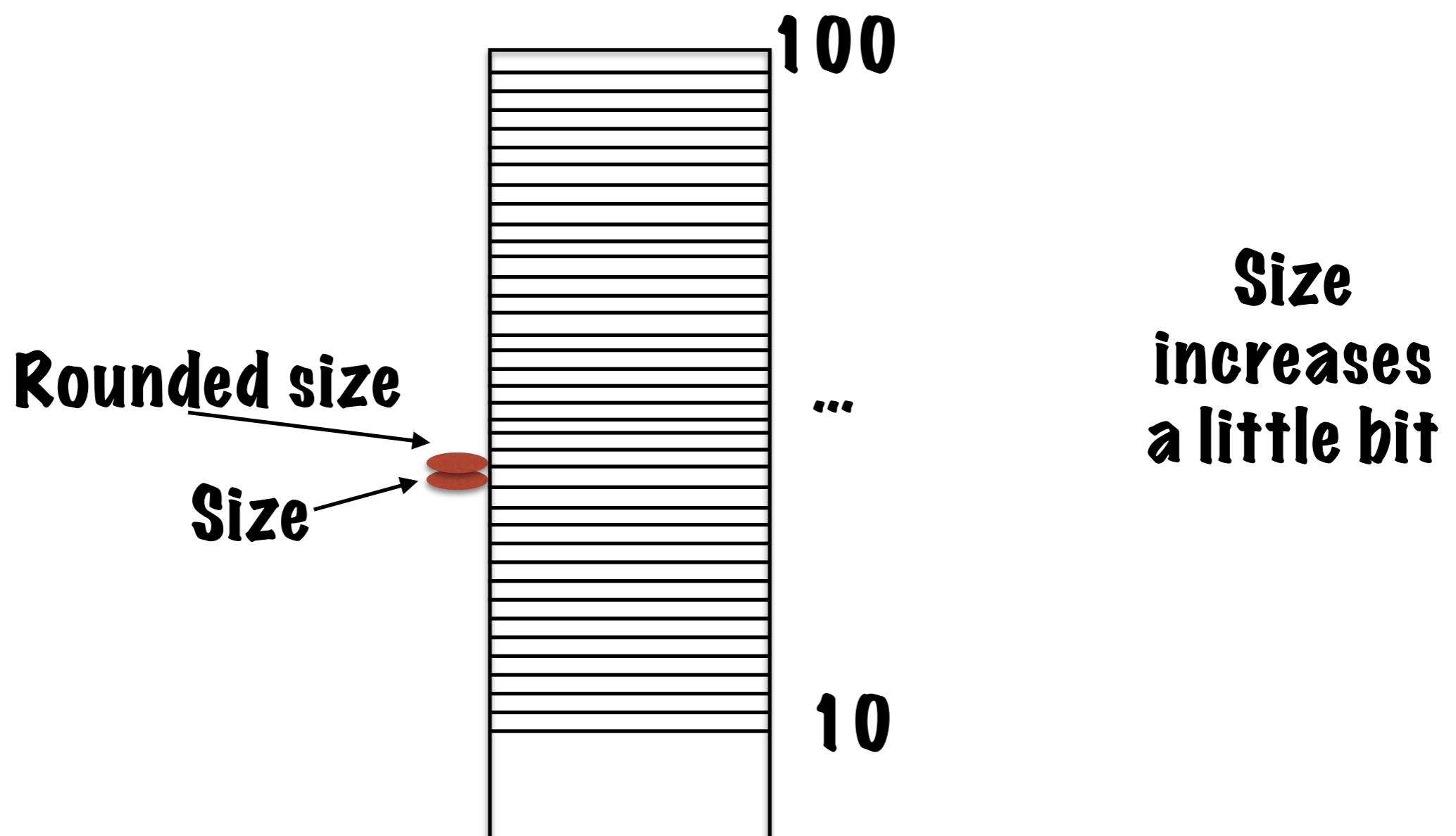
Round sizes **up** to nearest integer

Solve rounded problem

Observe:

It's a solution to original problem

But how good is it?



How much does OPT change?

Input:

Capacity 100

50 items with size $100*(1/3)$

50 items with size $100*(2/3)$

OPT=50

Rounded input:

Capacity 100

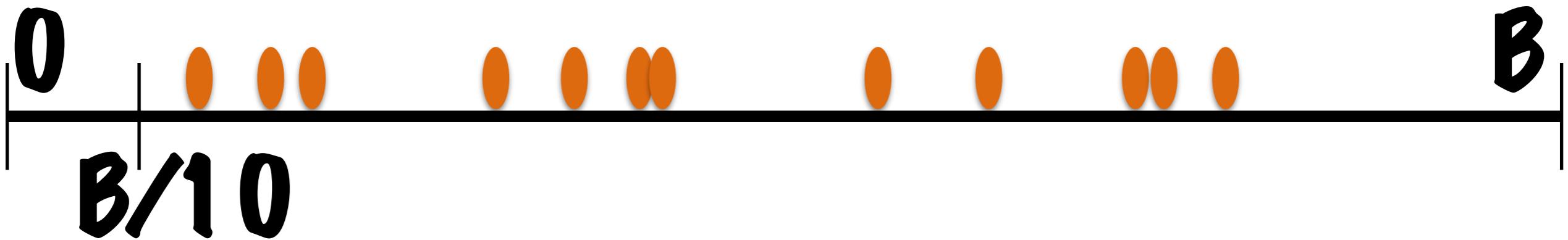
50 items with size 34

50 items with size 67

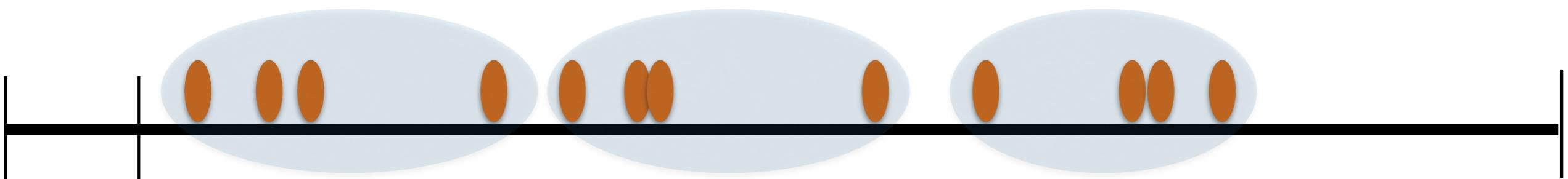
OPT'=75

This rounding fails

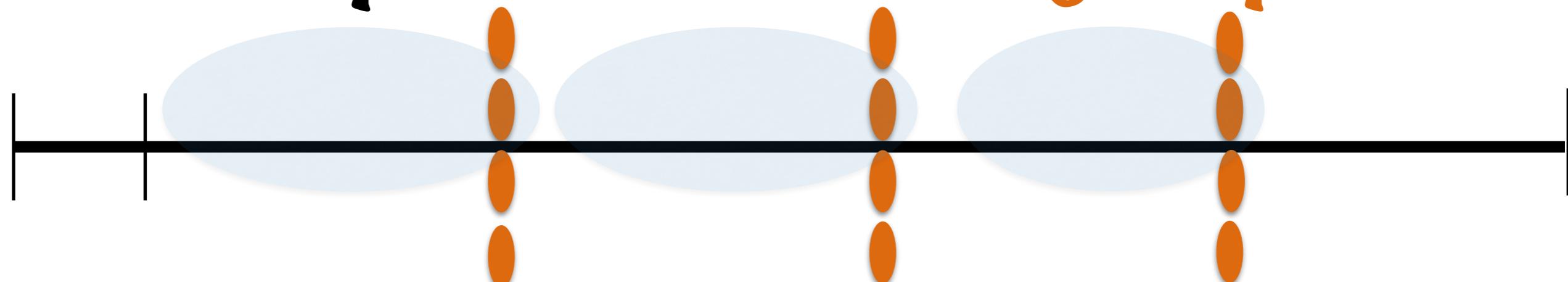
Adaptive rounding



Make groups of equal cardinality



Round up to max size in group



Algorithm - large items

Assume: sizes > capacity * ϵ

Sort sizes

Make groups of cardinality $n \times \epsilon^2$

Round up to max size in group

Solve rounded problem

Output corresponding packing

Observe: output is a packing

Observe: all sizes are $> \text{Capacity} \times \epsilon$

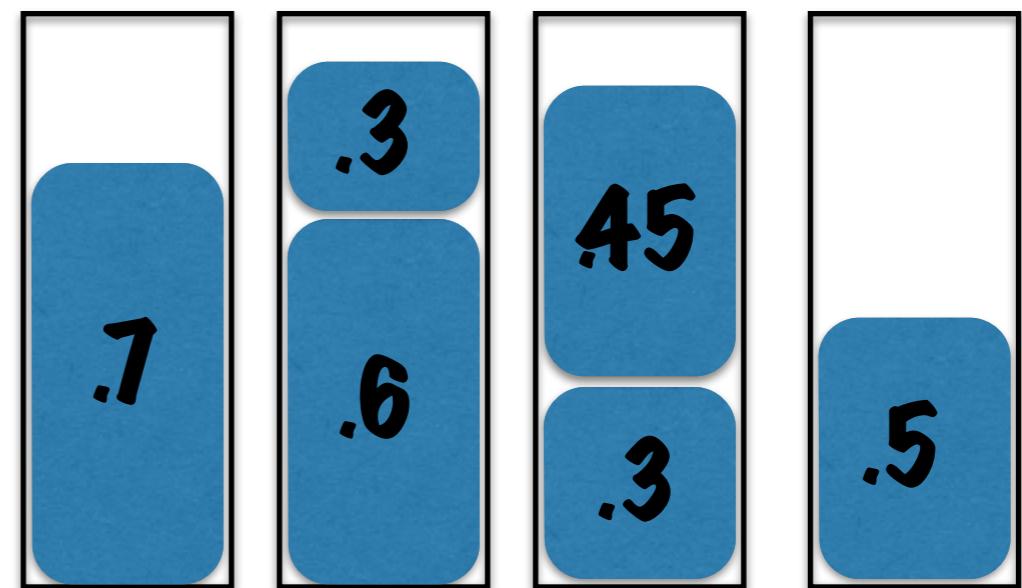
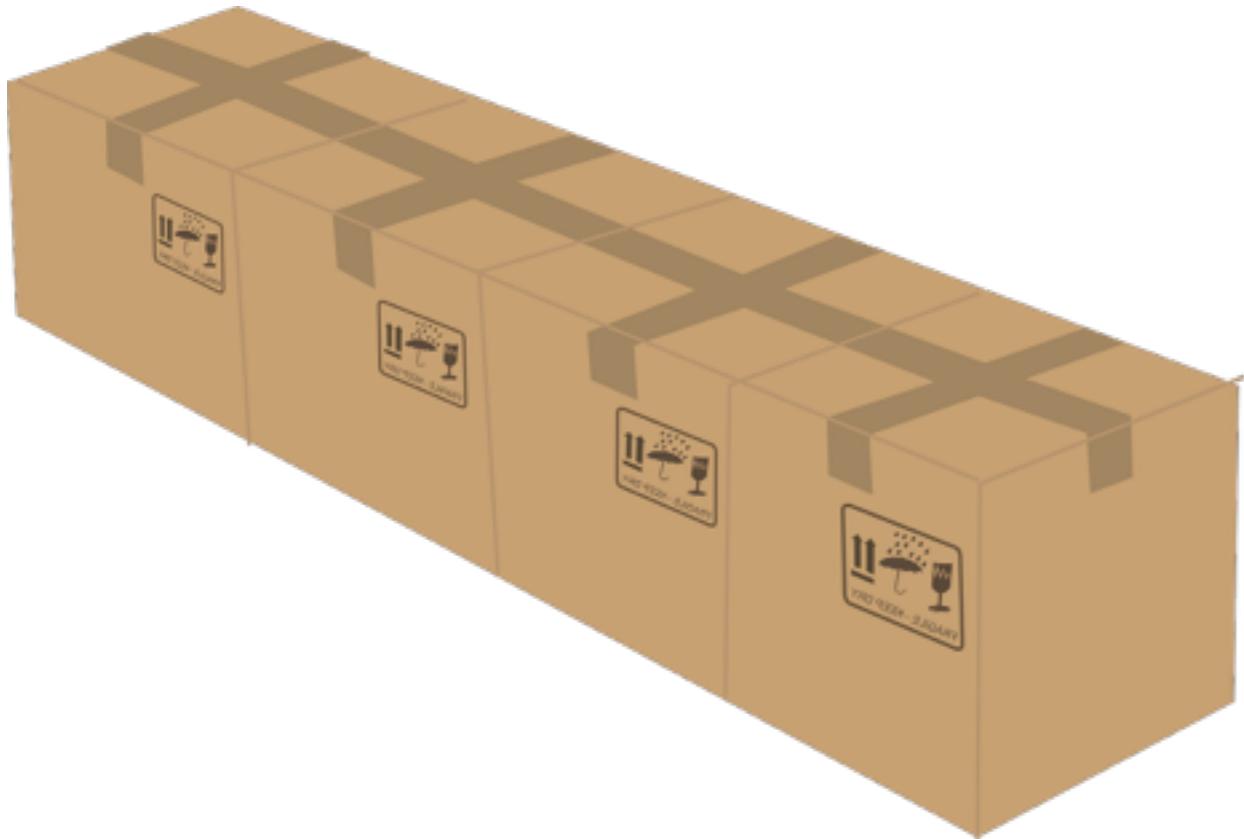
Observe: #distinct sizes $< 1/\epsilon^2$

Runtime : polynomial

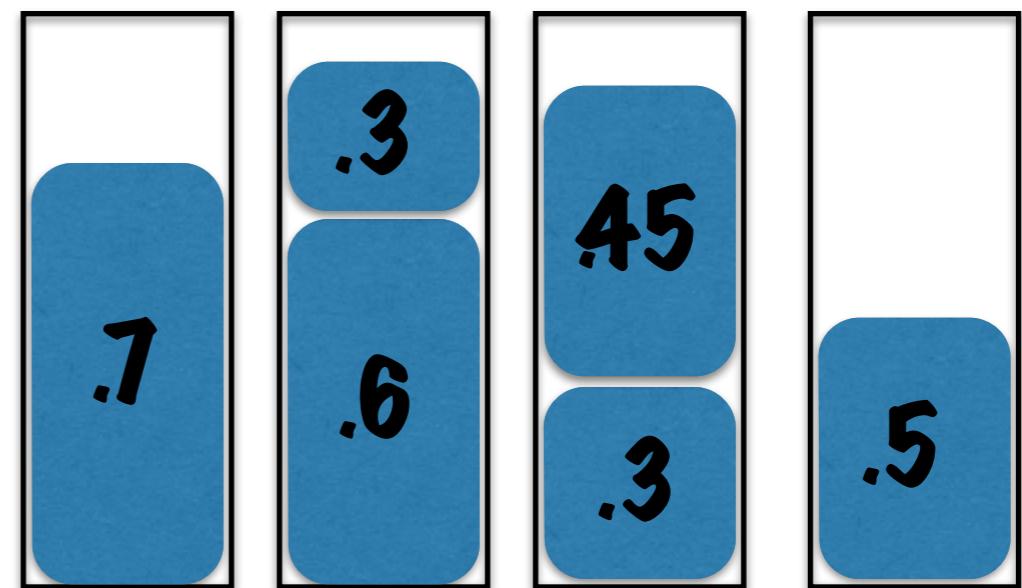
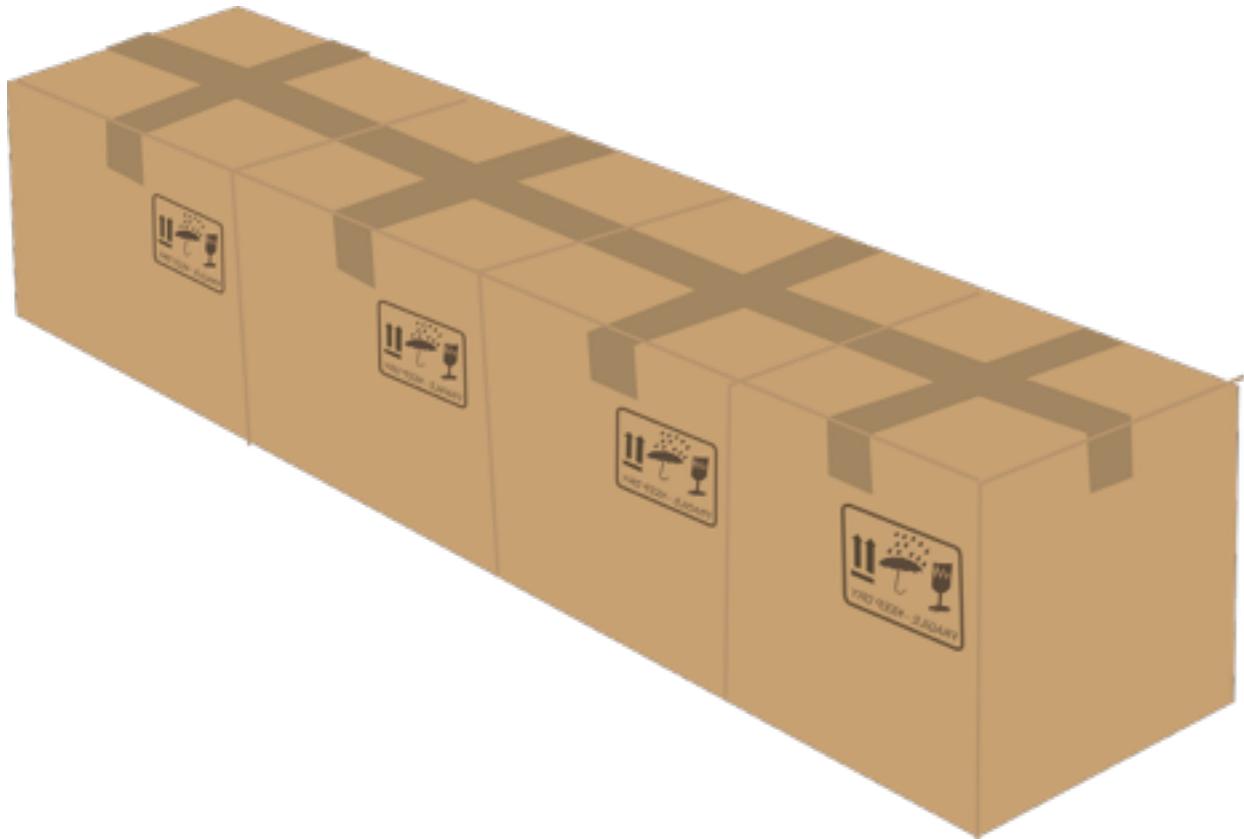
But how good is it?

$$\text{Value}(\text{Output}) < \text{OPT} * (1 + O(\epsilon))$$

Bin packing, linear programming and rounding



Bin packing, linear programming and rounding



Algorithm - large items

Assume: sizes > capacity * ϵ

Sort sizes

Make groups of cardinality $n \times \epsilon^2$

Round up to max size in group

Solve rounded problem U

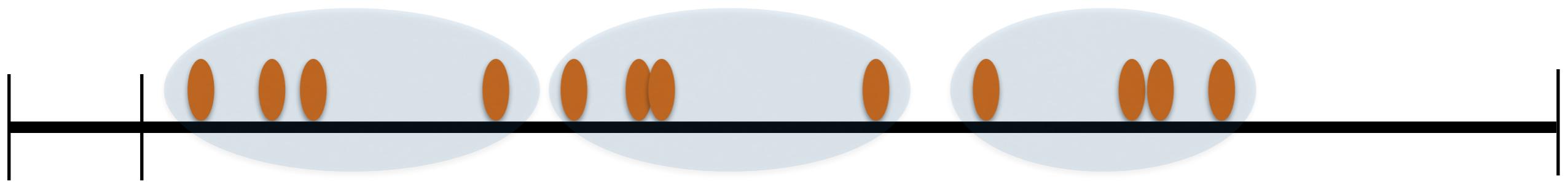
Output corresponding packing

But how good is it?

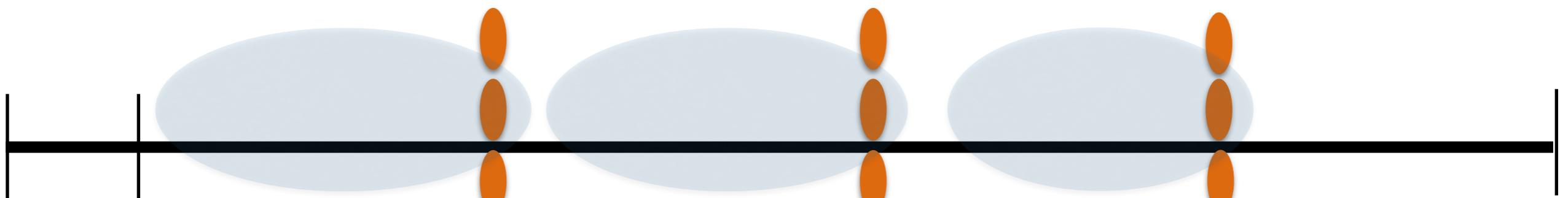
$\text{Value}(\text{Output}) = \text{OPT}(U)$

Must relate $\text{OPT}(U)$ to $\text{OPT}(I)$

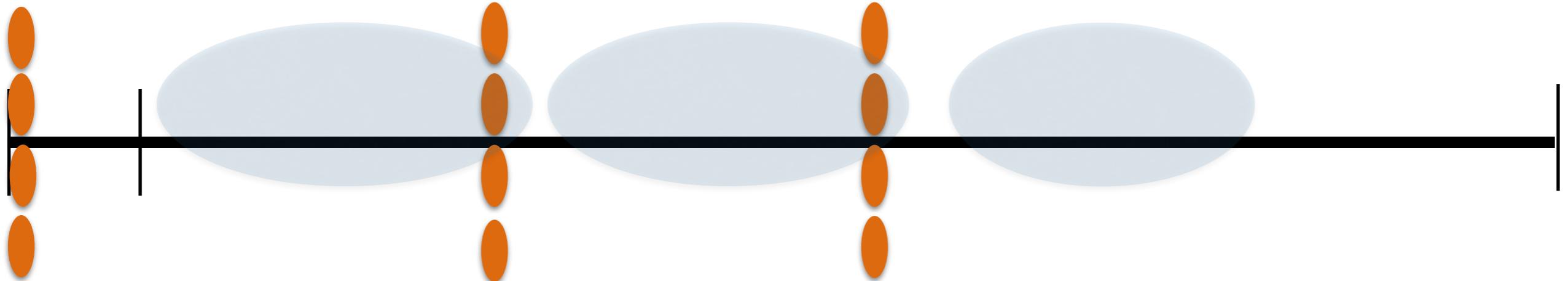
I: Input



U: Round **up**: max of group



D: Round **down**: max of previous group



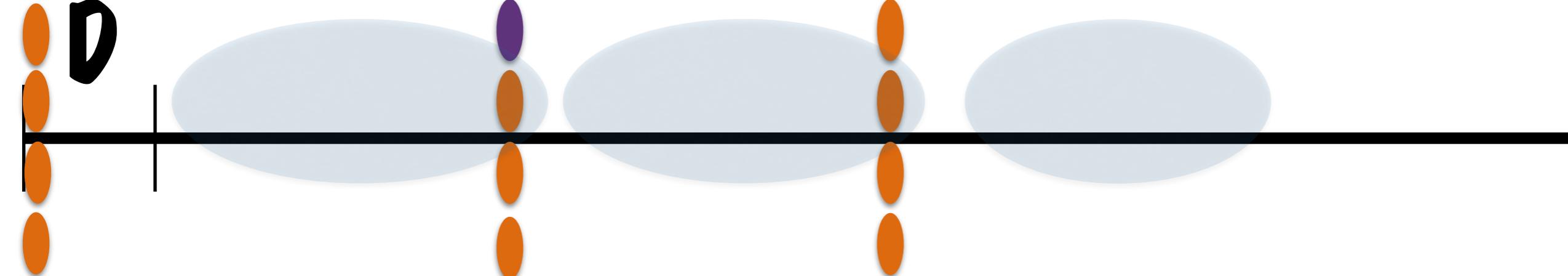
U



I



D



up

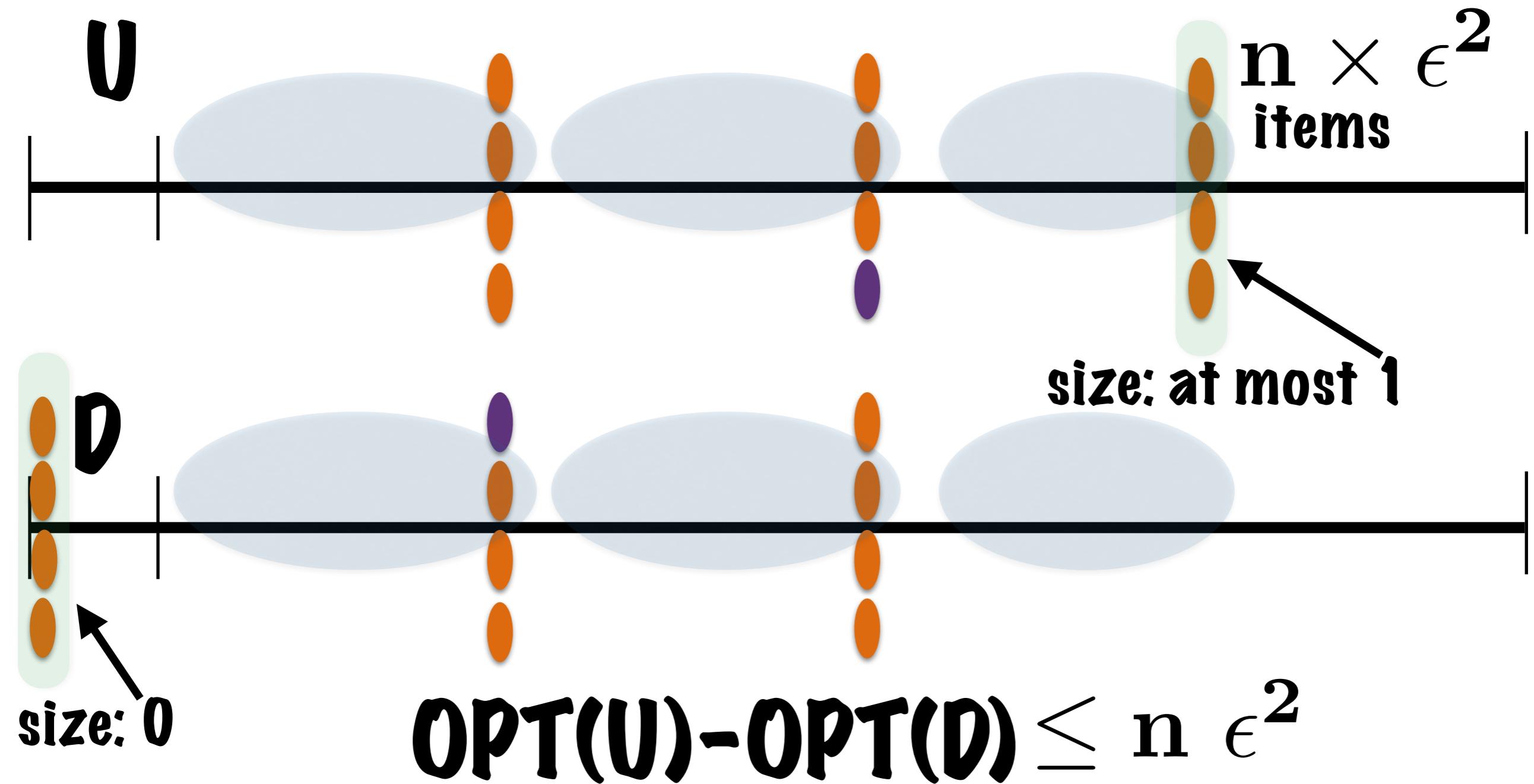
down

Relating input to rounded input

Observe:
Increasing sizes
can only increase OPT

$$\text{OPT}(D) \leq \text{OPT}(I) \leq \text{OPT}(U)$$

U and D are similar!



Combine:

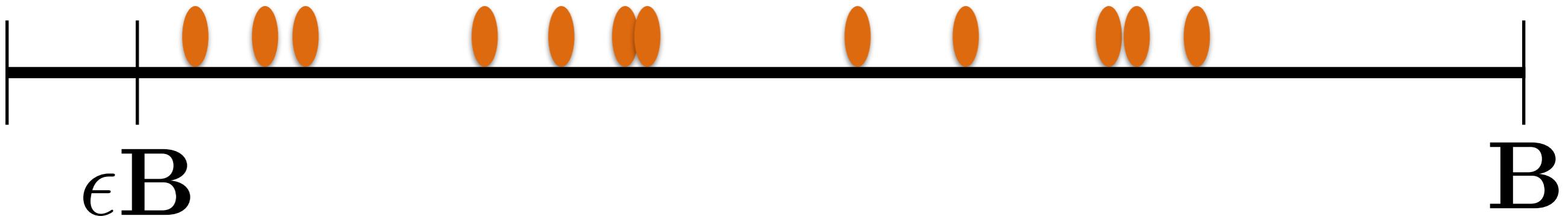
$$\text{OPT}(D) \leq \text{OPT}(I) \leq \text{OPT}(U)$$

$$\text{OPT}(U) - \text{OPT}(D) \leq n \epsilon^2$$

$$\text{OPT}(U) \leq \text{OPT}(I) + n \epsilon^2$$

Additive error $n \epsilon^2$

Lower bound OPT



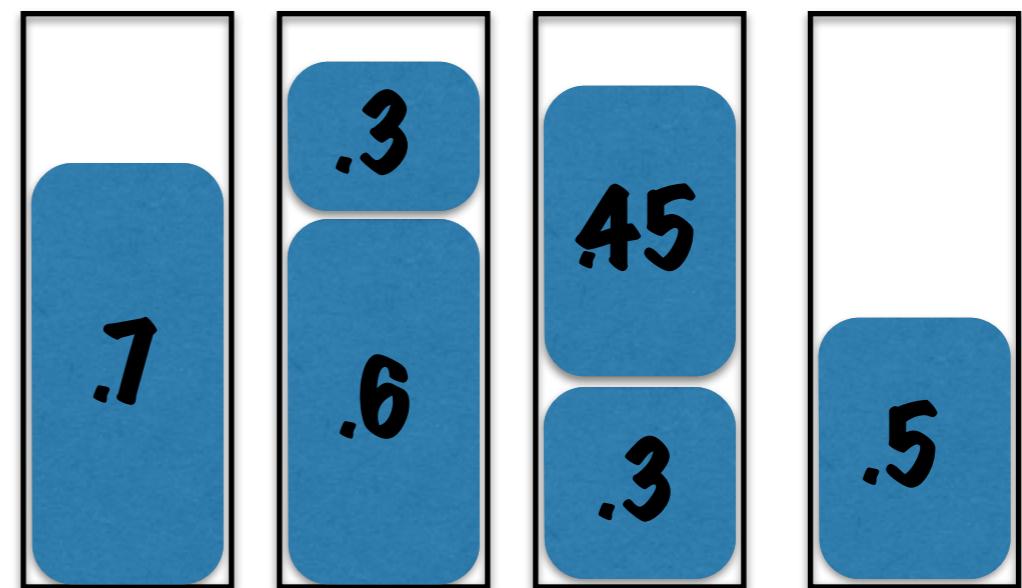
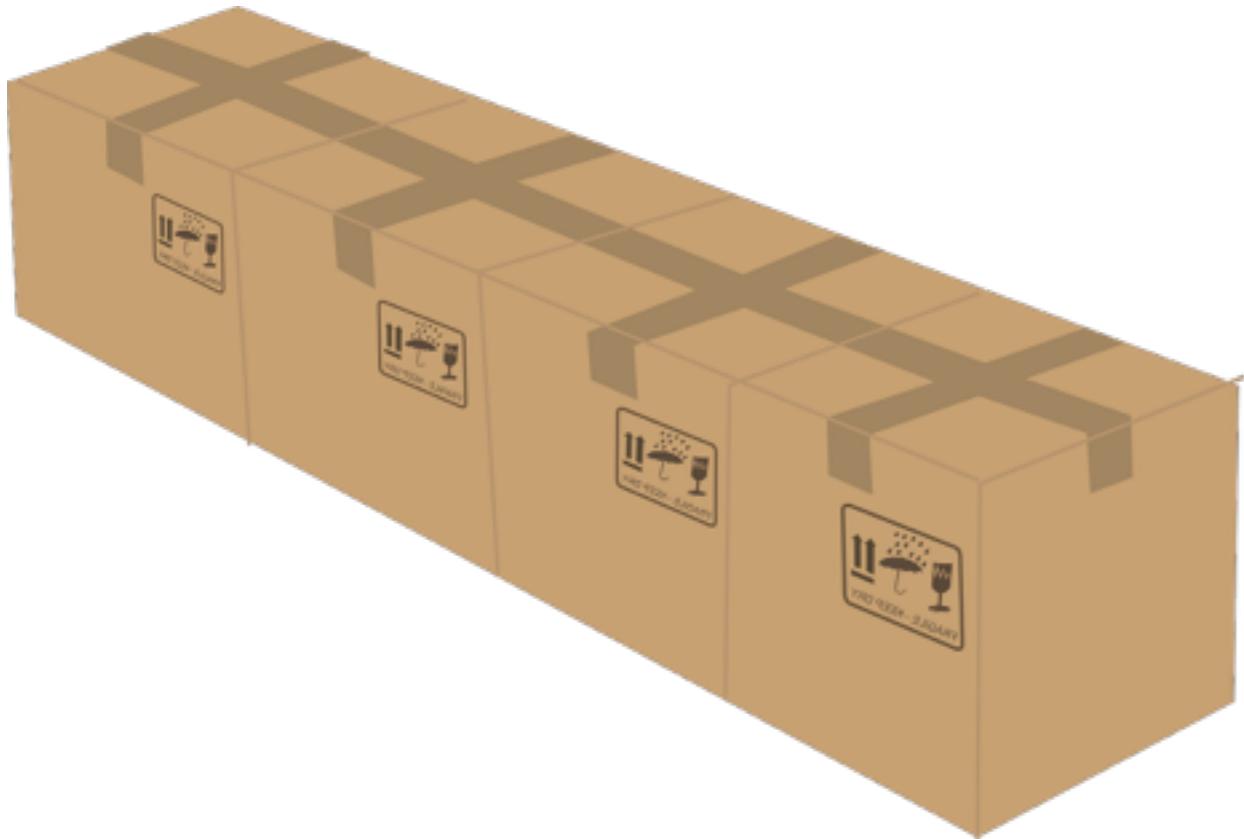
n items
max #items per bin: $1/\epsilon$
—————
min #bins: ϵn

$$n\epsilon^2 \leq \epsilon \times (n\epsilon) \leq \epsilon \text{ OPT}$$

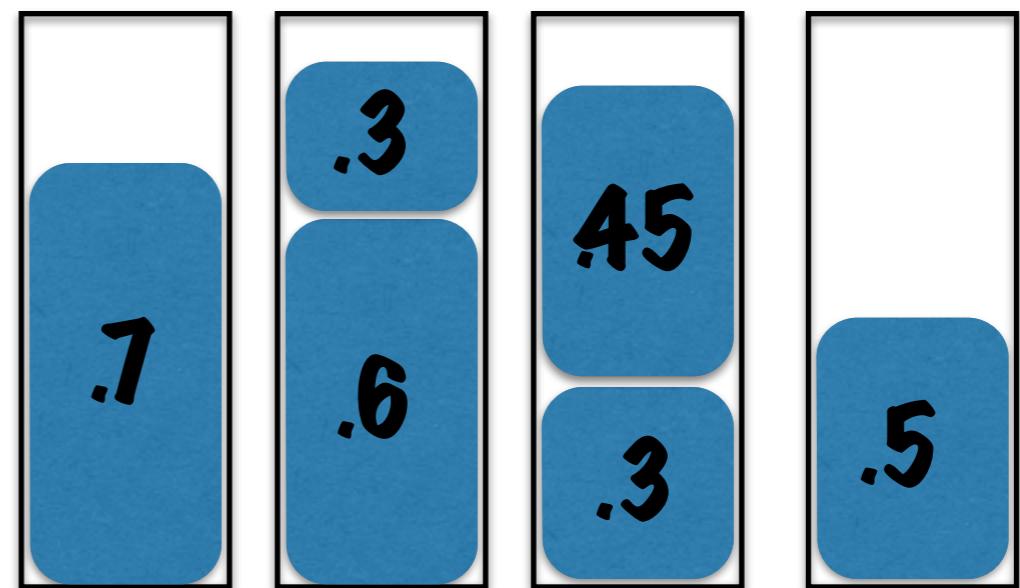
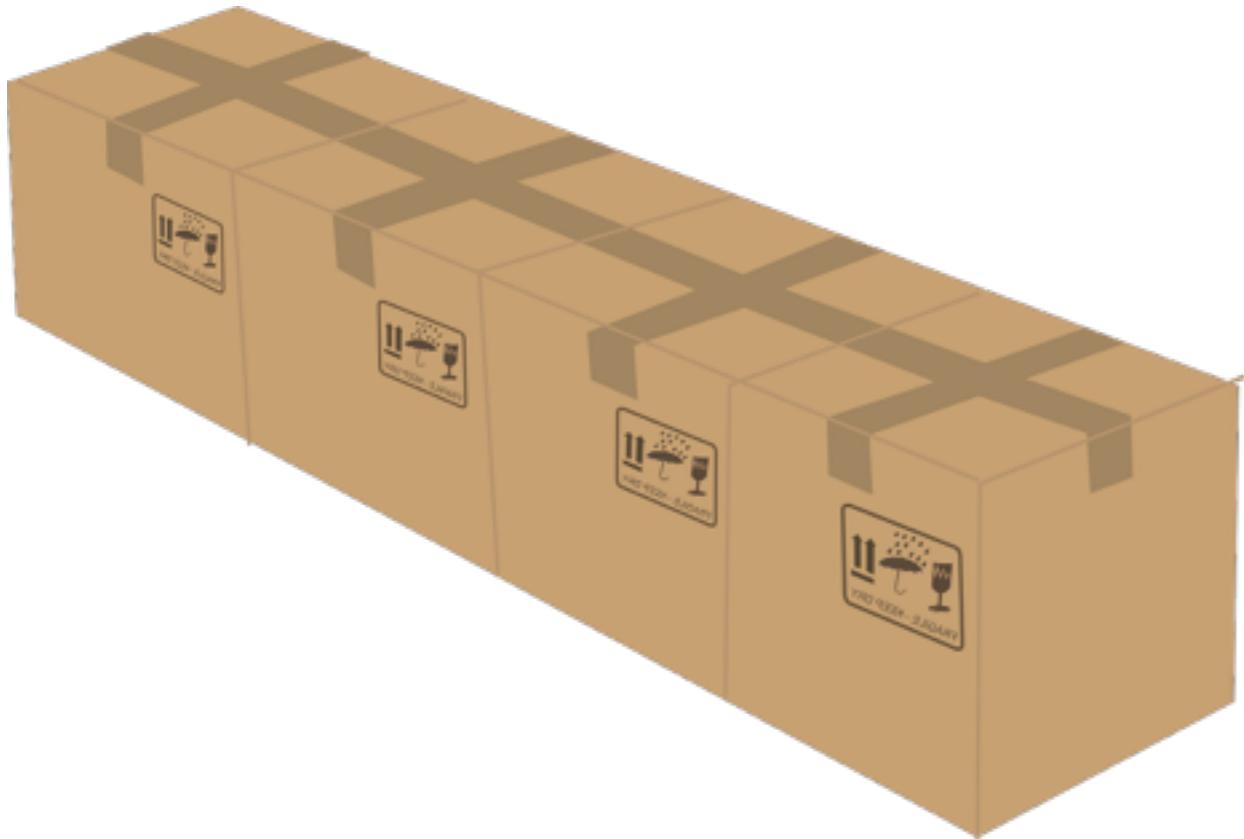
Theorem

When all sizes are $> \epsilon B$
algorithm, in polynomial time
gives packing s.t.
 $\text{Value}(\text{Output}) < \text{OPT} * (1 + \epsilon)$

Bin packing, linear programming and rounding



Bin packing, linear programming and rounding



General algorithm

Set aside: sizes < cap. * ϵ (small)

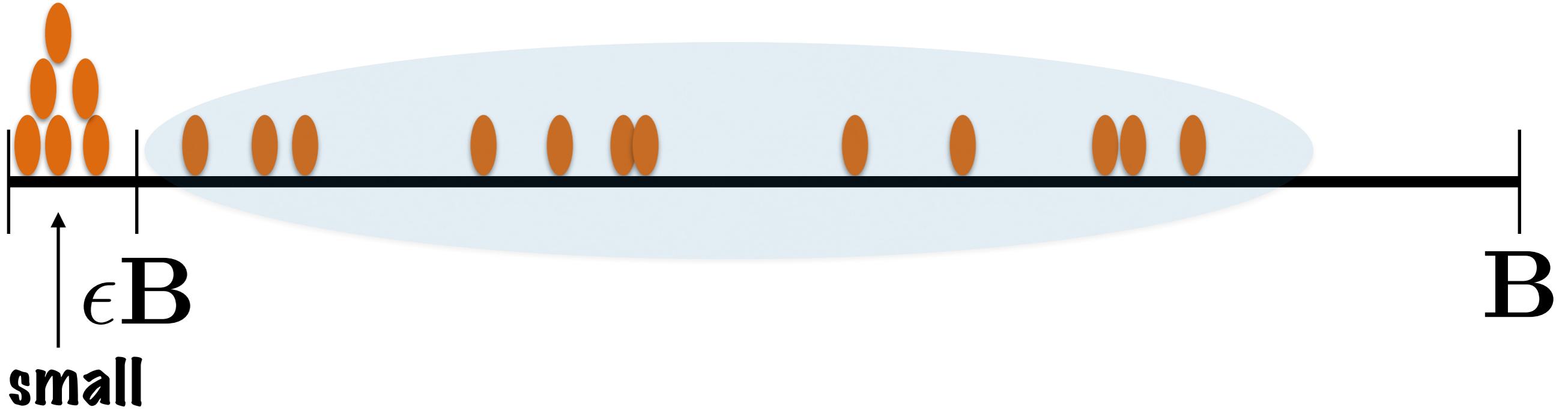
Sort remaining sizes

Make groups of cardinality $n \times \epsilon^2$

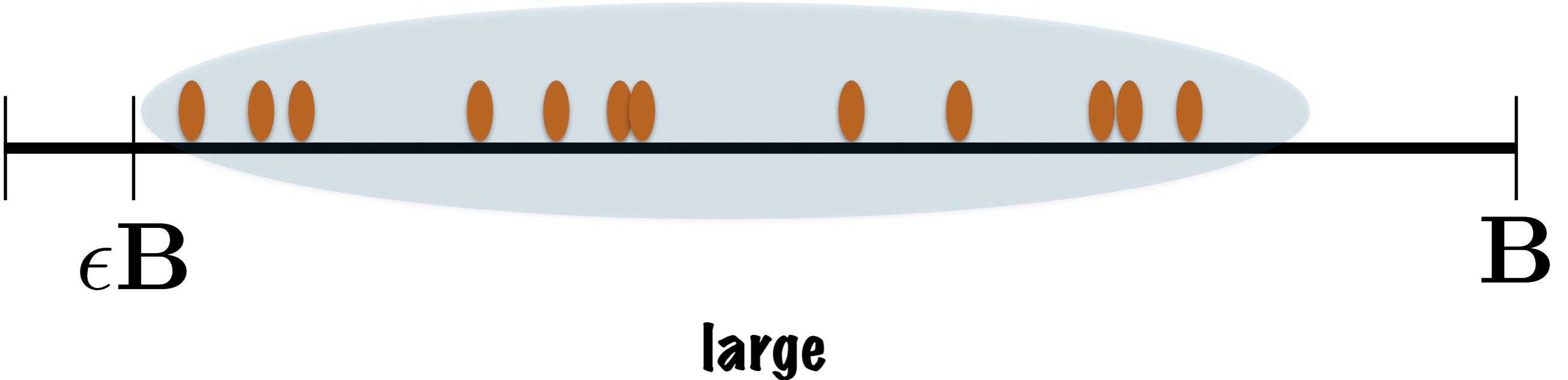
Round up to max size in group

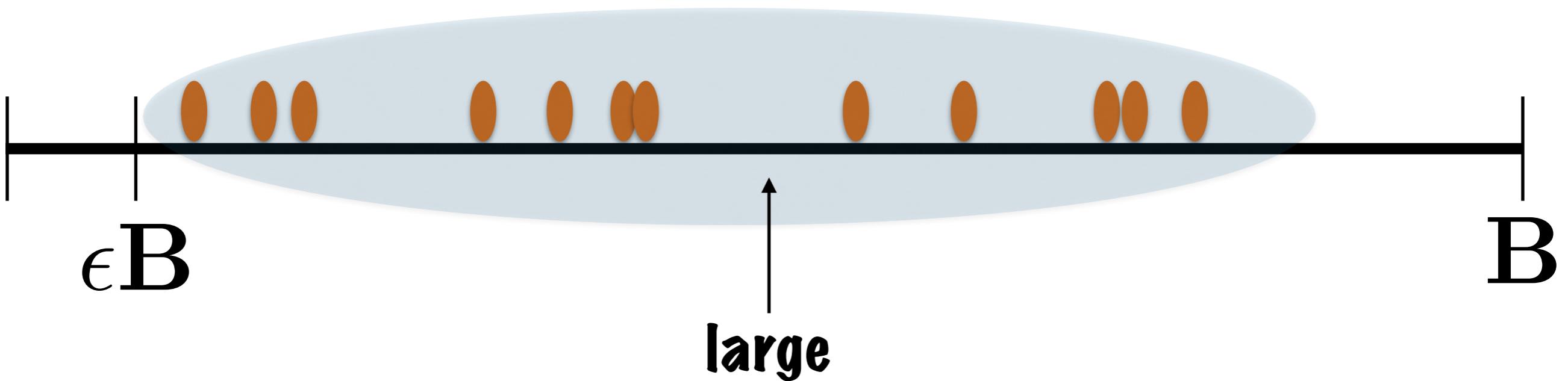
Solve rounded problem U

Greedily add small sizes

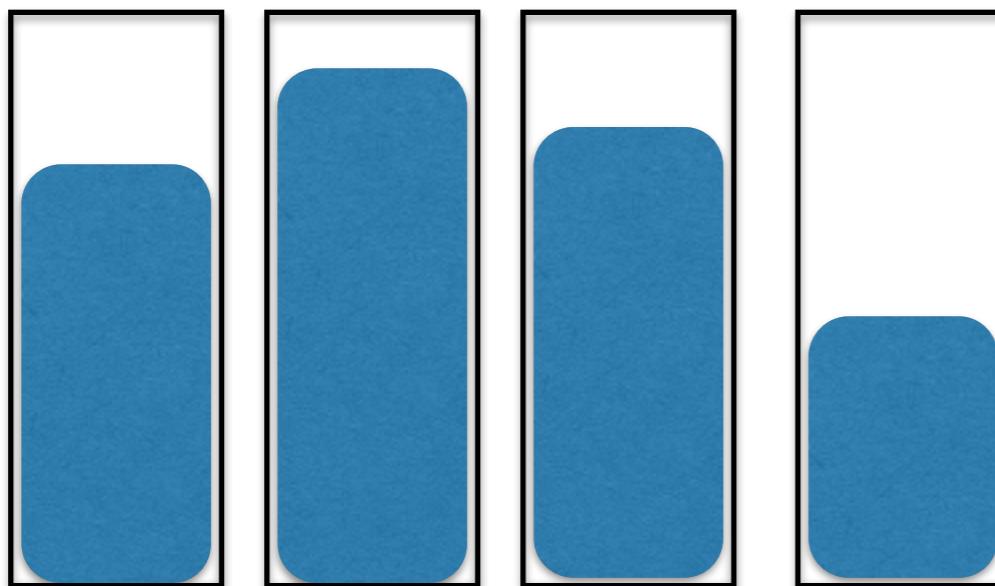


Set aside: sizes $< B^\epsilon$ (small)

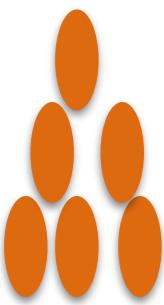




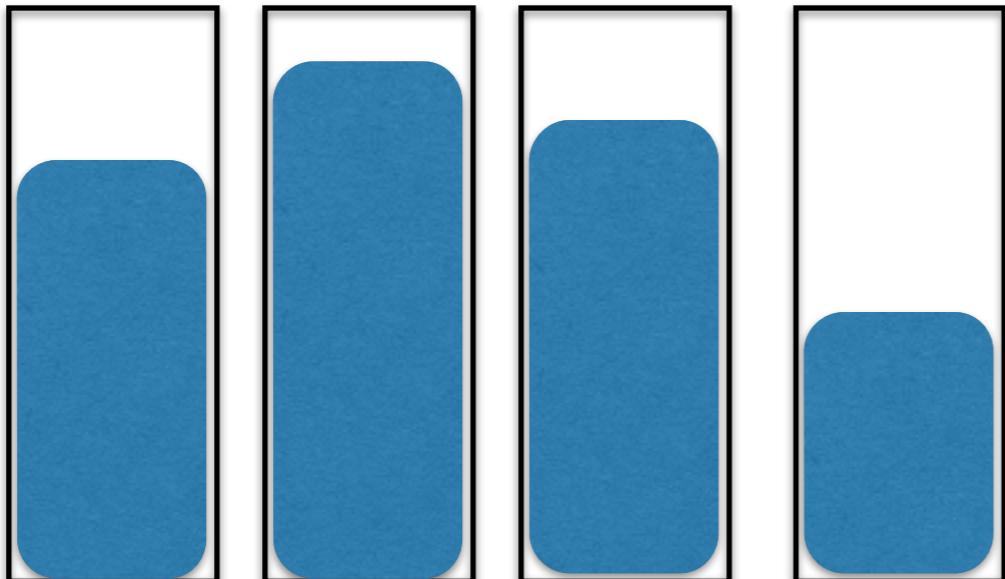
Solve for remaining sizes



**Packing of
large items**

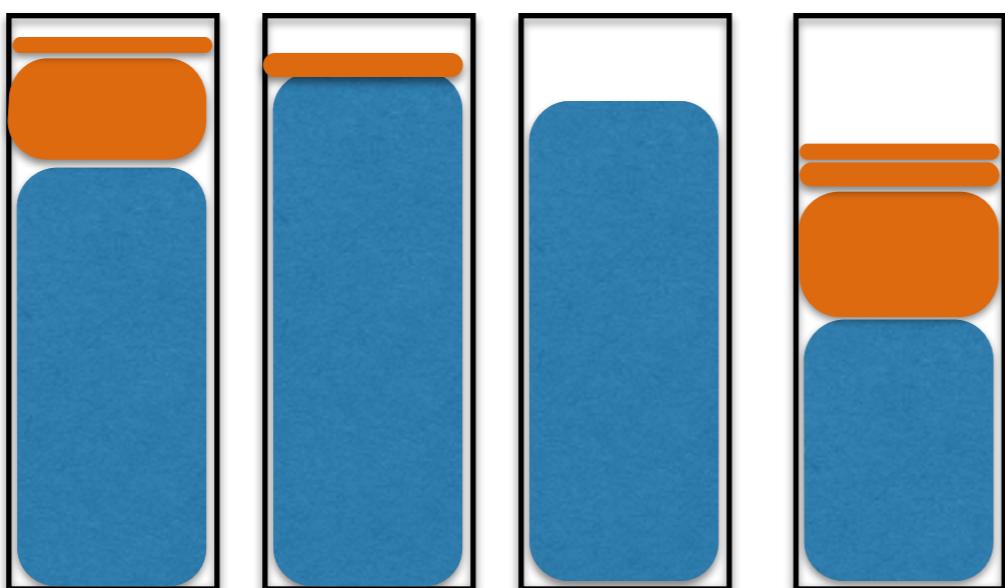


+
**small
items**



**Packing of
large items**

Greedily add small sizes



Analysis

Input $I = S \cup L$

Case 1

No new bins opened by S :
then

$$\begin{aligned}\text{Value(Output)} &= \\ \text{Value(packing of } L) &\leq (1 + \epsilon) \cdot \text{OPT}(L) \\ &\leq (1 + \epsilon) \cdot \text{OPT}(I)\end{aligned}$$

Case 2

Some new bin opened by S:
then all bins except last
are filled to B times

$$\geq 1 - \epsilon$$

$$(1/B) \sum s_i \geq (\#bins - 1)(1 - \epsilon)$$
$$(1/B) \sum s_i \leq OPT$$

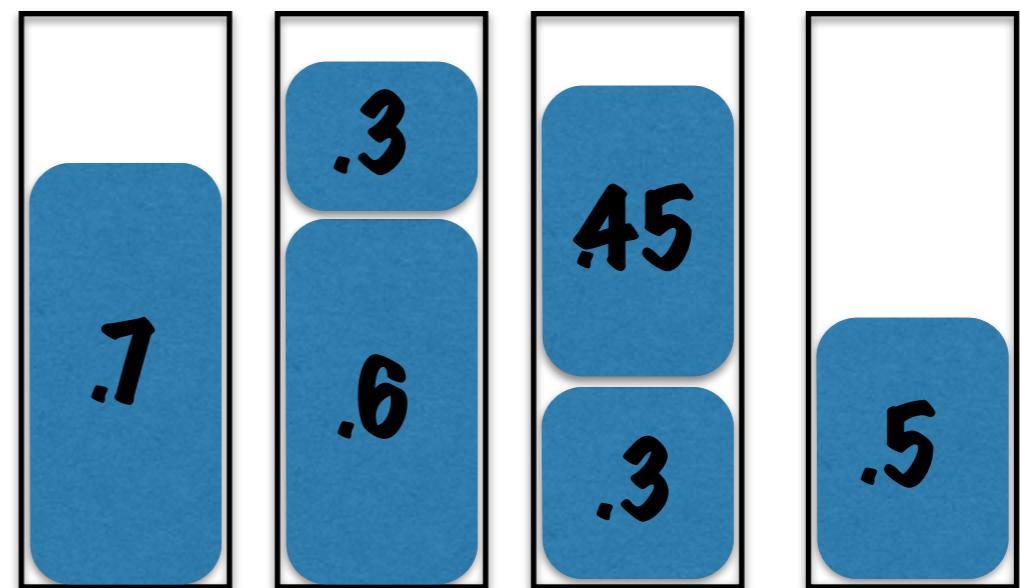
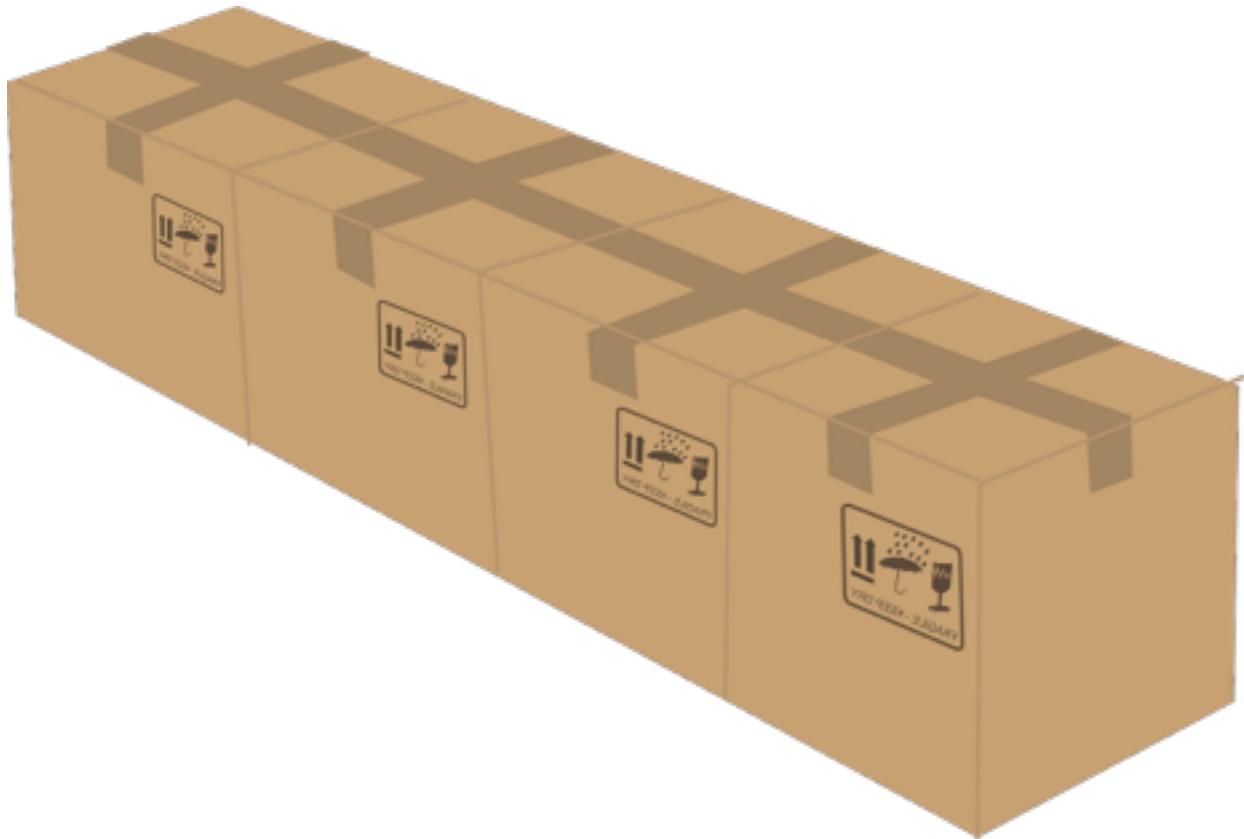
$$\text{Value(Output)} \leq \frac{1}{1-\epsilon} OPT + 1$$

Theorem

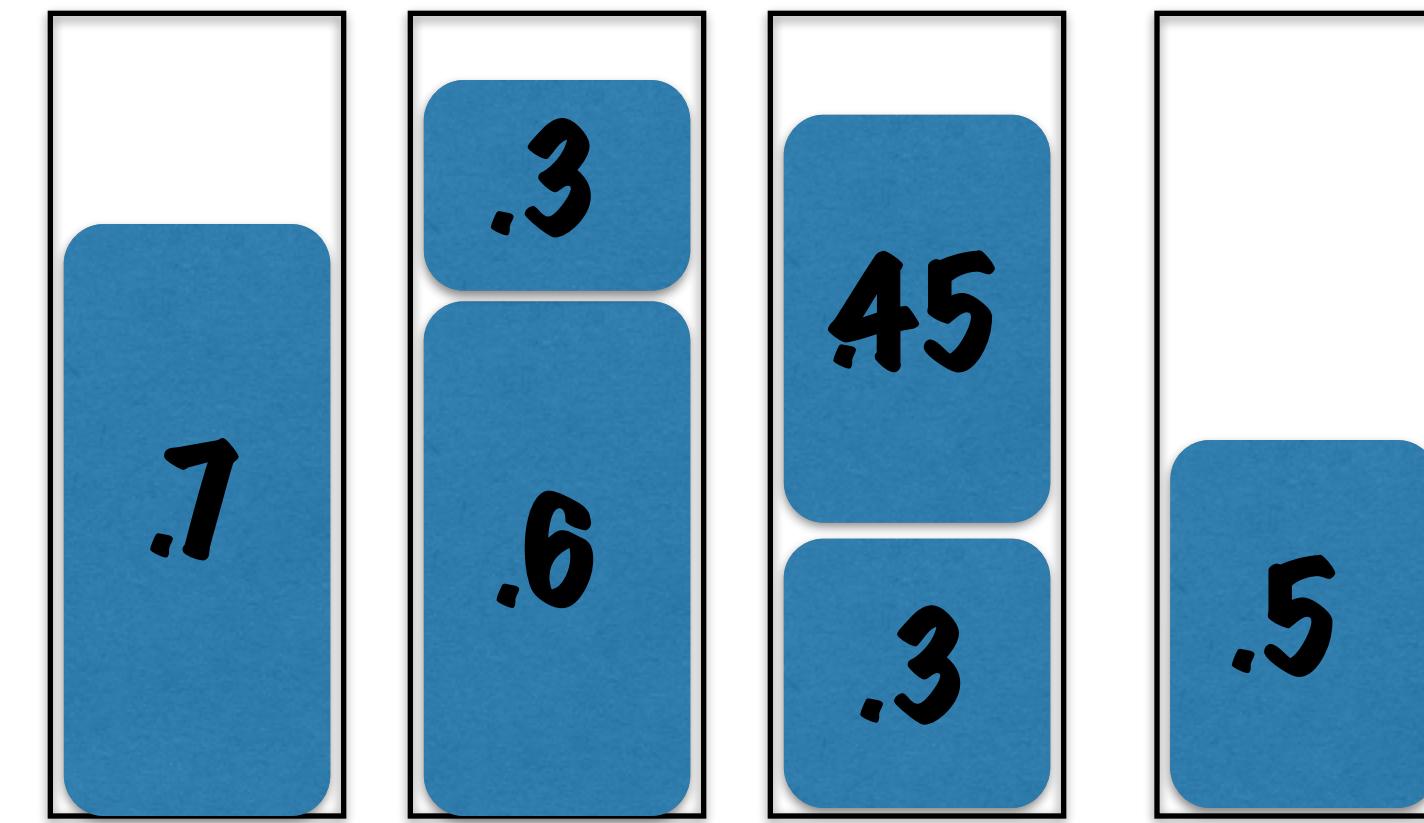
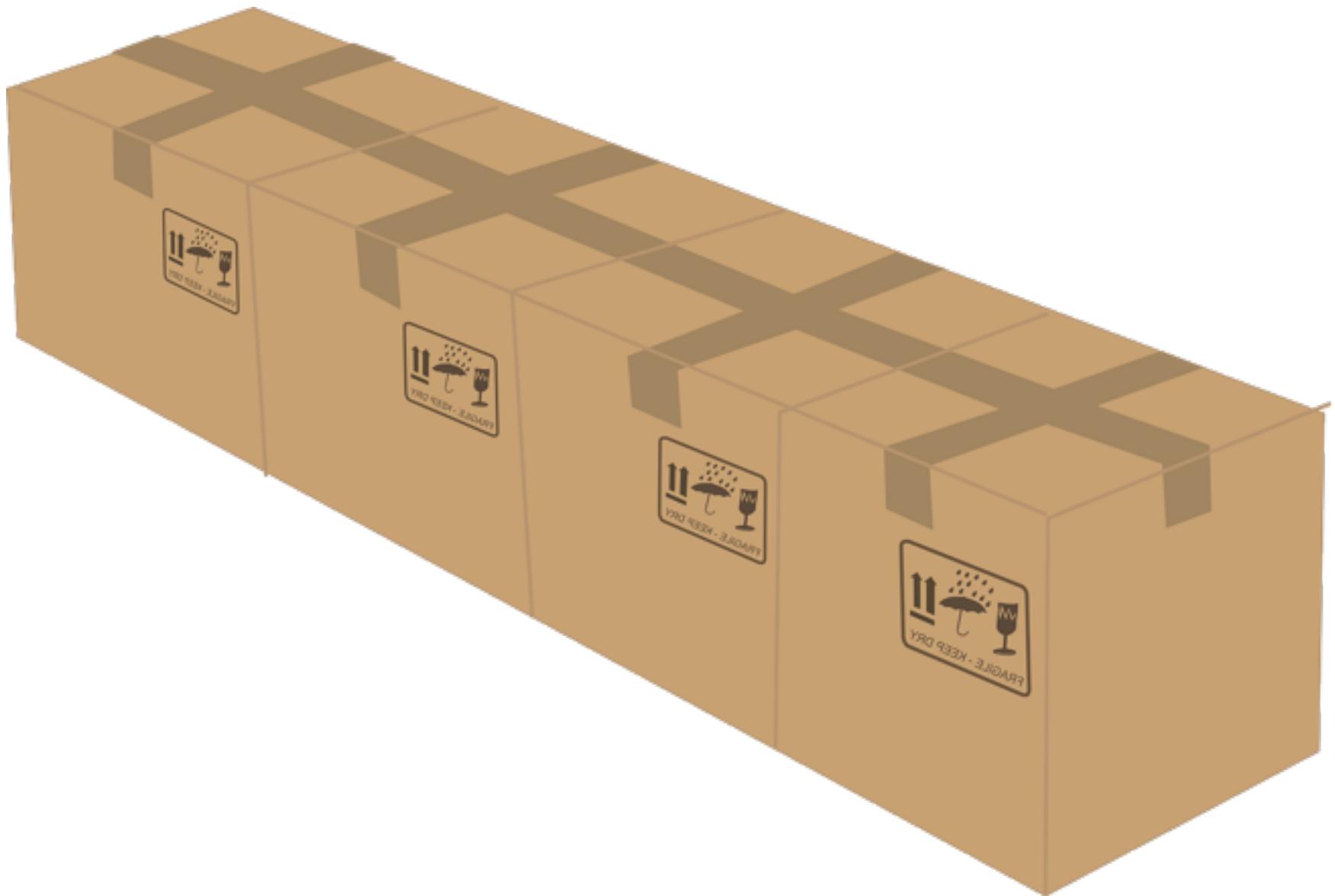
Algorithm, in polynomial time
gives packing s.t.
 $\text{Value}(\text{Output}) <$

$$\text{OPT}(1 + O(\epsilon)) + 1$$

Bin packing, linear programming and rounding



Bin packing, linear programming and rounding



Techniques

Adaptive rounding of input

Bin packing variants

Two-dimensional,
three-dimensional

Online

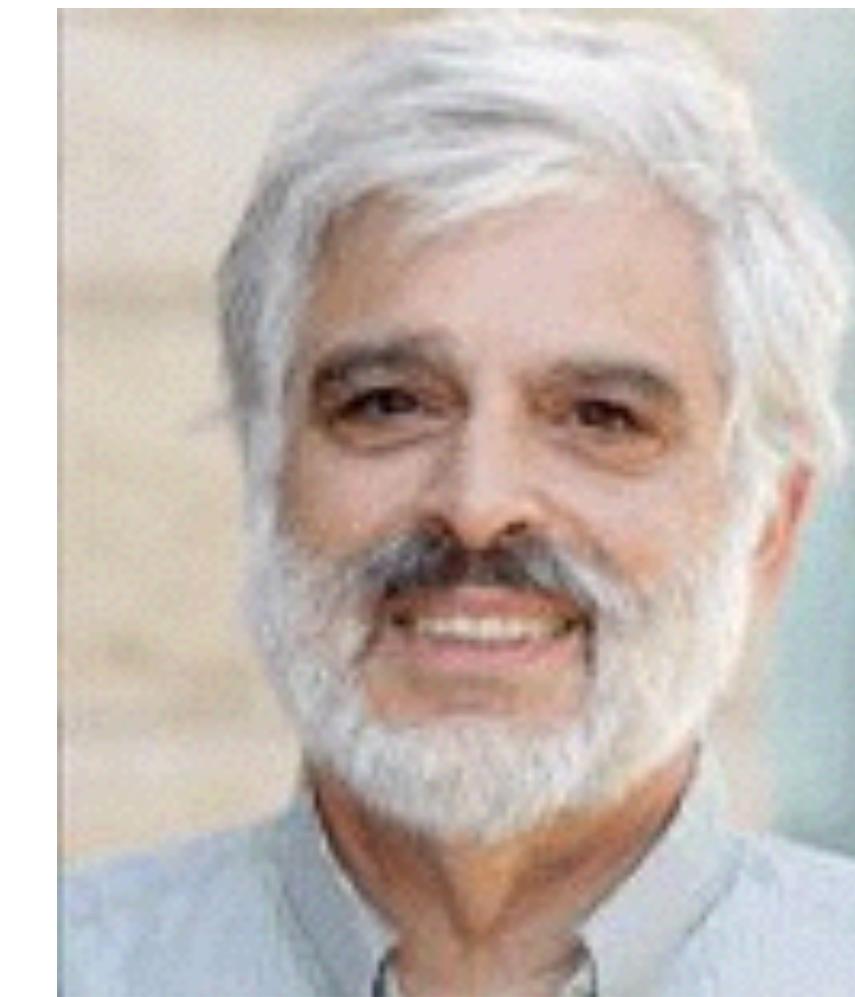


**Karp (1972)
NP-complete**

The pioneers



**David S. Johnson
Constant factor
approximation algorithms
Jeffrey D. Ullman**





Richard Karp



Narendra Karmarkar

Average case analysis



David S. Johnson



L. McGeoch

...



Wenceslas Fernandez de la Vega

$\text{OPT}(1 + O(\epsilon)) + 1$



George S. Lueker



Narendra Karmarkar

Richard Karp

$\text{OPT} + O(\log^2 \text{OPT})$



Asymptotic approximation schemes

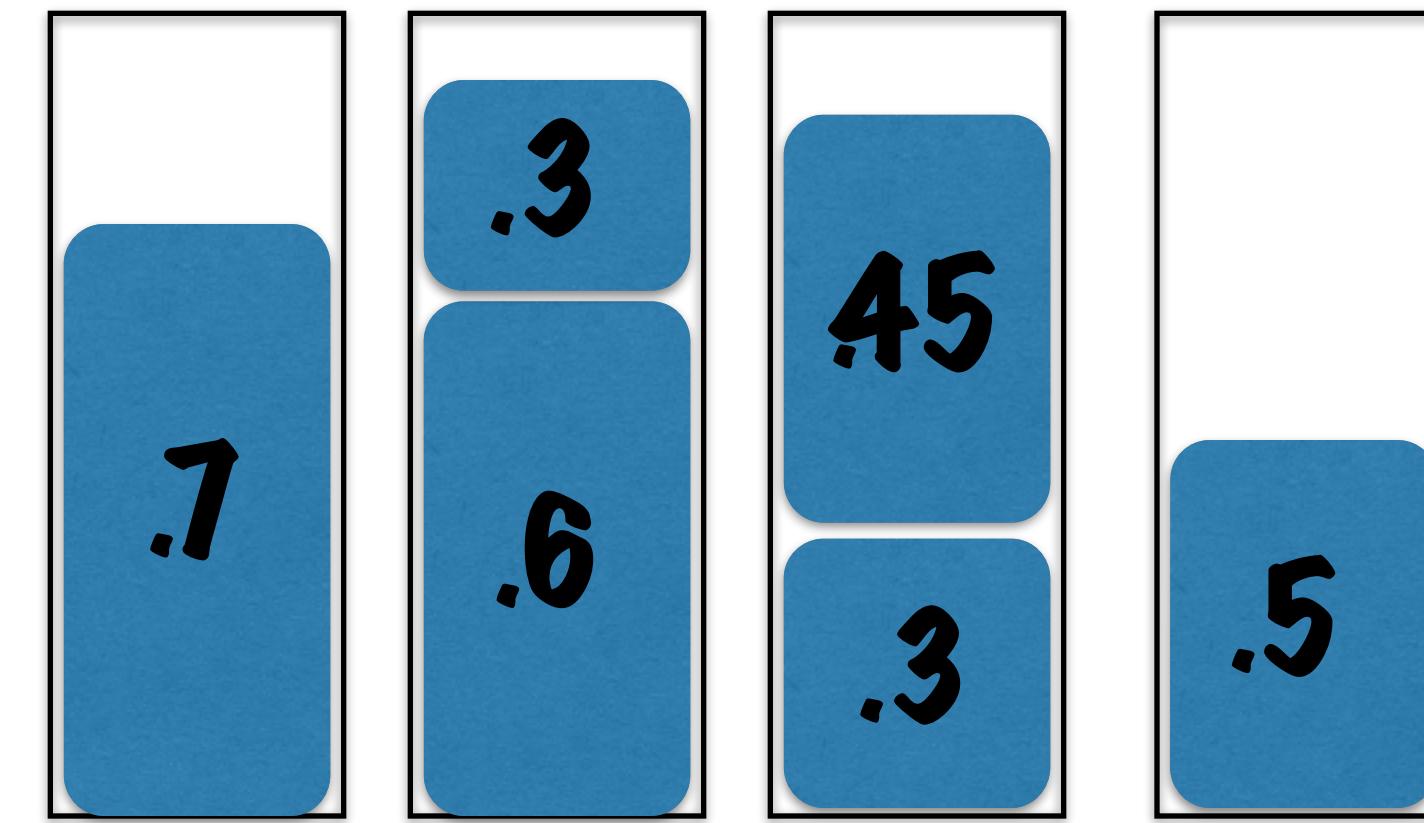
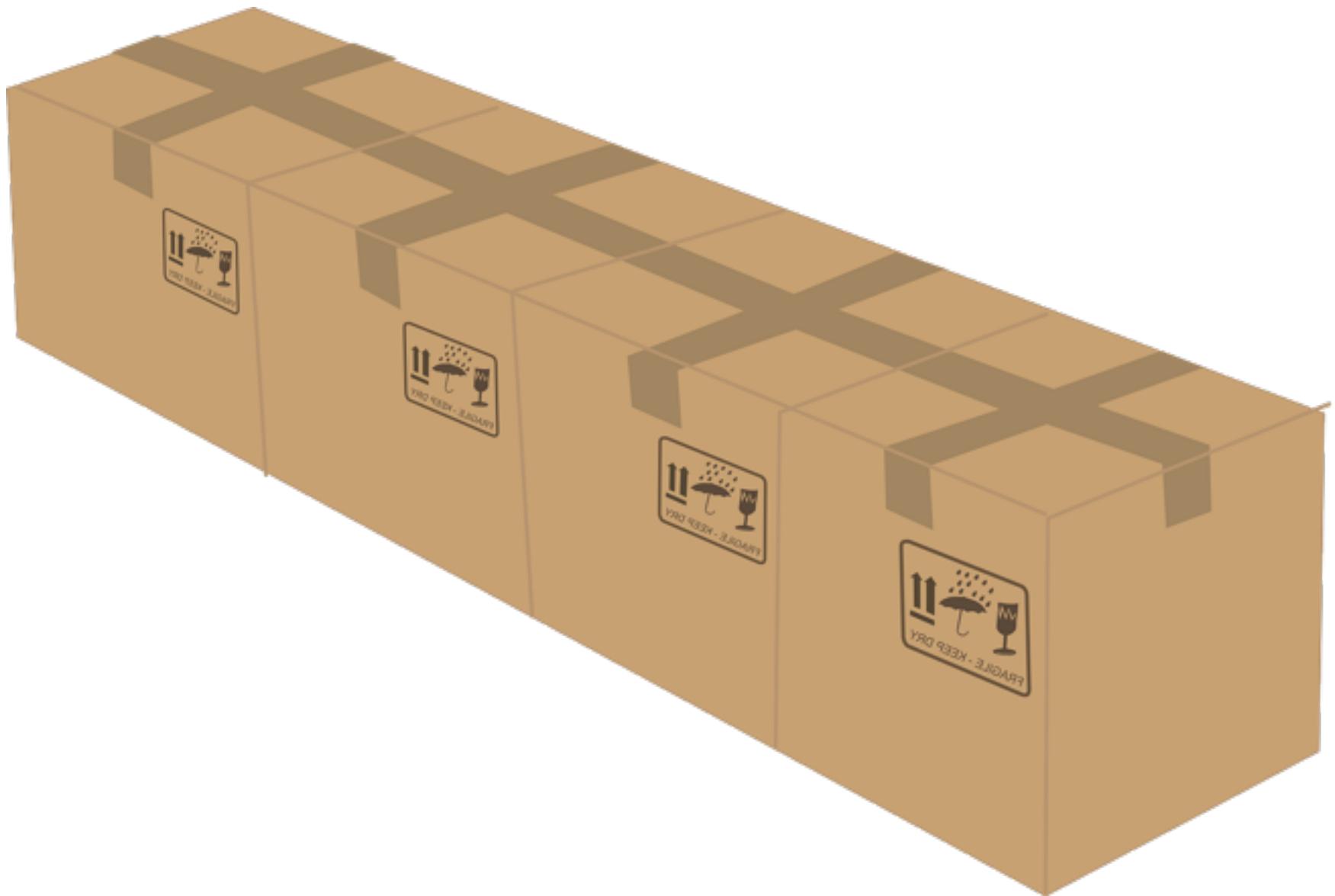


**Rebecca Hoberg
Thomas Rothvoss**

OPT + O(log OPT)

Asymptotic approximation schemes

Bin packing, linear programming and rounding





([https://accounts.coursera.org/i/zendesk/courserahelp?
return_to=https://learner.coursera.help/hc](https://accounts.coursera.org/i/zendesk/courserahelp?return_to=https://learner.coursera.help/hc))

Congratulations! [View Final Grade](#)

 You've completed the course. (/learn/approximation-algorithms-part-1/home/welcome)

Bin Packing, Linear Programming and Rounding



Claire Mathieu

This module shows the sophistication of rounding by using a clever variant for another basic problem: bin packing. (This is a more advanced module.)

The Next-Fit algorithm

 Lecture: Next Fit 13 min

(/learn/approximation-algorithms-part-1/lecture/vAkWL/lecture-next-fit)

 Slides

(/learn/approximation-algorithms-part-1/supplement/lLoqL/slides)

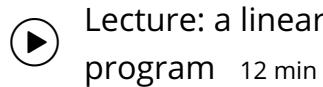
[Help Center](#)

 Practice Quiz: Quiz

1 2 questions

(/learn/approximation-algorithms-part-1/quiz/W3Mfd/quiz-1)

A linear program for Bin-Packing



Lecture: a linear program 12 min

(/learn/approximation-algorithms-part-1/lecture/eCdte/lecture-a-linear-program)



(/learn/approximation-algorithms-part-1/supplement/Qc9k0/slides)



Practice Quiz: Quiz

2 3 questions

(/learn/approximation-algorithms-part-1/quiz/Cjz8N/quiz-2)

Small items



Lecture: small items 7 min

(/learn/approximation-algorithms-part-1/lecture/wlQwr/lecture-small-items)



(/learn/approximation-algorithms-part-1/supplement/hkQHW/slides)

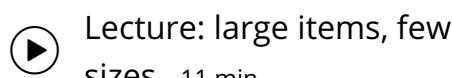


Practice Quiz: Quiz

3 1 question

(/learn/approximation-algorithms-part-1/quiz/z0EiW/quiz-3)

Large items, few sizes



Lecture: large items, few sizes 11 min

(/learn/approximation-algorithms-part-1/lecture/8kPiW/lecture-large-items-few-sizes)

 Slides

(/learn/approximation-algorithms-part-1/supplement/Mc2Cm/slides)

 Practice Quiz: Quiz

4 3 questions

(/learn/approximation-algorithms-part-1/quiz/OvkLO/quiz-4)

Large items, many sizes

 Slides

(/learn/approximation-algorithms-part-1/supplement/u9dx0/slides)

 Practice Quiz: Quiz

5 2 questions

(/learn/approximation-algorithms-part-1/quiz/Br4wT/quiz-5)

 Large items, many sizes 9 min

(/learn/approximation-algorithms-part-1/lecture/lYSvd/large-items-many-sizes)

Large items, analysis

 Lecture: large items analysis 8 min

(/learn/approximation-algorithms-part-1/lecture/G3ky3/lecture-large-items-analysis)

 Slides

(/learn/approximation-algorithms-part-1/supplement/z8BN0/slides)

 Practice Quiz: Quiz

6 3 questions

(/learn/approximation-algorithms-part-1/quiz/MH1OR/quiz-6)

General algorithm



Lecture: general
algorithm 8 min

(/learn/approximation-algorithms-part-
1/lecture/ubpXD/lecture-general-algorithm)



Slides

(/learn/approximation-algorithms-part-
1/supplement/jofjY/slides)



Practice Quiz: Quiz

7 3 questions

(/learn/approximation-algorithms-part-
1/quiz/td3Js/quiz-7)

Bin packing: conclusion



Lecture: conclusion 7 min

(/learn/approximation-algorithms-part-
1/lecture/2P5uO/lecture-conclusion)



Slides

(/learn/approximation-algorithms-part-
1/supplement/PQgAk/slides)

Exercises



Practice Exercises

(/learn/approximation-algorithms-part-
1/supplement/UfsvU/practice-exercises)

Assignment: Peer

✓ Assignment: Bin-
Packing 30 min

(/learn/approximation-algorithms-part-
1/peer/0OOH0/peer-assignment-bin-packing)

Review Classmates: Peer

✓ Assignment: Bin-

Packing 30 min

(/learn/approximation-algorithms-part-1/peer/0OOH0/peer-assignment-bin-packing/give-feedback)

- All slides together in one file

(/learn/approximation-algorithms-part-1/supplement/lxP2B/all-slides-together-in-one-file)

[Course Home \(/learn/approximation-algorithms-part-1/home/welcome\)](#) ➤ [Week 3 \(/learn/appro...](#)

Review Classmates: Peer Assignment: Bin-Packing

Review by December 30, 11:59 PM PT

Reviews 3 left to complete

Bin packing



by Xinyue Liu

December 28, 2015

like Flag this submission

Consider the instance of Bin-Packing given by fourteen items of size 6 and eight items of size 10. The bins have size 30. Recall that a feasible packing for a bin is called a *configuration* if the sum of the items considered is not greater than 30.

1. How many different configurations is possible to construct for the instance above?

In the sequel assume the following *sparseness* property: it is possible to find a solution for the configuration linear program that has at most t nonzero entries, where t is the number of different sizes. The algorithm below provides a solution for the bin-packing problem using the configuration linear program:

Algorithm: Rounding(x)

Step 1. Find a sparse solution x of the configuration linear program in instance I .

Step 2. Open $\lfloor x_C \rfloor$ bins with configuration C .

Step 3. If some item has not been packed, consider the instance \tilde{I} of all those remaining items.

Step 4. For \tilde{I} , let P_1 be the packing given by opening a new bin for each C such that x_C is fractional, and let P_2 be the packing given by Next-Fit. The output is the best among these two.

In the instance described above the number of different sizes is equal to $t = 2$. In fact, let C_1 the configuration given by five items of size 6 and C_2 the configuration given by three items of size 10.

Help Center

2. Consider $x_{C_1} = 14/5$, $x_{C_2} = 8/3$ and $x_C = 0$ for every other configuration C . Prove that it is feasible for the configuration linear program of the instance described above.
3. Is this solution sparse?
4. Consider the solution shown above and go to Step 2 of the algorithm. How many bins are opened in configurations C_1 and C_2 respectively?
5. In the instance \tilde{I} of the remaining items at Step 3, how many items in \tilde{I} have size 6 and 10, respectively?
6. In the packing P_1 at Step 4, how many bins are opened in configuration C_1 and C_2 respectively?
7. Run Next-Fit for \tilde{I} . How many bins are opened?
8. Conclude that the algorithm above returns the optimal number of bins for the instance.

[Bin_packing_\(https://s3.amazonaws.com/coursera-uploads/peer-review/bnQLDSclEeWbiBJCM9ziNQ/2857e258bc2535131e0844314d7da0f3/h3.pdf\)](https://s3.amazonaws.com/coursera-uploads/peer-review/bnQLDSclEeWbiBJCM9ziNQ/2857e258bc2535131e0844314d7da0f3/h3.pdf)

The answer to 1. has the following (or similar) form:

The number of possible configurations is **13**. Let x, y be the number of items of size 6 and 10, respectively, in a configuration. The capacity of the bins is 30, and then, the number of possible configurations corresponds to the non-negative integer solutions of $6x + 10y \leq 30$, $x \geq 0$ and $y \geq 0$. To count the number of solutions of this system one can fix the value of y and then find the possible values for x .

- If $y = 0$, then $x \in \{0, 1, 2, 3, 4, 5\}$.
- If $y = 1$, then $x \in \{0, 1, 2, 3\}$.
- If $y = 2$ then $x \in \{0, 1\}$.
- If $y = 3$ then $x = 0$.

- 4 pts
Yes
- 0 pts
No

The answer to 2. has the following (or similar) form:

The non-negativity constraints are satisfied. We first check the constraint associated to size 6: $\sum_C a_{6,C} x_C = 5 \cdot x_{C_1} = 5 \cdot 14/5 = 14 = n_6$. Now we check the constraint associated to size 10: $\sum_C a_{10,C} x_C = 3 \cdot x_{C_2} = 3 \cdot 8/3 = 8 = n_{10}$.

- 3 pts
Yes
- 0 pts
No

The answer to 3. has the following (or similar) form:

The solution is sparse since it takes positive values in $t = 2$ configurations.

- 1 pt
Yes
- 0 pts
No

The answer to 4. has the following (or similar) form:

Step 2 opens $\lfloor x_{C_1} \rfloor = \lfloor 14/5 \rfloor = 2$ in configuration C_1 , and $\lfloor x_{C_2} \rfloor = \lfloor 8/3 \rfloor = 2$ in configuration C_2 .

- 2 pts
Yes
- 0 pts
No

The answer to 5. has the following (or similar) form:

In \tilde{I} there are four items of size 6, and two items of size 10.

- 2 pts
Yes
- 0 pts
No

The answer to 6. has the following (or similar) form:

It is opened one extra bin in configuration C_1 and one extra in configuration C_2 . This extra two bins allocate at most five items of size 6, and three of size 10. Then it induces a feasible packing since in \tilde{I} it remains four of size 6 and two of size 10 to be packed.

- 4 pts
Yes
- 0 pts
No

The answer to 7. has the following (or similar) form:

No matter how we sort the items in \tilde{I} , the number of bins opened is exactly 2. This follows since it is always possible to pack at least three (and at most four) items of the six in \tilde{I} in a bin of size 30.

- 4 pts
Yes
- 0 pts
No

The answer to 8. has the following (or similar) form:

The total volume of the items in the instance is $14 \cdot 6 + 8 \cdot 10 = 164$. Then, a lower bound on the number of bins of capacity 30 used in any feasible solution is $\lceil 164/30 \rceil = 6$. The optimality follows since the algorithm above matches the lower bound.

- 5 pts
Yes
- 0 pts
No

[Submit Review](#)

Comments

Visible to classmates



share your thoughts...

◀ (/learn/approximation-algorithms-part-1/supplement/UfsVU/practice-exercises)
https://accounts.coursera.org/zendesk/courserahelp?return_to=https://learner.coursera.help/hc/1/supplement/UfsVU/practice-exercises

▶ (/learn/approximation-algorithms-part-1/supplement/lxP2B/all-slides-together-in-one-file)
<https://www.coursera.org/learn/approximation-algorithms-part-1/peer/0OOH0/peer-assignment-bin-packing/review-next>

PRACTICE EXERCISES: BIN-PACKING

THE SOLUTIONS WILL BE AVAILABLE IN 1-3 WEEKS

Hardness and Inapproximability. The objective of this exercise is to prove that the decision version of Bin-Packung is NP-complete and an inapproximability result. In the decision version of Bin-Packung we are given a set of items, a positive integer number k , and the goal is to decide whether it is possible to pack the items in those k bins of capacity one. The proof comes from a reduction from the NP-complete *Partition problem*,

Partition: Given a sequence a_1, a_2, \dots, a_n of non-negative integers, decide whether there is a subset $S \subseteq [n]$ such that

$$\sum_{j \in S} a_j = \sum_{j \notin S} a_j.$$

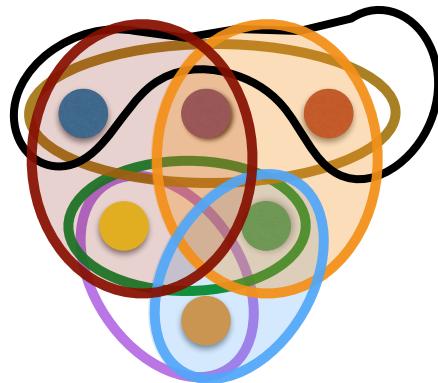
Theorem 1. *The decision version of Bin-Packung is NP-complete.*

Theorem 2. *There is no α -approximation algorithm for Bin-Packung with $\alpha < 3/2$, unless $P=NP$.*

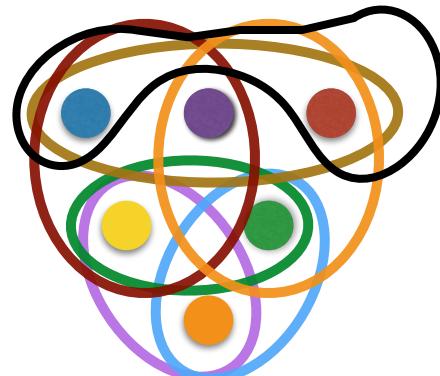
- (1) Given an instance I_P to Partition, the following instance I_B for Bin-Packung is constructed: there are two bins and for each number a_j there is an item j of size $s_j = 2a_j/A$, where $A = \sum_{j \in [n]} a_j$.
 - (a) Prove that if I_P is a YES instance for Partition, then I_B is a YES instance for Bin-Packung.
 - (b) Prove that if I_B is a YES instance for Bin-Packung with two bins, then I_P is a YES instance for Partition.
 - (c) Conclude Theorem 1.
- (2) Suppose that exists $\varepsilon > 0$ and an algorithm that is a $(3/2 - \varepsilon)$ -approximation for Bin-Packung. In particular, this algorithm can be run over those instances where the optimal packing uses two bins.
 - (a) Given an instance I for the partition problem, construct the same instance for Bin-Packung as before, and use $(3/2 - \varepsilon)$ -approximation to decide whether I is a YES instance.
 - (b) Conclude Theorem 2 using the fact that the algorithm runs in polynomial time, and that Partition is an NP-complete problem.

FFD algorithm. Given a Bin-Packung instance with n items and sizes s_1, \dots, s_n , we sort them according to non-increasing order. Consider the *First-Fit decreasing* algorithm: a bin $j = 1$ is opened and if item 1 fits in this bin then it is packed in it. We continue with item 2, if it fits into the bin $j = 1$ then it is packed there, and if not then a new bin $j = 2$ is opened. In general, given an item i , it is packed into the first bin where it is possible to pack, and if not then a new bin is opened. Prove that this algorithm returns a packing using at most $3/2 \cdot \text{opt} + 1$ bins.

Set cover, linear programming and randomized rounding



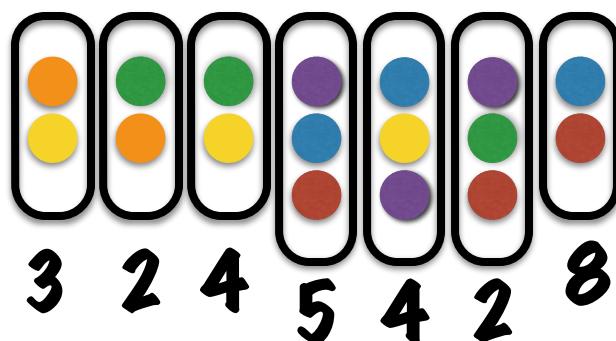
The set cover problem



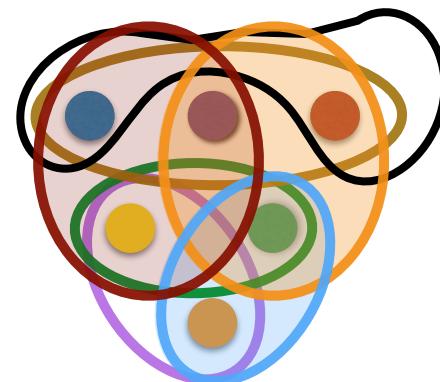
Elements



Subsets with costs



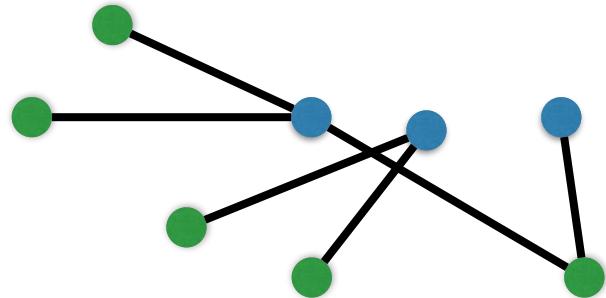
Choose subsets
Cover elements
at min cost



$$\text{Cost} = 4 + 2 + 2 = 8$$

Does this ring a bell?

Vertex cover



Cover all edges
with the fewest
vertices

edges = elements
vertices = sets

Integer program
for
Set cover

Variable for subset S:

$$x_S = 1$$

iff S in cover

Constraint for element i:

$$\sum_{S:e \in S} x_S \geq 1$$

Objective:

$$\min \sum_S c_S x_S$$

Linear programming relaxation

$$\min \sum_S c_S x_S$$

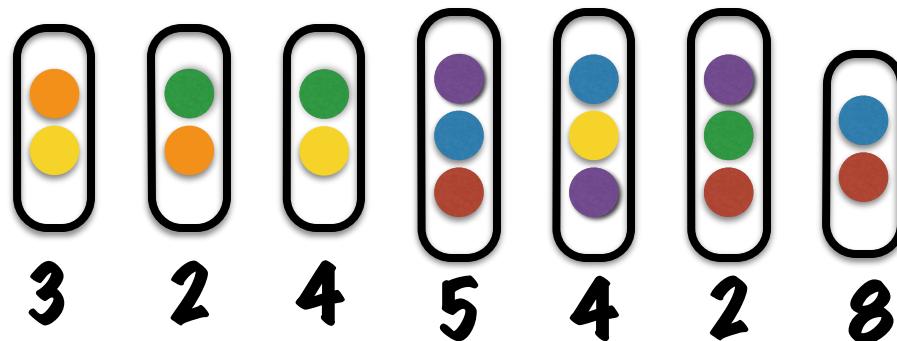
such that

$$\begin{cases} \sum_{S: e \in S} x_S \geq 1 & \forall e \\ 0 \leq x_S \leq 1 & \forall S \end{cases}$$

Rounding

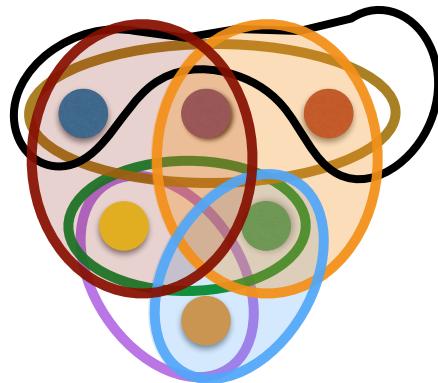
Like vertex cover:
round to 1 iff $x_u \geq 1/2$

Fails

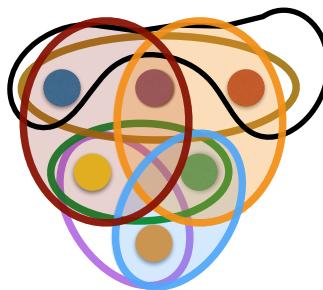


$$x_S : \frac{1}{3}, \frac{2}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{3}, \frac{1}{2}, \frac{1}{3}$$

Set cover, linear programming and randomized rounding



Set cover, linear programming and randomized rounding



Linear programming relaxation

$$\min \sum_S c_S x_S$$

such that

$$\begin{cases} \sum_{S:e \in S} x_S \geq 1 & \forall e \\ 0 \leq x_S \leq 1 & \forall S \end{cases}$$

How do we round the LP solution?

Randomized Rounding: An algorithm

$x_i = .9$: should probably go to 1

$x_i = .1$: should probably go to 0

Cannot fix a threshold

Idea: randomized rounding

$x_i = .8 \implies$ round to 1
w.p. 80%

New rounding algorithm

For each set S
with probability x_S
put S in the cover

Analysis

**Is it efficient?
Is the output a cover?
How good is it?**

Analysis

**Is it efficient?
Yes**

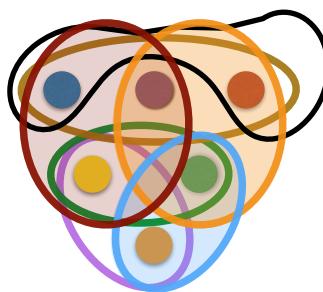
Analysis

**Is the output a cover?
Maybe, maybe not**

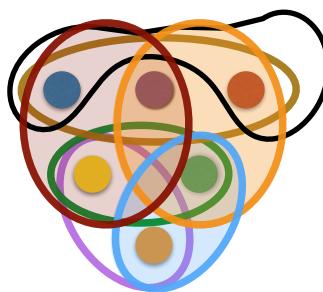
Analysis

How good is it?
It depends...

Set cover, linear programming and randomized rounding



Set cover, linear programming and randomized rounding



How good is it?
It depends...

$$\text{Value}(\text{Output}) = \sum_{S \text{ in cover}} c_S$$

How good is it *on average*?

$$E[\sum_S 1(S \text{ in cover})c_S] = ?$$

Linearity of expectation

$$E[A + B] = E[A] + E[B]$$

$$\begin{aligned} E\left[\sum_S 1(S \text{ in cover})c_S\right] &= \\ \sum_S E[1(S \text{ in cover})c_S] \end{aligned}$$

$$E[\lambda X] = \lambda E[X]$$

$$\begin{aligned} \sum_S E[1(S \text{ in cover}) c_S] &= \\ \sum_S E[1(S \text{ in cover})] c_S \end{aligned}$$

$$\begin{aligned} E[1(S \text{ in } \text{cover})] &= \\ \Pr[S \text{ in } \text{cover})] \end{aligned}$$

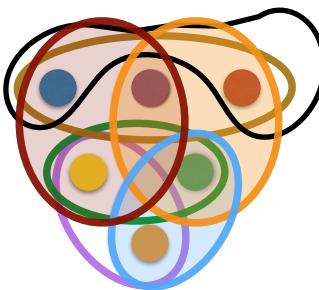
$$\Pr[\mathbf{S} \text{ in cover}] = \mathbf{x_S}$$

Together

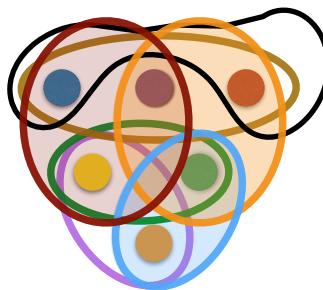
$$E[Value(Output)] = \sum_S x_S c_S$$

Value of the linear program!

Set cover, linear programming and randomized rounding



Set cover, linear programming and randomized rounding



**Is the output a cover?
Maybe, maybe not**

Number of elements covered:

$$\sum_e 1(e \text{ covered})$$

On average:

$$\sum_e \Pr[e \text{ covered}]$$

**Consider an element e .
With what probability
is it covered by output?**

$\Pr[\mathbf{e \text{ covered}}] =$
 $\Pr[\text{there exists } S \text{ in output:}$
 $\mathbf{e \in S}]$

$\Pr[\text{there exists } S \text{ in output:}$
 $e \in S] =$
 $1 - \Pr[\text{for all } S \text{ containing } e:$
 $S \text{ not in output}]$

Independence

If independence:

$$\Pr[A \text{ and } B] = \Pr[A] \times \Pr[B]$$

$\Pr[\text{for all } S \text{ containing } e:$

$S \text{ not in output}] =$

$$\prod_{S:e \in S} \Pr[S \text{ not in output}]$$

$$\Pr[S \text{ not in output}] = 1 - x_S$$

Together

$$\Pr[\text{e covered}] = 1 - \prod_{S:e \in S} (1 - x_S)$$

Algebra

$$X = e^{\ln X}$$

$$\ln(XY) = \ln X + \ln Y$$

$$\prod_{S:e \in S} (1 - x_S) = e^{\sum_{S:e \in S} \ln(1 - x_S)}$$

Algebra

$$\ln(1 - X) \leq -X$$

$$e^{\sum_{S:e \in S} \ln(1-x_S)} \leq e^{-\sum_{S:e \in S} x_S}$$

Use LP constraint

$$\sum_{S:e \in S} x_S \geq 1$$

$$e^{-\sum_{S:e \in S} x_S} \leq e^{-1}$$

Combining

$$\Pr[e \text{ covered}] \geq 1 - 1/e$$

**Average number of
elements covered:**

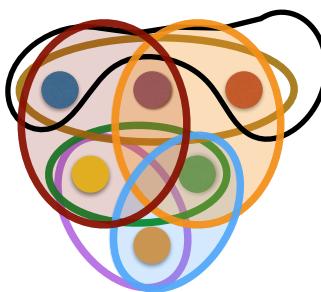
$$\#(\text{elements})(1 - 1/e)$$

$$1 - 1/e = 0.63\dots$$

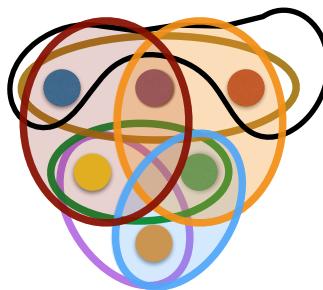
Recap

**Randomized rounding gives
collection of sets
with average cost
at most OPT
and covering on average
63% of the elements.**

Set cover, linear programming and randomized rounding



Set cover, linear programming and randomized rounding



Getting a set cover

Idea: repeat!

Randomized rounding algorithm

$n = \# \text{elements}$

Repeat $\ln(n) + 3$ times

For each S

Put S in cover w.pr. $x(S)$
(if not there already)

Note: $e^3 = 20.0\dots$

Cost

In expectation:
at most
 $(\ln(n) + 3)OPT$

Correctness

$$\Pr[\text{cover}] = 1 - \Pr[\text{not cover}]$$

$$\begin{aligned}\Pr[\text{not cover}] &= \\ \Pr[\exists \text{ element not covered}] &\leq \\ \sum_e \Pr[e \text{ not covered}] &\end{aligned}$$

**For one element e
and for one iteration**

$$\Pr[e \text{ not covered}] < 1/e$$

**For one element e
and for all iterations together**

$$\Pr[e \text{ not covered}] < (1/e)^{\ln(n)+3} = \frac{1}{e^3 n}$$

$$\sum_e \Pr[e \text{ not covered}] < n \frac{1}{e^3 n} = \frac{1}{e^3} < 0.05$$

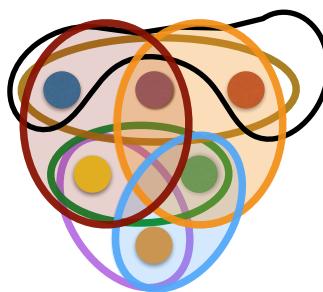
So:

$$\Pr[\text{cover}] > 0.95$$

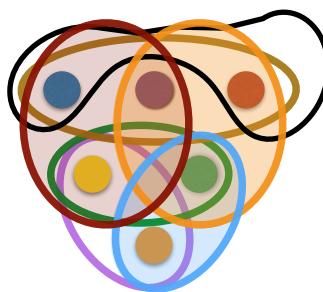
Result

**Iterated randomized rounding gives
collection of sets
that is a set cover with
probability 95% and
with average cost
at most $(\ln(n)+3)$ OPT.**

Set cover, linear programming and randomized rounding



Set cover, linear programming and randomized rounding



Result so far

Iterated randomized rounding gives collection of sets

that is a set cover with probability 95%
and with average cost at most $(\ln(n)+3)$ OPT.

Not guaranteed!

Not guaranteed!

**Q: What if you want the output
to always be a set cover?**

Desired result:
algorithm that gives
collection of sets
that **is** a set cover
and with **average**
cost at most $O(\ln(n))$ OPT.

Guaranteed!

Obvious solution:
Replace
“repeat $\ln(n)+3$ times”
by
“repeat until you have a set cover.”

Equivalent algorithms

Repeat
For each S
Put S in cover w.pr. $x(S)$
(if not there yet)
Until you have a set cover

Repeat
Choose S w.pr. $x_S / \sum_{S'} x(S')$
Put S in cover
(if not there yet)
Until you have a set cover

Sample-and-iterate algorithm

Repeat

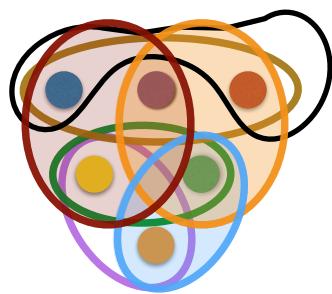
Choose S w.pr. $x_S / \sum_{S'} x(S')$

Put S in cover

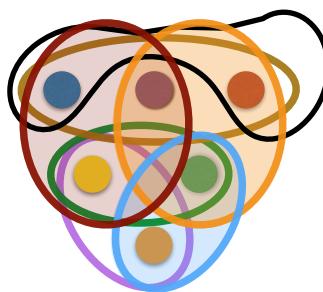
(if not there yet)

Until you have a set cover

Set cover, linear programming and randomized rounding



Set cover, linear programming and randomized rounding



Sample-and-iterate algorithm

Repeat

Choose S w.pr. $x_S / \sum_{S'} x(S')$

Put S in cover

(if not there yet)

Until you have a set cover

Repeat

Choose S w.pr. $x_S / \sum_{S'} x(S')$

Put S in cover

(if not there yet)

Until you have a set cover

Analysis

1. Expected cost of output

$T = \text{\#iterations (stopping time)}$

$C_t = \text{cost of set chosen at iteration } t$

Cost of output: $\sum_{t=1}^T C_t$

Repeat

Choose S w.pr. $x_S / \sum_{S'} x(S')$

Put S in cover

(if not there yet)

Until you have a set cover

Cost of output: $\sum_{t=1}^T C_t$

Expected cost of output: $E[\sum_{t=1}^T C_t]$

NB: cannot exchange $E[]$ and summation here!

Wald's equation

T stopping time

X_t random variable

If X_t bounded from above:

$$\text{then } E\left[\sum_{t \leq T} X_t\right] \leq \mu E[T]$$

NB: can now exchange E[] and summation!

Expected cost of set chosen at iteration t:

$$E[C_t] = \sum_S \Pr[S \text{ chosen}] c_S = \frac{\sum_S c_S x_S}{\sum_S x_S} = \mu$$

Expected cost of output: $E[T]\mu = E[T] \frac{\sum_S c_S x_S}{\sum_S x_S}$

Next problem: What is $E[T]$?

Repeat

Choose S w.pr. $x_S / \sum_{S'} x(S')$

Put S in cover

(if not there yet)

Until you have a set cover

More
notations

2. Expected number of iterations

$n_t = \#\text{elts not yet covered after } t \text{ iterations}$

$$n_0 = n, n_T = 0, n_{T-1} \geq 1$$

Wald's equation for dependent decrements

Consider a random decreasing sequence

$$n_0, n_1, \dots, n_T,$$

where T is a stopping time. If

$$E[n_t - n_{t+1} | n_t] \geq f(n_t),$$

where f is an increasing function, then

$$E[T] \leq 1 + E\left[\int_{n_{T-1}}^{n_0} \frac{1}{f(z)} dz\right].$$

To bound $E[T]$, analyze
#elts not yet covered after t iterations

Analyze n_t

Analyze $E[n_t - n_{t+1} | n_t]$: elts covered in next iteration

Fix an element e among the n_t

$\Pr[e \text{ covered in next iteration}] =$

$\Pr[S \text{ chosen contains } e] =$

$$\frac{\sum_{S:e \in S} x_S}{\sum_{S'} x_{S'}} \geq \\ 1 / \sum_{S'} x_{S'}$$

Sum over e

$$E[n_t - n_{t+1} | n_t] \geq f(n_t)$$

$$E[n_t - n_{t+1} | n_t] \geq n_t / \sum_{S'} x_{S'}$$

Wald's equation for dependent decrements

Consider a random decreasing sequence

$$n_0, n_1, \dots, n_T,$$

where T is a stopping time. If

$$E[n_t - n_{t+1} | n_t] \geq f(n_t),$$

where f is an increasing function, then

$$E[T] \leq 1 + E\left[\int_{n_{T-1}}^{n_0} \frac{1}{f(z)} dz\right].$$

Application:

Stopping time $E[T] \leq 1 + \int_1^n \frac{\sum_S x_S}{z} dz = 1 + \sum_S x_S \ln(n)$

Together

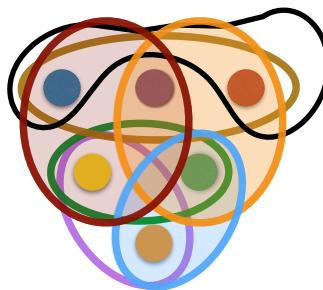
Expected cost of output:

$$(1 + (\sum_S x_S) \ln(n)) \frac{\sum_S c_S x_S}{\sum_S x_S} \leq (1 + \ln(n)) \text{ OPT}$$

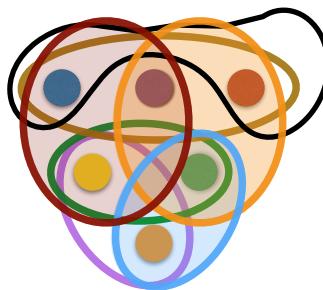
Result:

the algorithm gives
a collection of sets that is a set cover
with average cost at most $(1 + \ln(n)) \text{ OPT}$.

Set cover, linear programming and randomized rounding



Set cover, linear programming and randomized rounding



Result

The sample-and-iterate algorithm gives a collection of sets that is a set cover with average cost at most $(1 + \ln(n)) \text{ OPT}$.

A more efficient algorithm

**Linear programming takes
polynomial time
but
is often slower than
combinatorial algorithms**

Greedy

Repeat

Choose S maximizing $\#(\text{new elts covered})/c_S$

Put S in cover

Until you have a set cover

Result:
Greedy also gives
a collection of sets
that **is** a set cover
and with
cost at most $(1 + \ln(n)) \text{ OPT.}$

Can we do better?

No:

**It is NP-hard to obtain (in polynomial time) a
better-than- $\ln(n)$ approximation
for set cover**



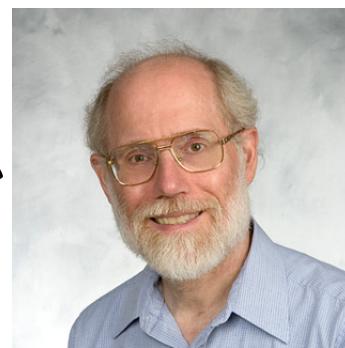
**Uri
Feige**



**Vašek
Chvátal**



**László
Lovász**



**David
Johnson**

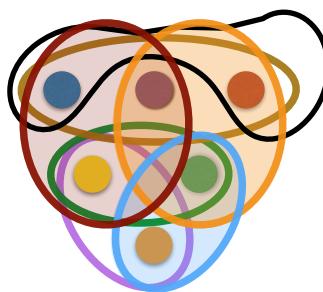


**Neal
Young**

What have we learned?

- Famous problem: set cover
- Concept: Randomization
- Algorithmic technique: Randomized rounding
- Analysis tool: Linearity of expectation
- Laying out an analysis: slow & steady, orderly - like hiking up a mountain

Set cover, linear programming and randomized rounding



Course Home (/learn/approximation-algorithms-part-1/home/welcome) > Week 4 (/learn/approximation-algorithms-part-1/home/week/4) > Ex

Review Classmates: Peer Assig Set Cover

Review by January 6, 11:59 PM PT

Reviews 3 left to complete

Assignment 4



by M W

January 5, 2016

like Flag this submission

Consider the following version of Wald's equation: Given a random sequence n_0, n_1, \dots, n_T , where T is a stopping time and $n_i \geq 1/2$ for $i \leq T$. If $E[n_t - n_{t+1} | n_t] \geq f(n_t)$ for all t , where $f(\cdot)$ is an increasing and differentiable function, then

$$E[T] \leq \frac{1}{2f(1/2)} + E\left[\int_{n_{T-1}}^{n_0} \frac{1}{f(z)} dz\right].$$

Let say you go to a casino with $n_0 = \$x$ and you start with play the following game. In every round t you bet all your money n_t you have at that round.

With probability $1/2$ you have $n_{t+1} = (9/8) \cdot n_t$. With probability $1/4$ nothing happens $n_{t+1} = n_t$ and with probability $1/4$ you lose $1/2$ of your money: $n_{t+1} = n_t/2$. You leave when you have less than $\$1$.

Assume that money is arbitrarily divisible, so you can have for example $\$0.00002$.

We are interested in how long it takes until you leave the casino. Execute the following steps

1. Give a definition for n_t for $t \in \mathbb{N}$.
2. Calculate $E[n_t - n_{t+1} | n_t]$.
3. Calculate $f(n_t)$.
4. Bound $E[T]$ using Wald's equation.

HW4_ApAlg.pdf (https://s3.amazonaws.com/coursera-uploads/peer-review/bnQLDSclEeWbiBJCM9ziNQ/ccd67a49af09eb0ee88c5582936c2b08/HW4_ApAlg.pdf)

One possible solution:

1. Let n_t be the money of the gambler at round t .

We have

$$E[n_{t+1} | n_t] = (9/8)n_t \cdot 1/2 + n_t \cdot 1/4 + n_t/2 \cdot 1/4 = \frac{15}{16}n_t. \text{ (For marking: Note that } 15/16 = 0.9375\text{)}$$

Thus, we derive

$$2. E[n_t - n_{t+1} | n_t] = n_t - E[n_{t+1} | n_t] = n_t - \frac{15}{16}n_t = (1 - \frac{15}{16})n_t. \text{ (For marking: Note that } 1-15/16 = 1/16=0.0625\text{)}$$

$$3. \text{ Hence } f(n_t) = (1 - \frac{15}{16})n_t$$

$$\text{and } E\left[\int_{n_{T-1}}^{n_0} \frac{1}{f(z)} dz\right] = (1 - \frac{15}{16})^{-1} (\log(n_0) - \log(n_{T-1})) \leq (1 - \frac{15}{16})^{-1} \cdot \log(x) = 16 \log(x).$$

$$4. \text{ Hence } E[T] \leq 16 + 16 \log(x).$$

1. The definition for $\$1$ matches the definition proposed by the solution.

- 2 pts
- Yes

- 0 pts
No

2. The conclusion $E[n_t - n_{t+1} | n_t] = \dots = (1 - \frac{15}{16})n_t$. is stated. (Or $= n_t/16$, or $= 0.0625n_t$.)

- 2 pts
Yes
 0 pts
No

3. The conclusion that $f(n_t) = (1 - \frac{15}{16})n_t$ or an equivalent version is stated.

- 2 pts
Yes
 0 pts
No

4. $E[T]$ is bounded by something which is at most $16 + 16\log(x)$.

- 2 pts
Yes
 0 pts
No

Is the answer presented in a clear and comprehensible way?

- 2 pts
Yes
 0 pts
No

You ask ten of your friends to give you one dollar (coin) each so that you can buy your favourite pizza.

When you reach the pizza place you realize that it's closed. So you

return the coins to your friends by giving each of them a random coin.

1. How likely is that each of them gets the coin they put back?
2. How many of them will get in expectation the coin they put back? (Try to prove this elegantly using techniques discussed in the course) Define the random variables you use.
3. Now suppose that you take one coin to buy a coffee somewhere else. You redistribute the others dollar coins giving each of your friends one coin chosen uniformly at random. How many of your friends will get in expectation the coin they put back?

1. the probability of getting the same coin back as the one put in is $1/10$.

2. if we define a random variable X_i for each coin i taking the values 0 or 1 such that $X_i = 1$ if coin i got back to friend i , and $X_i = 0$ otherwise, then we have $P(X_i = 1) = 1/10$ and $P(X_i = 0) = 9/10$. We also have

Let us define S as the total number of friend that got the same coin as they put in. Then

$$S = \sum_{i=1}^{10} X_i$$

therefore

$$E[S] = \sum_{i=1}^{10} E[X_i] = \sum_{i=1}^{10} 1/10 = 1$$



3. In this case, for each coin there is a $1/10$ probability that the coin is the one used to buy coffee, in which case it will not be returned to its owner. Therefore the $P(X_i = 0) = 1/10 + (1 - 1/10) \times 9/10 = 91/100$ and

$$E[S] = \sum_{i=1}^{10} E[X_i] = \sum_{i=1}^{10} 9/100 = 9/10$$

One possible solution:

$$1. \frac{1}{10} \cdot \frac{1}{9} \cdot \dots \cdot \frac{1}{2} \cdot \frac{1}{1} \approx 2.75573192 \cdot 10^{-7}$$

2. X_i is $0/1$ -variable which is 1 if person i gets their coin back.

$E[X_i] = P(X_i = 1)$. Thus, by linearity of expectation,

$$E[\sum_{i=1}^{10} X_i] = \sum_{i=1}^{10} E[X_i] = 1$$

3. W.l.o.g. assume the coin of person \$10\$ was used for your coffee.

Hence, we get $E[\sum_{i \leq 9} X_i] = \sum_i E[X_i] = 0.9$.

1. The question asked the probability that each one gets his coin back, not the probability that just one gets his coin back.

The value calculated for 1. is correct

- 2 pts
Yes
- 0 pts
No

A random variable was defined

- 1 pt
Yes
- 0 pts
No

The result of 2. is 1

- 2 pts
Yes
- 0 pts
No

The result of 3. is 0.9

- 3 pts
Yes
- 0 pts
No

Is the answer presented in a clear and comprehensible way?

- 2 pts
Yes
- 0 pts
No

[Submit Review](#)

Comments

Visible to classmates



share your thoughts...

◀ (/learn/approximation-algorithms-part-

(https://account.coursera.org/account/supplement/BbCuD/practice-exercise)=https://learner.coursera.org/account/supplement/BbCuD/all-slides-together-in-one-file)

▶ (/learn/approximation-algorithms-part-

Course Home (/learn/approximation-algorithms-part-1/home/welcome) > Week 4 (/learn/approximation-algorithms-part-1/home/week/4) > Ex

Assignment: Peer Assig Set Cover

Review classmates to see your grade

Your grade is ready, but you have not reviewed enough classmates yet. Review more classmates to see your grade.

[Review a Classmates' Work \(/learn/approximation-algorithms-part-1/peer/pLGal/peer-assig-set-cover/give-feedback\)](#)

Instructions (/learn/approximation-algorithms-part-1/peer/pLGal/peer-assig-set-cover)

My submission (/learn/approximation-algorithms-part-1/peer/pLGal/peer-assig-set-cover/submit)

Discussions (/learn/approximation-algorithms-part-1/peer/pLGal/peer-assig-set-cover/discussions)

Thank you for completing a peer reviewed assignment! Please take this survey to help us improve your experience.

(<https://www.surveymonkey.com/r/Z8KFSCC>)

c=assignmentId%3DlvYO5rREeWjygrdNyX3jw%405&c=courseId%3DbnQLDSclEeWbiBJCM9ziNQ&c=itemId%3DpLGal&c=submissionId%3DbgnTnLDNEeWoGg6u

Exercise 4 Solutions

January 2, 2016

Shareable Link (<https://www.coursera.org/learn/approximation-algorithms-part-1/peer/pLGal/peer-assig-set-cover/review/bgnTnLDNEeWoGg6uIZMPEw>)

Consider the following version of Wald's equation: Given a random sequence n_0, n_1, \dots, n_T , where T is a stopping time and $n_i \geq 1/2$ for $i \leq T$. If $E[n_t - n_{t+1} | n_t] \geq f(n_t)$ for all t , where $f(\cdot)$ is an increasing and differentiable function, then

$$E[T] \leq \frac{1}{2f(1/2)} + E\left[\int_{n_{T-1}}^{n_0} \frac{1}{f(z)} dz\right].$$

Let say you go to a casino with $n_0 = \$x$ and you start with play the following game. In every round t you bet all your money n_t you have at that round.

With probability $1/2$ you have $n_{t+1} = (9/8) \cdot n_t$. With probability $1/4$ nothing happens $n_{t+1} = n_t$ and with probability $1/4$ you lose $1/2$ of your money: $n_{t+1} = n_t / 2$. You leave when you have less than $\$1$.

Assume that money is arbitrarily divisible, so you can have for example $\$0.00002$.

We are interested in how long it takes until you leave the casino. Execute the following steps

1. Give a definition for n_t for $t \in \mathbb{N}$.
2. Calculate $E[n_t - n_{t+1} | n_t]$.
3. Calculate $f(n_t)$.
4. Bound $E[T]$ using Wald's equation.

[Exercise 4 Solutions](https://s3.amazonaws.com/coursera-uploads/peer-review/bnQLDSclEeWbiBJCM9ziNQ/b77e3aa2a9eecde7faad2450081c94e/exercise.pdf) (<https://s3.amazonaws.com/coursera-uploads/peer-review/bnQLDSclEeWbiBJCM9ziNQ/b77e3aa2a9eecde7faad2450081c94e/exercise.pdf>)

Exercise 4 Solutions

One possible solution:

1. Let n_t be the money of the gambler at round t .

We have

$$E[n_{t+1} | n_t] = (9/8)n_t \cdot 1/2 + n_t \cdot 1/4 + n_t / 2 \cdot 1/4 = \frac{15}{16}n_t. \text{ (For marking: Note that } 15/16 = 0.9375\text{)}$$

Thus, we derive

$$2. E[n_t - n_{t+1} | n_t] = n_t - E[n_{t+1} | n_t] = n_t - \frac{15}{16}n_t = (1 - \frac{15}{16})n_t. \text{ (For marking: Note that } 1-15/16 = 1/16 = 0.0625\text{)}$$

$$3. \text{ Hence } f(n_t) = (1 - \frac{15}{16})n_t$$

$$\text{and } E\left[\int_{n_{T-1}}^{n_0} \frac{1}{f(z)} dz\right] = (1 - \frac{15}{16})^{-1} (\log(n_0) - \log(n_{T-1})) \leq (1 - \frac{15}{16})^{-1} \cdot \log(x) = 16 \log(x).$$

$$4. \text{ Hence } E[T] \leq 16 + 16 \log(x).$$



Refik Arkut
Correct



Gilad Kutil
--



David Ellis Hershkowitz
Nice job!

1. The definition for \$1\$ matches the definition proposed by the solution.

- 2 pts
Yes
- 0 pts
No

Help Center

2. The conclusion $E[n_t - n_{t+1} | n_t] = \dots = (1 - \frac{15}{16})n_t$. is stated. (Or $= n_t/16$, or $= 0.0625n_t$.)

- 2 pts
Yes
- 0 pts
No

3. The conclusion that $f(n_t) = (1 - \frac{15}{16})n_t$ or an equivalent version is stated.

- 2 pts
Yes
- 0 pts
No

4. $E[T]$ is bounded by something which is at most $16 + 16\log(x)$.

- 2 pts
Yes
- 0 pts
No

Is the answer presented in a clear and comprehensible way?

- 2 pts
Yes
- 0 pts
No

You ask ten of your friends to give you one dollar (coin) each so that you can buy your favourite pizza.

When you reach the pizza place you realize that it's closed. So you

return the coins to your friends by giving each of them a random coin.

1. How likely is that each of them gets the coin they put back?
2. How many of them will get in expectation the coin they put back? (Try to prove this elegantly using techniques discussed in the course) Define the random variables you use.
3. Now suppose that you take one coin to buy a coffee somewhere else. You redistribute the others dollar coins giving each of your friends one coin chosen uniformly at random. How many of your friends will get in expectation the coin they put back?

Defining the random variables

- Let's number the coins (w.l.o.g.) s.t. the coin that the i^{th} friend has initially as coin i , $\forall i \in \{1, 2, \dots, 10\}$.
- Let X_i denote the **random variable** representing the coin that the i^{th} friend gets back.
- Obviously, for the i^{th} friend, $X_i \in \{1, 2, \dots, 10\}$ (since he can get back any of the 10 coins), with $P(X_i = j) = \frac{1}{10}$, $j \in \{1, 2, \dots, 10\}$ (all the events are equally likely).
- The i^{th} friend gets back his **own** coin $\Leftrightarrow X_i = i$.
- Let's define the **indicator** random variable $1(X_i)$ to be the variable denoting whether the i^{th} friend gets back his coin or not.

$$\begin{aligned} 1(X_i) &= 1, \text{ if } X_i = i \\ &= 0, \text{ otherwise} \end{aligned}$$

1. How likely is that each of them gets the coin they put back?

The **probability** that each of them gets the coin they put back =

$$\begin{aligned} &P(\bigcap_{i=1}^{10} 1(X_i) = 1) \\ &= P(1(X_1) = 1) \cdot P(1(X_2) = 1 | 1(X_1) = 1) \cdot P(1(X_3) = 1 | 1(X_1) = 1 \wedge 1(X_2) = 1) \dots P(1(X_{10}) = 1 | 1(X_1) = 1 \wedge 1(X_2) = 1 \wedge \dots 1(X_9) = 1) \\ &= \frac{1}{10} \cdot \frac{1}{9} \cdot \frac{1}{8} \cdots \frac{1}{1} \\ &= \frac{1}{10!} \end{aligned}$$

2. How many of them will get in expectation the coin they put back? (Try to prove this elegantly using techniques discussed in the course)

The **expected number of friends** getting their coins back =

$$\begin{aligned} &E\left[\sum_{i=1}^{10} 1(X_i)\right] \\ &= \sum_{i=1}^{10} E[1(X_i)] \text{ (by linearity of expectation)} \\ &= \sum_{i=1}^{10} P(X_i = i) \\ &= \sum_{i=1}^{10} \frac{1}{10} \text{ (all the 10 values are equally likely)} \\ &= 1 \end{aligned}$$

3. Now suppose that you take one coin to buy a coffee somewhere else. You redistribute the others dollar coins giving each of your friends one coin chosen uniformly at random. How many of your friends will get in expectation the coin they put back?

- Let's w.l.o.g. assume that I have taken the 10^{th} coin to buy a coffee.
- Obviously, for the i^{th} friend (where $1 \leq i \leq 9$), $X_i \in \{1, 2, \dots, 9\}$ (since he can get back any of the remaining 9 coins), with $P(X_i = j) = \frac{1}{9}$, $j \in \{1, 2, \dots, 9\}$ (all the events are equally likely).
- For the 10^{th} friend, he can never get back his **own** coin, hence, $P(X_{10} = 10) = 0$.
- Hence, now the **expected number of friends** getting their coins back =

$$\begin{aligned} &E\left[\sum_{i=1}^{10} 1(X_i)\right] \\ &= \sum_{i=1}^{10} E[1(X_i)] \text{ (by linearity of expectation)} \\ &= \sum_{i=1}^{10} P(X_i = i) \\ &= \sum_{i=1}^9 P(X_i = i) + P(X_{10} = 10) \\ &= \sum_{i=1}^9 \frac{1}{9} + 0 \\ &= 1 \end{aligned}$$

which remains same as earlier.

One possible solution:

$$1. \frac{1}{10} \cdot \frac{1}{9} \cdot \frac{1}{8} \cdots \frac{1}{1} \approx 2.75573192 \cdot 10^{-7}$$

2. X_i is \$0/1\$-variable which is \$1\$ if person i gets their coin back.

$E[X_i] = P(X_i = 1)$. Thus, by linearity of expectation,

$$E[\sum_{i \leq 10} X_i] = \sum_{i \leq 10} E[X_i] = 10 \cdot \frac{1}{9} = \frac{10}{9}$$

3. W.l.o.g. assume the coin of person 10 was used for your coffee.

Hence, we get $E[\sum_{i \leq 9} X_i] = \sum_i E[X_i] = 9 \cdot \frac{1}{9} = 1$.



Refik Arkut
Correct.



Gilad Kutiel
--



David Ellis Hershkowitz
Nice job!

The value calculated for 1. is correct

- 2 pts
Yes
- 0 pts
No

A random variable was defined

- 1 pt
Yes
- 0 pts
No

The result of 2. is 1

- 2 pts
Yes
- 0 pts
No

The result of 3. is 0.9

- 3 pts
Yes
- 0 pts
No

Is the answer presented in a clear and comprehensible way?

- 2 pts
Yes
- 0 pts
No

Edit submission

Comments

Visible to classmates



share your thoughts...

◀ (/learn/approximation-algorithms-part-

(https://account.coursera.org/supplement/E6Bd/practice-exercise)=https://learner.coursera.org/suppli

▶ (/learn/approximation-algorithms-part-

(https://account.coursera.org/supplement/BbCuD/all-slides-together-in-one-file)

Exercise 5

- Consider the following version of Wald's equation:

Given a random sequence n_0, n_1, \dots, n_T , where T is a stopping time and $n_i \geq \frac{1}{2}$ for $i \leq T$. If $E[n_t - n_{t+1}|n_t] \geq f(n_t)$, $\forall t$, where $f(\cdot)$ is an increasing and differentiable function, then $E[T] \leq \frac{1}{2f(\frac{1}{2})} + E[\int_{n_{T-1}}^{n_0} \frac{1}{f(z)} dz]$.

- Let's say you go to a casino with $n_0 = \$x$ and you start with play the following game. In every round t you bet all your money n_t you have at that round.
- With probability $\frac{1}{2}$ you have $n_{t+1} = \frac{3}{8} \cdot n_t$.
- With probability $\frac{1}{4}$ nothing happens $n_{t+1} = n_t$ and
- with probability $\frac{1}{4}$ you lose $\frac{1}{2}$ of your money: $n_{t+1} = \frac{n_t}{2}$.
- You leave when you have less than $\$1$.
- Assume that money is *arbitrarily divisible*, so you can have for example $\$0.00002$.
- We are interested in how long it takes until you leave the casino. Execute the following steps:

- Give a definition for n_t for $t \in \mathbb{N}$.

n_t is the money left (I have with me) at round t .

We have, $n_0 = \$x$, $n_T < 1$ by condition, also, $n_{T-1} \geq 1$.

Obviously, $n_i \geq 1 > \frac{1}{2}$, $\forall i \leq T-1$.

Also, by condition, $n_T \geq \frac{n_{T-1}}{2} \geq 1 \Rightarrow n_T \geq \frac{1}{2}$.

Combining, we have $n_i \geq \frac{1}{2}$, $\forall i \leq T$, satisfying the condition of Wald's equation.

- Calculate $E[n_t - n_{t+1}|n_t]$.

As given, I shall have $(n_t - n_{t+1})$

$= -\frac{1}{8}n_t$ with probability $\frac{1}{2}$.

$= 0$ with probability $\frac{1}{4}$.

$= \frac{1}{2}n_t$ with probability $\frac{1}{4}$.

Hence, $E[n_t - n_{t+1}|n_t] = \frac{1}{2} \cdot (-\frac{1}{8}n_t) + \frac{1}{4} \cdot 0 + \frac{1}{4} \cdot (\frac{1}{2}n_t) = \frac{1}{16}n_t$.

- Calculate $f(n_t)$.

$f(n_t) = \frac{n_t}{16}$, where $f(\cdot)$ is an increasing and differentiable function, with $f'(n_t) = \frac{1}{16}$, a constant, again satisfying the condition of Wald's equation..

- Bound $E[T]$ using Wald's equation.

$$\begin{aligned} E[T] &\leq \frac{1}{2f(\frac{1}{2})} + E[\int_{n_{T-1}}^{n_0} \frac{1}{f(z)} dz] = 16 + E[\int_{n_{T-1}}^{n_0} \frac{16}{z} dz] \\ &= 16 + 16E[\ln(n_0) - \ln(n_{T-1})] \\ &= 16 + 16\ln(\frac{n_0}{n_{T-1}}) \\ &\leq 16 + 16\ln(x/1). \quad (\text{Since } n_{T-1} \geq 1 \text{ otherwise I shall leave at } T-1). \end{aligned}$$

(also $\ln(x/1) \geq \ln(x/n_{T-1})$, **log** being **increasing** function)

Hence, $E[T] \leq 16(1 + \ln x)$.

Peer Assignment: Set Cover

Clark Grubb

January 2, 2016

Consider the following version of Wald's equation: Given a random sequence n_0, n_1, \dots, n_T , where T is a stopping time and $n_i \geq 1/2$ for $i \leq T$. If $E[n_t - n_{t+1}|n_t] \geq f(n_t)$ for all t , where $f(\cdot)$ is an increasing and differentiable function, then

$$E[T] \leq \frac{1}{2f(1/2)} + E \left[\int_{n_{T-1}}^{n_0} \frac{1}{f(z)} dz \right] \quad (1)$$

Let say you go to a casino with n_0 as your initial amount of money and you play the following game. In every round t you bet all your money n_t you have at that round.

With probability $1/2$ you have $n_{t+1} = (9/8) \cdot n_t$. With probability $1/4$ nothing happens $n_{t+1} = n_t$ and with probability $1/4$ you lose $1/2$ of your money: $n_{t+1} = n_t/2$. You leave when you have less than \$1.

Assume that money is arbitrarily divisible, so you can have for example \$0.00002.

We are interested in how long it takes until you leave the casino. Execute the following steps

Q1. Give a definition for n_t for $t \in \mathbb{N}$.

A1. n_0 is fixed. n_{t+1} is defined in terms of n_t :

$$n_{t+1} = \begin{cases} (9/8) \cdot n_t & 0 \leq p \leq 1/2 \\ n_t & 1/2 < p \leq 3/4 \\ n_t/2 & 3/4 < p \leq 1 \end{cases}$$

Q2. Calculate $E[n_t - n_{t+1}|n_t]$.

A2.

$$\begin{aligned}
E[n_t - n_{t+1} | n_t] &= 1/2 \cdot (n_t - (9/8)n_t) + 1/4 \cdot (n_t - n_t) + 1/4(n_t - n_t/2) \\
&= \frac{-n_t}{16} + \frac{n_t}{8} \\
&= \frac{n_t}{16}
\end{aligned}$$

Q3. Calculate $f(n_t)$.

A3. If we want to apply (1), we can use $f(n_t) = n_t/16$.

Q4. Bound $E[T]$ using Wald's equation.

A4.

$$\begin{aligned}
E[T] &\leq \frac{1}{2f(1/2)} + E \left[\int_{n_{T-1}}^{n_0} \frac{1}{z} dz \right] \\
&= \frac{1}{2 \cdot 1/2 \cdot 1/16} + E \left[\int_{n_{T-1}}^{n_0} \frac{16}{z} dz \right] \\
&= 16 + E \left[16 \ln z \Big|_{n_{T-1}}^{n_0} \right] \\
&= 16 + E[16 \ln n_0 - 16 \ln n_{T-1}] \\
&= 16[1 + \ln n_0 - E[\ln n_{T-1}]] \\
&< 16(1 + \ln n_0)
\end{aligned}$$

For the last step, recall that by the stopping condition, $n_{T-1} > 1$ and therefore $\ln n_{T-1} > 0$.

You ask ten of your friends to give you one dollar (coin) each so that you can buy your favourite pizza. When you reach the pizza place you realize that it's closed. So you return the coins to your friends by giving each of them a random coin.

Q5. How likely is it that each of them gets the coin they put back?

A5. Each friend has a 1 in 10 chance of getting their coin back.

Q6. How many of them will get in expectation the coin they put back? Try to prove this elegantly using techniques discussed in the course. Define the random variables you use.

A6. Let X_i be a Bernoulli variable which is 1 if the i -th friend gets their coin back. The expected number of friends who get their coin back is

$$E \left[\sum_{i=1}^{10} X_i \right] = \sum_{i=1}^{10} \frac{1}{10} = 1$$

Now suppose that you take one coin to buy a coffee somewhere else. You redistribute the others dollar coins giving each of your friends one coin chosen uniformly at random.

Q7. How many of your friends will get in expectation the coin they put back?

A7. The expected number of friends with coins is the same as if the spent coin is marked, all of the coins are redistributed, and then the friend with the marked coin gives up their coin. Let X_i be as before, namely the number of coins held by the i -th friend after redistribution, and let Y_i be the number of coins held by the i -th friend after the marked coin is given up. We are free to renumber the friends so that the tenth friend gives up their coin. Then $Y_i = X_i$ for $i < 10$ and $Y_{10} = 0$. Hence

$$E \left[\sum_{i=1}^{10} Y_i \right] = E \left[\sum_{i=1}^9 X_i \right] = \sum_{i=1}^9 \frac{1}{10} = \frac{9}{10}$$



([https://accounts.coursera.org/i/zendesk/courserahelp?
return_to=https://learner.coursera.help/hc](https://accounts.coursera.org/i/zendesk/courserahelp?return_to=https://learner.coursera.help/hc))

Congratulations! [View Final Grade](#)

 You've completed the course. (</learn/approximation-algorithms-part-1/home/welcome>)

Set Cover and Randomized Rounding



Claire Mathieu

This module introduces a simple and powerful variant of rounding, based on probability: randomized rounding. Its power is applied to another basic problem, the Set Cover problem.

Set Cover Definition

 Lecture: definition 14 min

(</learn/approximation-algorithms-part-1/lecture/LHFwj/lecture-definition>)

 Slides

(</learn/approximation-algorithms-part-1/supplement/cfyCJ/slides>)

[Help Center](#)

 Practice Quiz: Quiz

1 1 question

(</learn/approximation-algorithms-part-1/quiz/g2Pnm/quiz-1>)

Randomized Rounding

-  Lecture: randomized rounding 4 min

(/learn/approximation-algorithms-part-1/lecture/BT5Dg/lecture-randomized-rounding)

-  Slides

(/learn/approximation-algorithms-part-1/supplement/nPRtu/slides)

-  Practice Quiz: Quiz
2 1 question

(/learn/approximation-algorithms-part-1/quiz/kW9fM/quiz-2)

Cost Analysis

-  Lecture: cost analysis 6 min

(/learn/approximation-algorithms-part-1/lecture/lm0cp/lecture-cost-analysis)

-  Slides

(/learn/approximation-algorithms-part-1/supplement/4ZatV/slides)

-  Practice Quiz: Quiz
3 1 question

(/learn/approximation-algorithms-part-1/quiz/ks2MP/quiz-3)

Coverage Analysis

-  Lecture: coverage analysis 9 min

(/learn/approximation-algorithms-part-1/lecture/SAV1y/lecture-coverage-analysis)

 Slides

(/learn/approximation-algorithms-part-1/supplement/HGpSY/slides)

 Practice Quiz: Quiz
4 2 questions

(/learn/approximation-algorithms-part-1/quiz/FePlA/quiz-4)

Iterated Algorithm

 Lecture: iterated algorithm 4 min

(/learn/approximation-algorithms-part-1/lecture/mAjCi/lecture-iterated-algorithm)

 Slides

(/learn/approximation-algorithms-part-1/supplement/XRmxW/slides)

 Practice Quiz: Quiz
5 1 question

(/learn/approximation-algorithms-part-1/quiz/W6Yel/quiz-5)

Stopping Time Algorithm

 Lecture: stopping time algorithm 4 min

(/learn/approximation-algorithms-part-1/lecture/nnoyE/lecture-stopping-time-algorithm)

 Slides

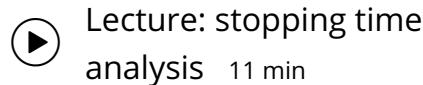
(/learn/approximation-algorithms-part-1/supplement/Pm0MU/slides)

 Practice Quiz: Quiz
6 1 question

(/learn/approximation-algorithms-part-

1/quiz/J8VWL/quiz-6

Stopping Time Analysis



Lecture: stopping time analysis 11 min

(/learn/approximation-algorithms-part-1/lecture/xYzLS/lecture-stopping-time-analysis)



Slides

(/learn/approximation-algorithms-part-1/supplement/3hlfY/slides)



Practice Quiz: Quiz

7 1 question

(/learn/approximation-algorithms-part-1/quiz/4nLtb/quiz-7)

Remarks



Lecture:final remarks 6 min

(/learn/approximation-algorithms-part-1/lecture/fj7s/lecture-final-remarks)



Slides

(/learn/approximation-algorithms-part-1/supplement/LgNVU/slides)



Practice Quiz: Quiz

8 2 questions

(/learn/approximation-algorithms-part-1/quiz/AKBUv/quiz-8)



A reference on this stopping time analysis

(/learn/approximation-algorithms-part-1/supplement/R81QO/a-reference-on-this-stopping-time-analysis)

Exercises

Practise Exercise

(/learn/approximation-algorithms-part-1 supplement/E6aBd/practise-exercise)

-  **Assignment:** Peer Assig
- Set Cover 30 min

(/learn/approximation-algorithms-part-1/peer/pLGal/peer-assig-set-cover)

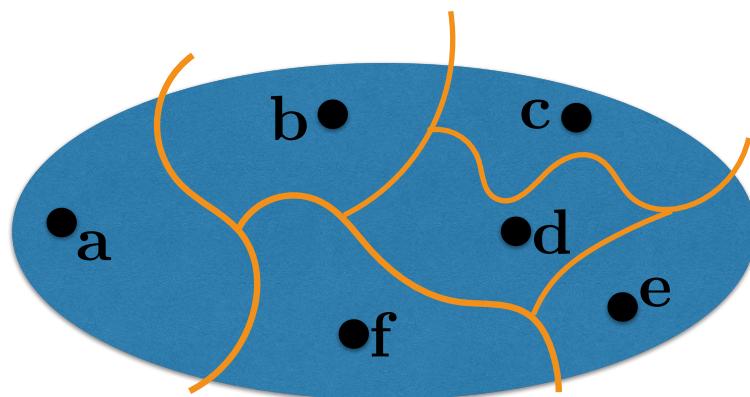
-  **Review Classmates:** Peer
- Assig Set Cover 30 min

(/learn/approximation-algorithms-part-1/peer/pLGal/peer-assig-set-cover/give-feedback)

All slides together in one file

(/learn/approximation-algorithms-part-1/supplement/BbCuD/all-slides-together-in-one-file)

Multiway cut, linear programming and randomized rounding



Let k be a fixed integer.

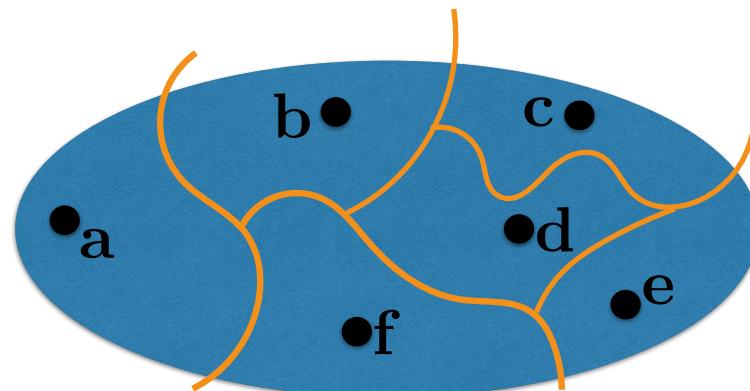
Given: graph G with edge weights,
 k vertices called "terminals"

Find: subset F of edges such that

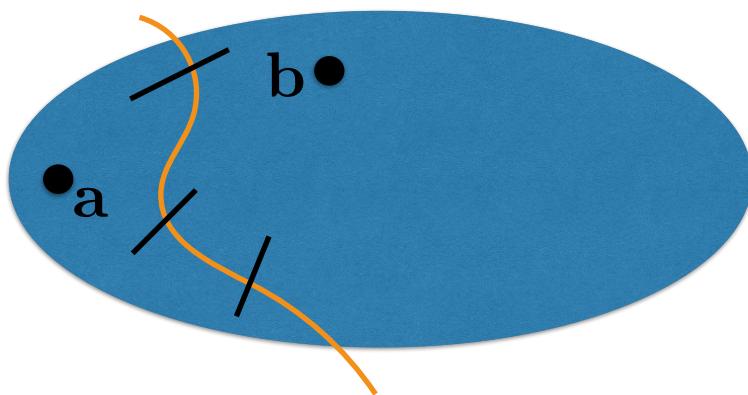
Removing F disconnects the
terminals from one another

Goal: minimize weight of F

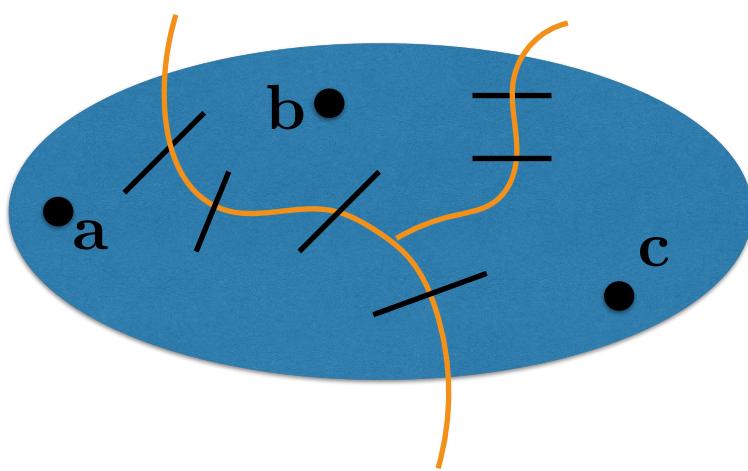
Problem



$k=2$
**min cut
in P**



$k=3$
NP-hard



Simple algorithm for k=3

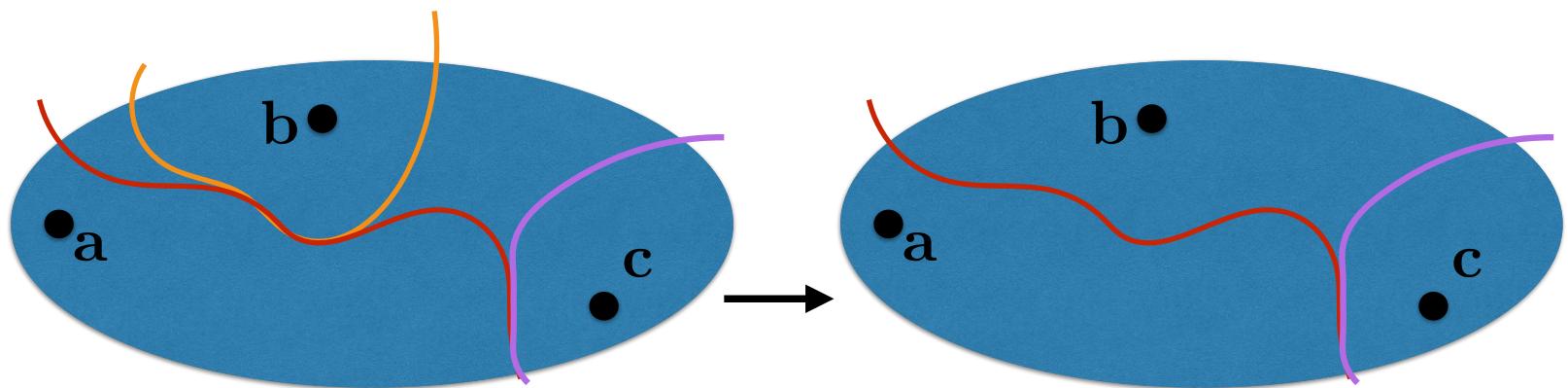
Terminals: a,b,c

Output the two smallest of

{ $\text{Mincut}(a, \{b, c\})$,

$\text{Mincut}(b, \{c, a\})$,

$\text{Mincut}(c, \{a, b\})$ }



Analysis

- It takes polynomial time...
- It's a correct multiway cut:
a,b,c are separated from one another
- But how good is it?

Cost of output

Output is at most

$$(2/3) (\text{Mincut}(a,bc) + \text{Mincut}(b,ac) + \text{Mincut}(c,ab))$$

- OPT is at least $\text{Mincut}(a,bc)$
- OPT is at least $\text{Mincut}(b,ac)$
- OPT is at least $\text{Mincut}(c,ab)$

So OPT is at least

$$(\text{Mincut}(a,bc) + \text{Mincut}(b,ac) + \text{Mincut}(c,ab))/3$$

Alg is a 2 approximation

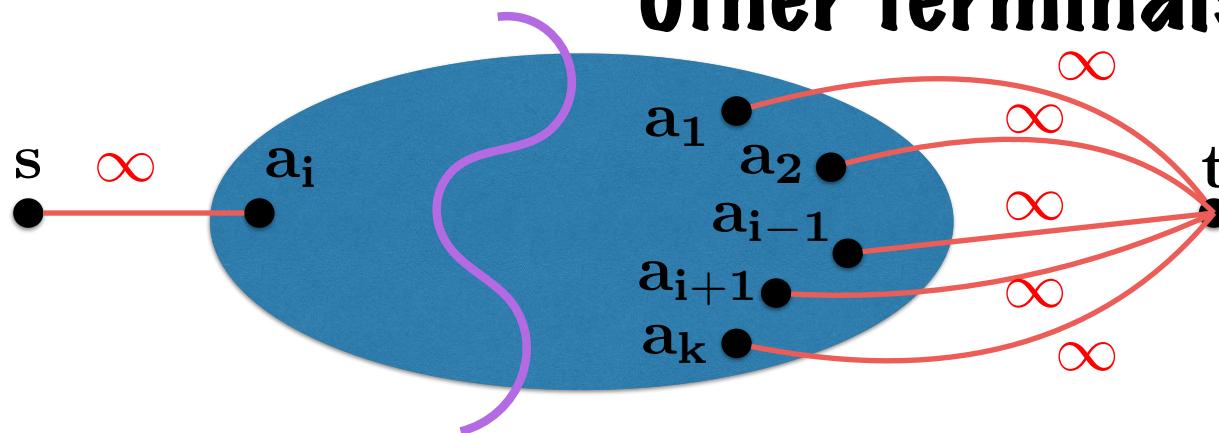
Extension: algorithm for k

Terminals: a_1, a_2, \dots, a_k

Output the $k-1$ smallest of

{ $\text{Mincut}(a_i, \{a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_k\})$: $i=1, 2, \dots, k$ }

To compute the min cut separation of a_i from the other terminals:



$\text{Mincut}(s, t)$

Analysis of output

**The $k-1$ smallest cuts cost
less than a random choice of $k-1$, so**

$$\text{Cost}(\text{Output}) \leq \frac{k-1}{k} \sum_i \text{Mincut}(a_i, \{a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_k\})$$

Analysis of OPT

Let $F(i) = \{\text{edges of OPT that separate } a_i \text{ from some } a_j\}$

Consider e in OPT

e separates some a_i from some a_j
 e belongs to $F(i)$ and to $F(j)$

so

$$\sum_i \text{Cost}(F_i) = 2\text{OPT}$$

$F(i) = \{ \text{edges of OPT that separate } a_i \text{ from some } a_j \}$

$F(i)$ separates a_i
from all other terminals
so

$\text{Cost}(F_i) \geq$
 $\text{Mincut}(a_i, \{a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_k\})$

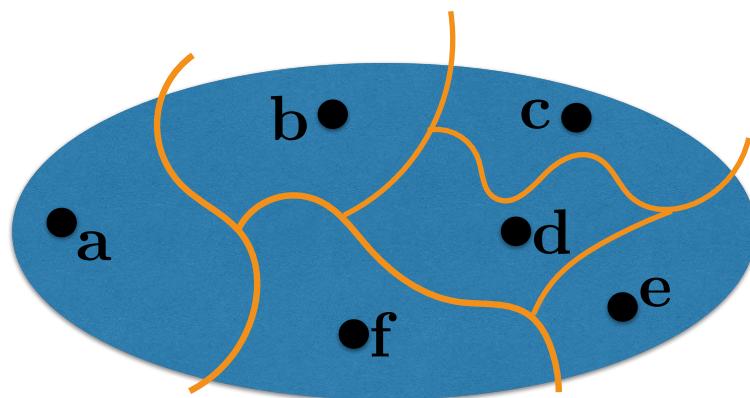
Together

$$\begin{aligned} & \text{Cost(Output)} \\ & \leq \frac{k-1}{k} \sum_i \text{Cost}(F_i) \\ & \leq 2\left(1 - \frac{1}{k}\right) \text{OPT} \end{aligned}$$

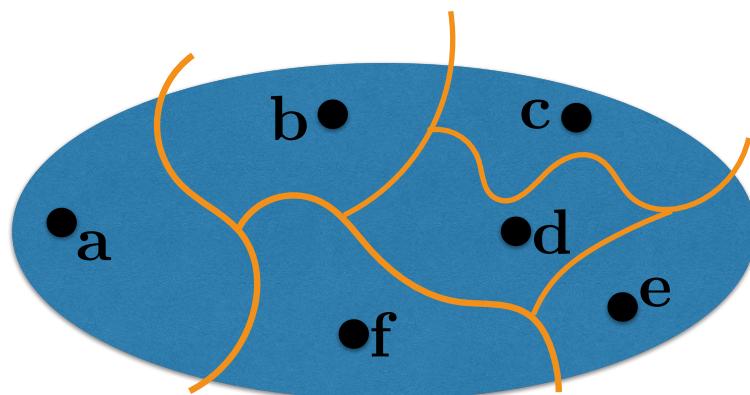
k=2: Optimal

k=3: it's a 4/3 approximation

Multiway cut, linear programming and randomized rounding



Multiway cut, linear programming and randomized rounding

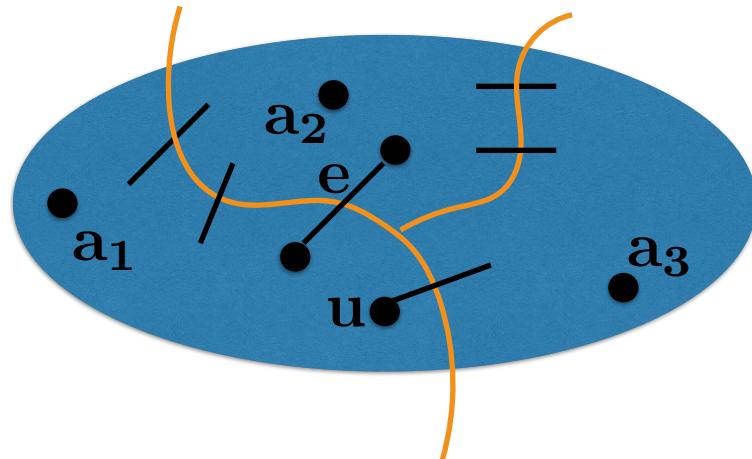


Variablen

$x_{u,i} = 1$ iff
vertex u belongs to the
cluster
of terminal a_i .

$z_{e,i} = 1$ iff
removal of edge e
separates a_i from
some other terminal

IP model



IP model

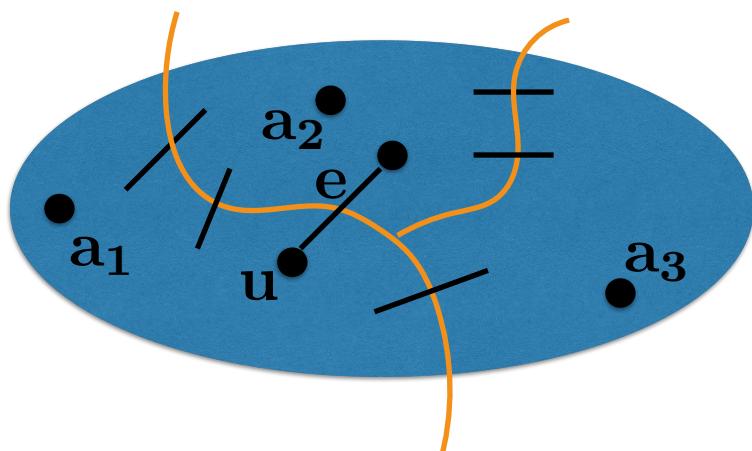
Constraints

$$x_{u,i}, z_{e,i} \in \{0, 1\}$$

$x_{a_i,i} = 1$ (a_i belongs to its own cluster)

$\sum_i x_{u,i} = 1$ (u belongs to some cluster)

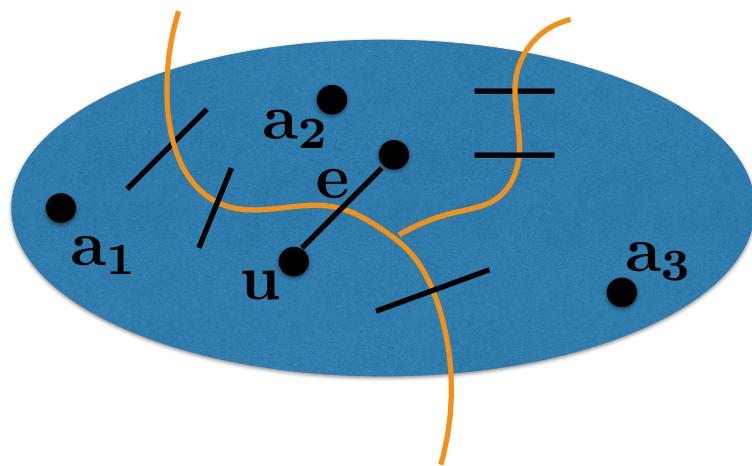
$z_{uv,i} = 1 \quad \text{iff} \quad x_{u,i} \neq x_{v,i} \quad ???$



$z_{uv,i} = 1 \quad \text{iff} \quad x_{u,i} \neq x_{v,i}$ IP model

Objective

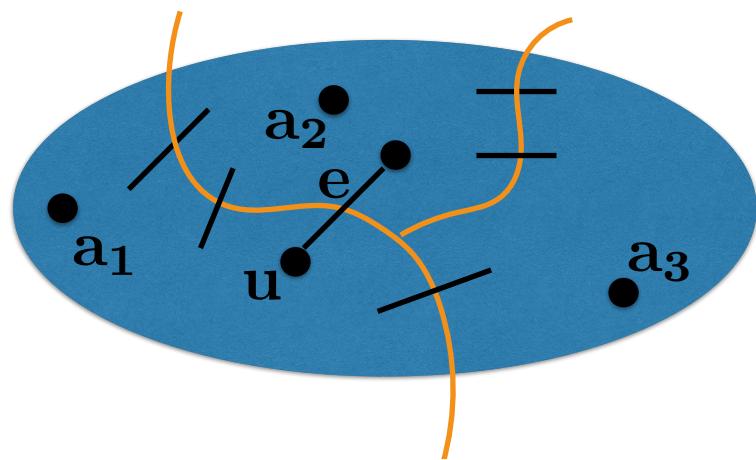
$$\min (1/2) \sum_{e,i} c_e z_{e,i}$$



$$z_{uv,i} = 1 \quad \text{iff} \quad x_{u,i} \neq x_{v,i} \quad \text{IP model}$$

How do we express that constraint?

$$\frac{z_{e,i} \geq |x_{u,i} - x_{v,i}|}{\begin{aligned} z_{e,i} &\geq x_{u,i} - x_{v,i} \\ z_{e,i} &\geq x_{v,i} - x_{u,i} \end{aligned}}$$



IP model

$$x_{u,i}, z_{e,i} \in \{0, 1\}$$

$$x_{a_i,i} = 1$$

$$\sum_i x_{u,i} = 1$$

$$z_{e,i} \geq x_{u,i} - x_{v,i}$$

$$z_{e,i} \geq x_{v,i} - x_{u,i}$$

$$\min (1/2) \sum_{e,i} c_e z_{e,i}$$

Linear programming relaxation

$$x_{u,i}, z_{e,i} \in \{0, 1\} \longrightarrow 0 \leq x_{u,i}, z_{e,i} \leq 1$$

$$x_{a_i,i} = 1$$

$$\sum_i x_{u,i} = 1$$

$$z_{e,i} \geq x_{u,i} - x_{v,i}$$

$$z_{e,i} \geq x_{v,i} - x_{u,i}$$

$$\min (1/2) \sum_{e,i} c_e z_{e,i}$$

A geometric interpretation

Variables

$x_{u,i} = 1$ iff

**vertex u belongs
to the cluster
of terminal ai.**

Vector $\mathbf{x}_u = (x_{u,i})_i$

One dimension for each terminal

$$\sum_i x_{u,i} = \sum_i |x_{u,i}| = |\mathbf{x}_u|_1$$

A geometric interpretation

Vector $\mathbf{x}_u = (x_{u,i})_i$

One dimension for each terminal

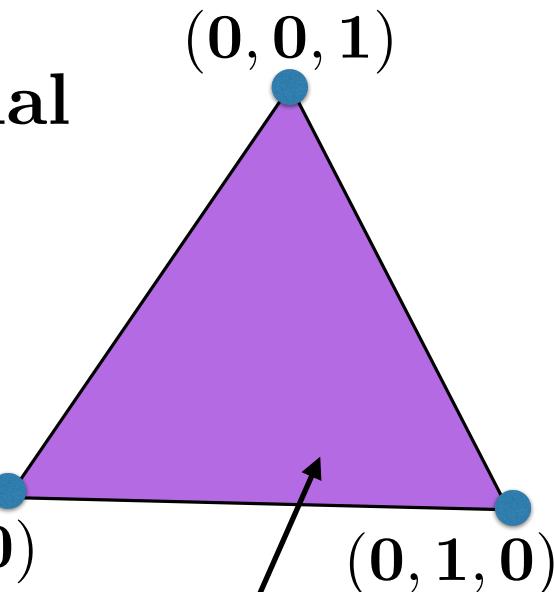
$$\sum_i x_{u,i} = \sum_i |x_{u,i}| = |\mathbf{x}_u|_1$$

$$|\mathbf{x}_u|_1 = 1 \quad \forall u$$

$$\mathbf{x}_u \in [0, 1]^k \quad \forall u$$

$$\mathbf{x}_{a_i} = (0, \dots, 0, 1, 0, \dots, 0) \quad \begin{cases} (1, 0, 0) \\ (0, 1, 0) \end{cases}$$
$$\left\{ \begin{array}{l} x_1 + x_2 + x_3 = 1 \\ x_i \geq 0 \end{array} \right.$$

$$\min \quad (1/2) \sum_{uv \in E} c_{uv} |\mathbf{x}_u - \mathbf{x}_v|_1$$



$$\min \quad (1/2) \sum_{uv \in E} c_{uv} |x_u - x_v|_1$$

$$x_{a_3} = (0, 0, 1)$$

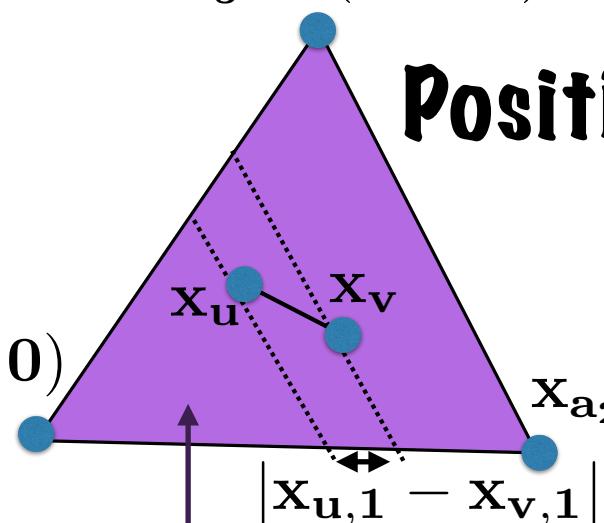
k=3:

**Position graph vertices
in triangle**

$$x_{a_1} = (1, 0, 0)$$

$$x_{a_2} = (0, 1, 0)$$

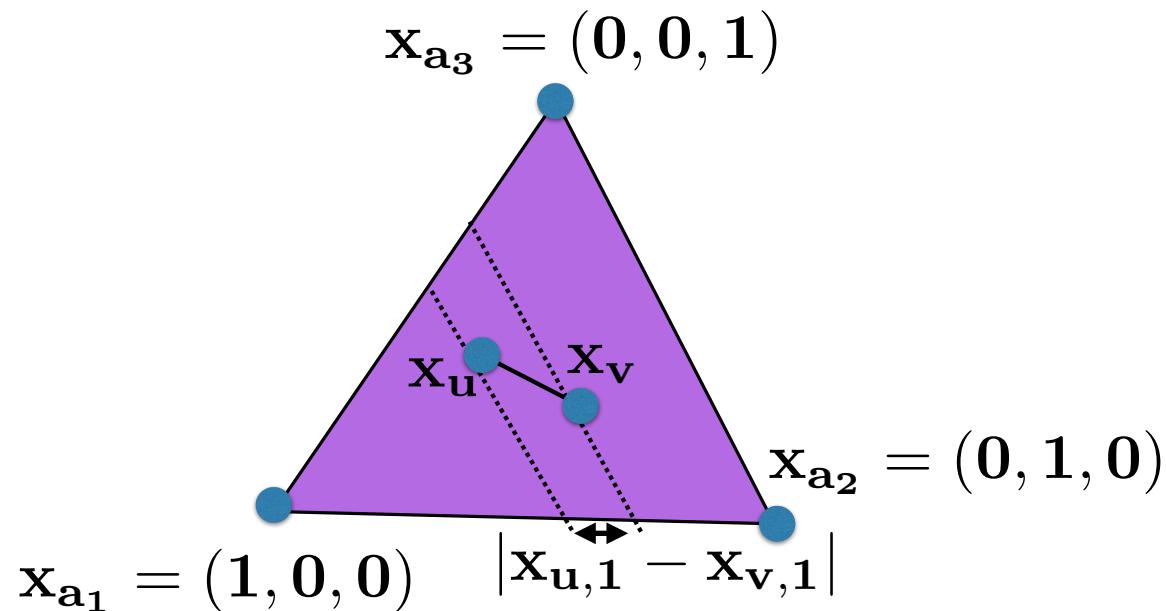
$$\left\{ \begin{array}{l} x_1 + x_2 + x_3 = 1 \\ x_i \geq 0 \end{array} \right.$$



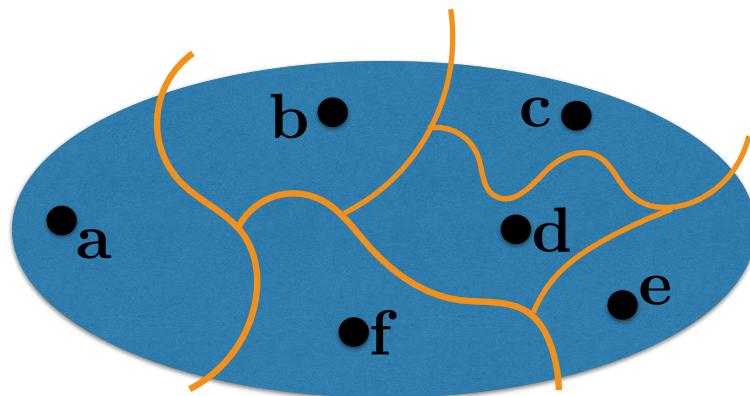
**to minimize
weighted lengths of
projections on sides**

A geometric interpretation ($k=3$)

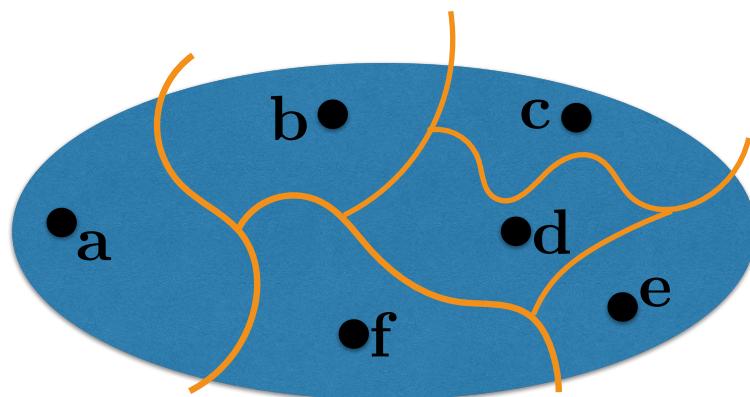
Position graph vertices in triangle
to minimize weighted lengths of
projections of graph edges on the sides



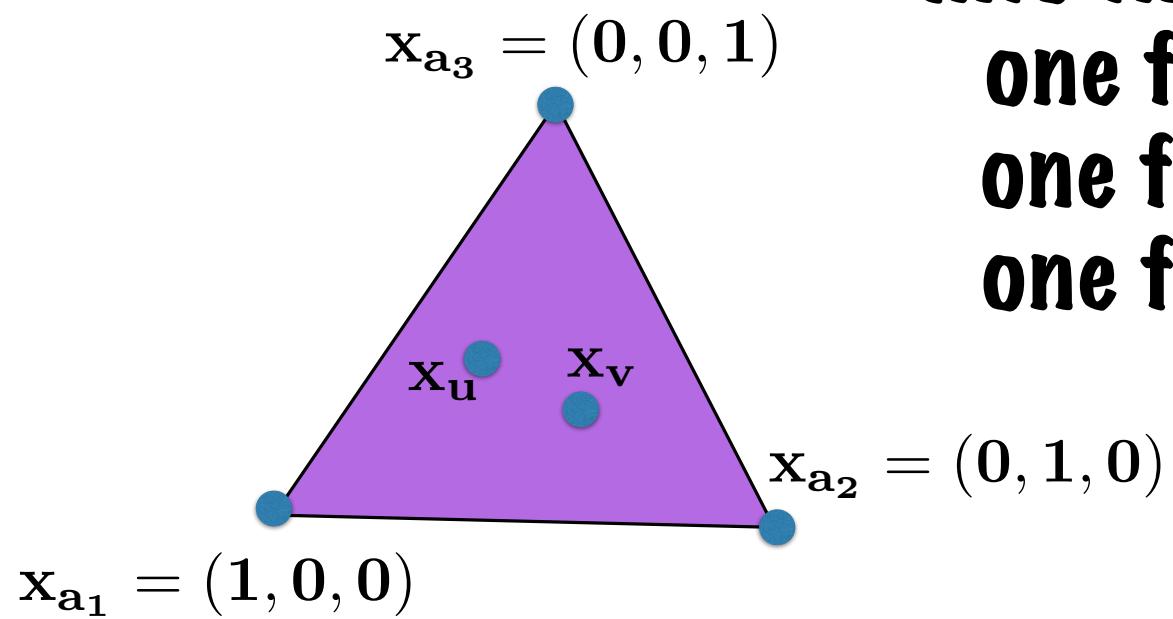
Multiway cut, linear programming and randomized rounding



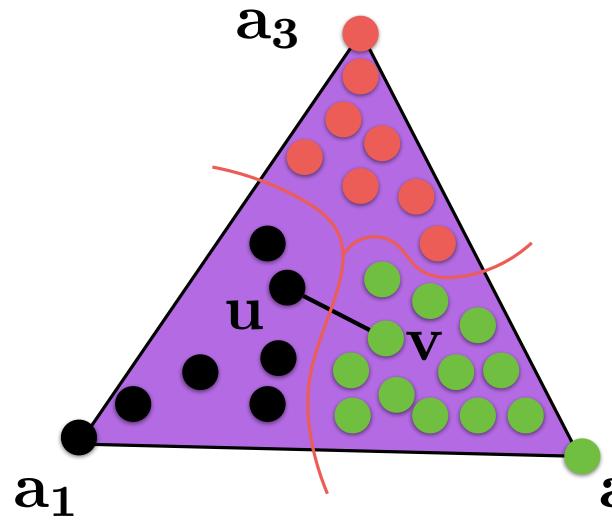
Multiway cut, linear programming and randomized rounding



How do we round?



**Partition triangle
into three areas:
one for $(0,0,1)$
one for $(1,0,0)$
one for $(0,1,0)$**



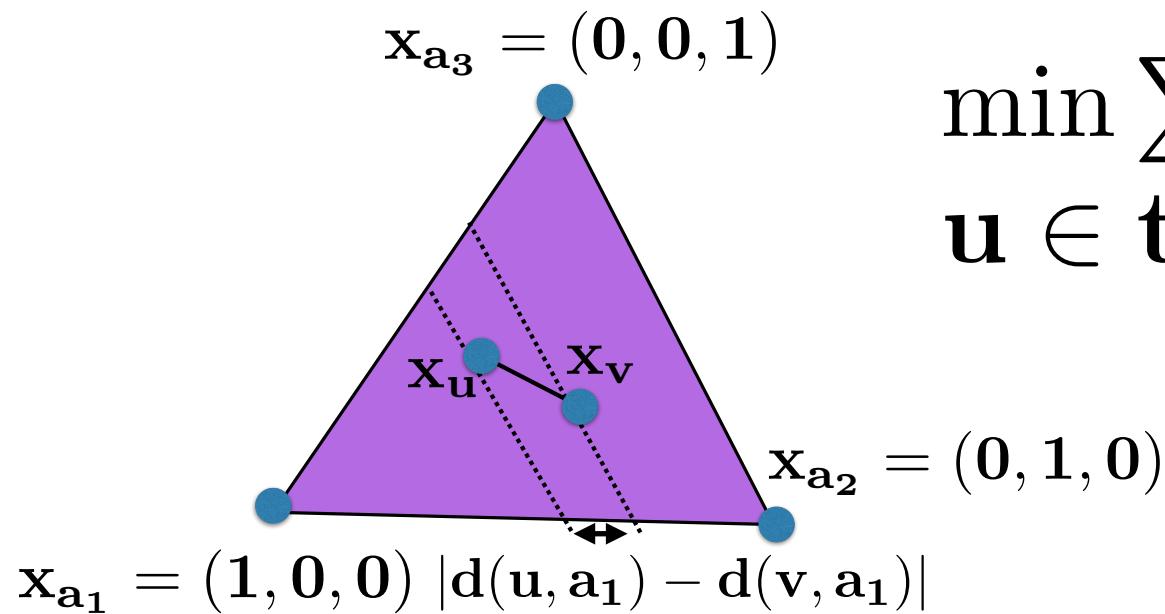
How do we round?

Vertices • go with a1
Vertices • go with a2
Vertices • go with a3
Pay cost of edges across

Good rounding = small cost choice of partition of triangle into three areas

LP relaxation

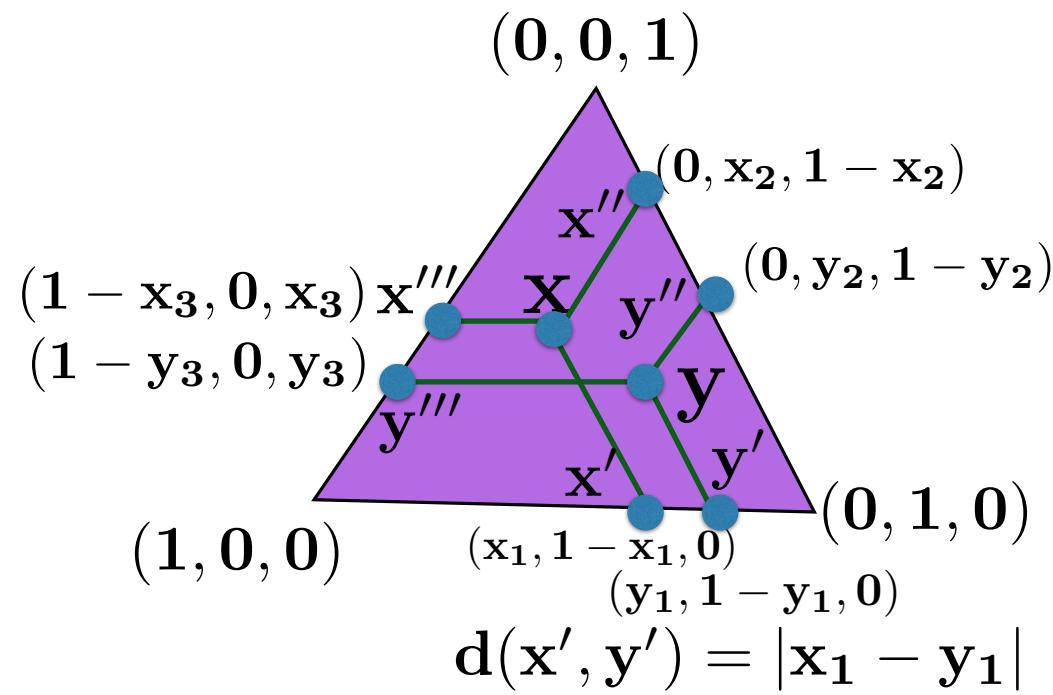
Place vertices in triangle, min lengths of edge projections on sides



$$d(u, v) = \frac{1}{2}|x_u - x_v|_1$$

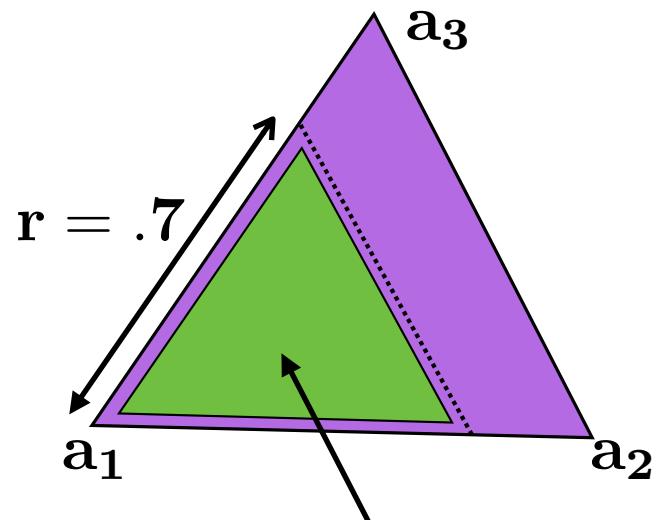
$$\min \sum_{uv \in E} c_{uv} d(u, v) : \\ u \in \text{triangle} \quad \forall u$$

How do we round?



$$\begin{aligned}
\mathbf{d}(\mathbf{u}, \mathbf{v}) &= \frac{1}{2}(|\mathbf{x}_1 - \mathbf{y}_1| + |\mathbf{x}_2 - \mathbf{y}_2| + |\mathbf{x}_3 - \mathbf{y}_3|) \\
&= \frac{1}{2}(\mathbf{d}(\mathbf{x}', \mathbf{y}') + \mathbf{d}(\mathbf{x}'', \mathbf{y}'') + \mathbf{d}(\mathbf{x}''', \mathbf{y}'''))
\end{aligned}$$

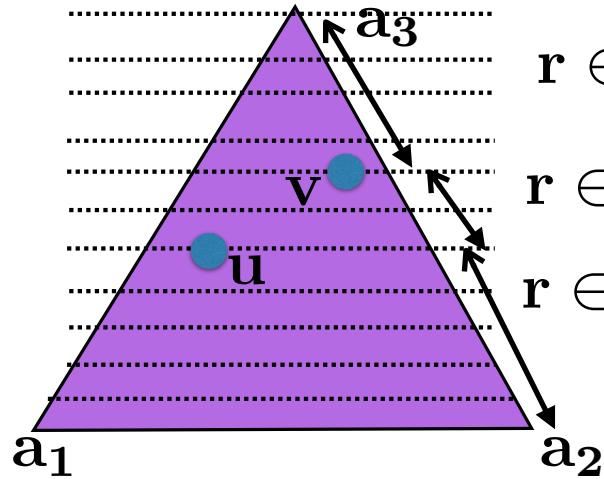
Using balls in ℓ_1 metric



$$d(u, v) = \frac{1}{2}|u - v|_1$$
$$d(a_i, a_j) = 1$$

$$B(a_1, r) = \{u : d(a_1, u) \leq r\}$$

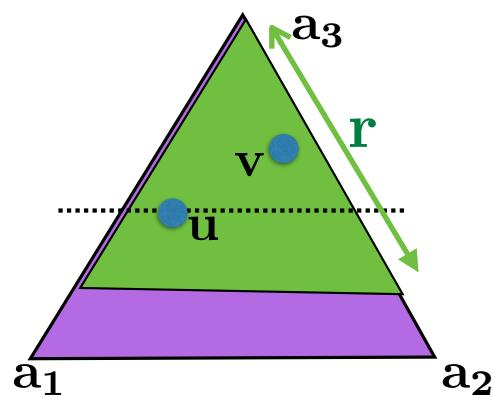
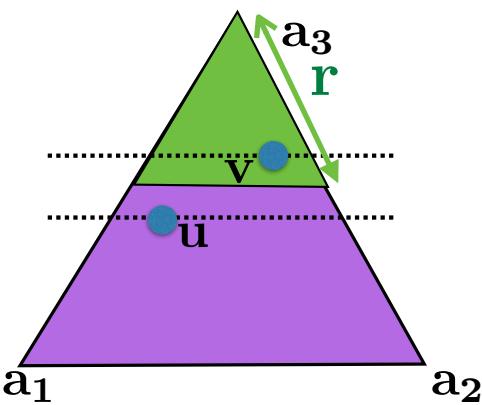
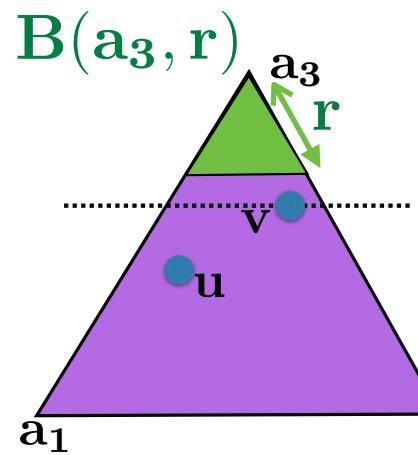
Pick random r , assign $B(a_3, r)$ to a_3



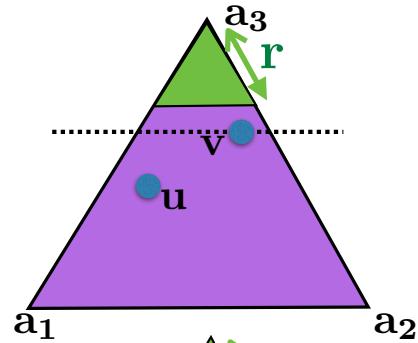
$$r \in [0, d(v, a_3)]$$

$$r \in [d(v, a_3), d(u, a_3)]$$

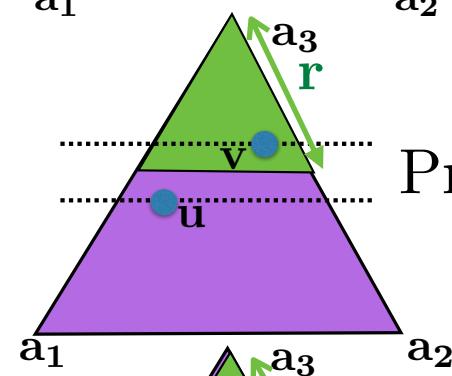
$$r \in [d(u, a_3), 1]$$



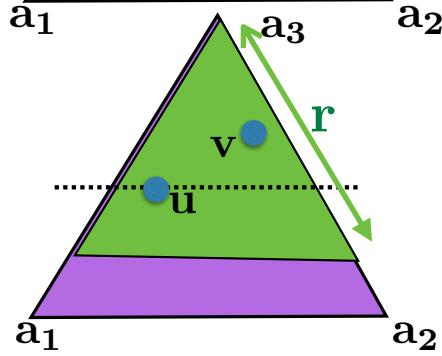
Pick random r , assign $B(a_3, r)$ to a_3



$$\Pr[u, v \notin B(a_3, r)] = \Pr[r < d(a_3, v)] = d(a_3, v)$$



$$\begin{aligned} \Pr[u, v \text{ separated by } B(a_3, r)] &= d(a_3, u) - d(a_3, v) \\ &= |\mathbf{u}_3 - \mathbf{v}_3| \end{aligned}$$



$$\Pr[u, v \in B(a_3, r)] = 1 - d(a_3, u)$$

Consider terminals in random order

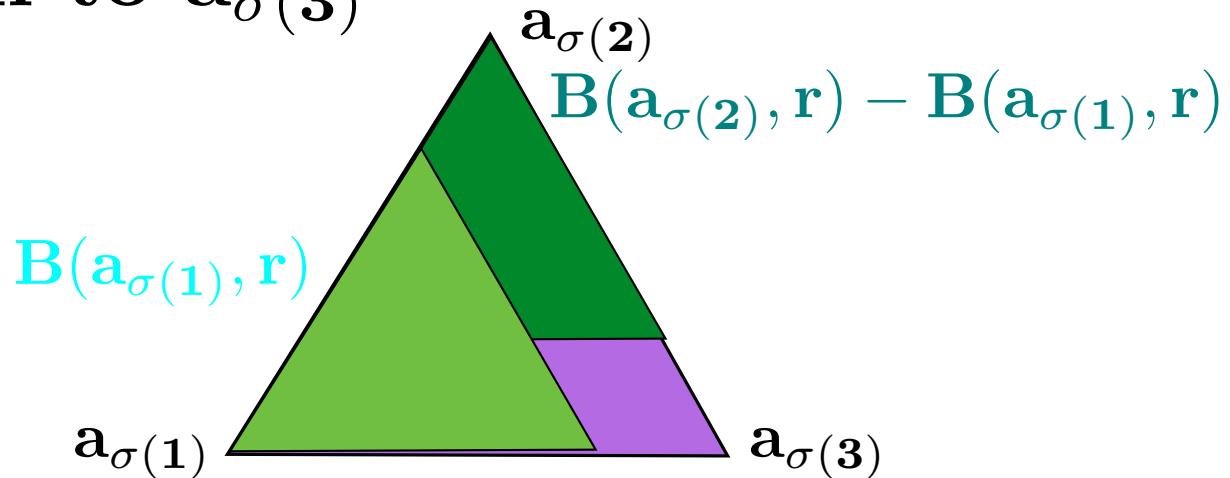
$$\mathbf{a}_{\sigma(1)}, \mathbf{a}_{\sigma(2)}, \mathbf{a}_{\sigma(3)}$$

Assigning u

if $d(\mathbf{a}_{\sigma(1)}, \mathbf{u}) < r$ then assign to $\mathbf{a}_{\sigma(1)}$

else if $d(\mathbf{a}_{\sigma(2)}, \mathbf{u}) < r$ then assign to $\mathbf{a}_{\sigma(2)}$

else assign to $\mathbf{a}_{\sigma(3)}$



Full Algorithm for 3-way cut

Solve relaxation: embed vertices in triangle

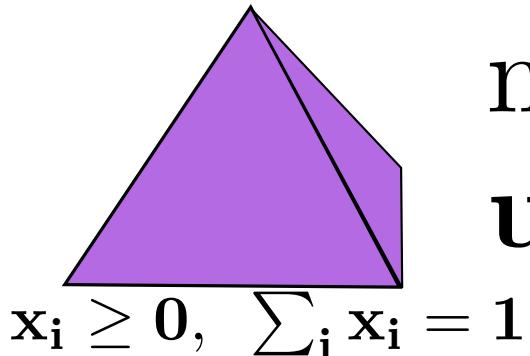
Pick random permutation of terminals

- assign to **first** terminal a all vertices in $B(a,r)$ where r is random uniform in $[0,1]$
- assign to **second** terminal b all unassigned vertices in $B(b,r)$
- assign to **third** terminal remaining vertices

Algorithm

$$d(u, v) = \frac{1}{2} |x_u - x_v|_1$$

**LP relaxation: embed vertices in k-simplex
with terminals at corners**



$$\min \sum_{uv \in E} c_{uv} d(u, v) : \\ u \in \text{k-simplex} \quad \forall u$$

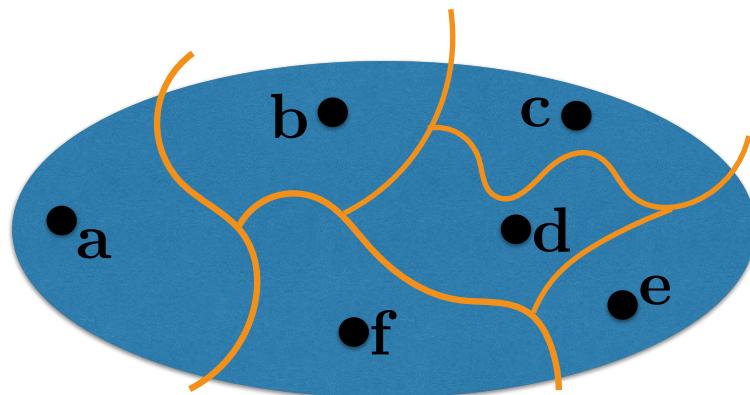
random ordering $a_{\sigma(1)}, a_{\sigma(2)}, \dots, a_{\sigma(k)}$ $r \in [0, 1]$

for $i = 1, \dots, k - 1$:

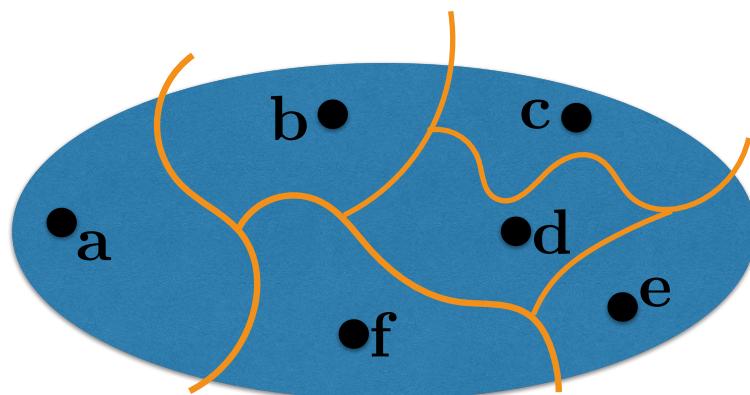
 assign to $a_{\sigma(i)}$ unassigned vertices of $B(a_{\sigma(i)}, r)$

 assign rest to $a_{\sigma(k)}$

Multiway cut, linear programming and randomized rounding



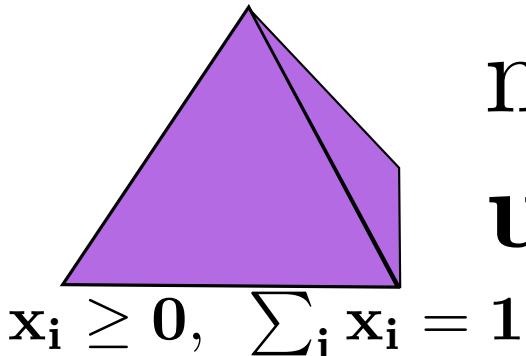
Multiway cut, linear programming and randomized rounding



Algorithm

$$d(u, v) = \frac{1}{2} |x_u - x_v|_1$$

**LP relaxation: embed vertices in k-simplex
with terminals at corners**



$$\min \sum_{uv \in E} c_{uv} d(u, v) : \\ u \in \text{k-simplex} \quad \forall u$$

random ordering $a_{\sigma(1)}, a_{\sigma(2)}, \dots, a_{\sigma(k)}$ $r \in [0, 1]$

for $i = 1, \dots, k - 1$:

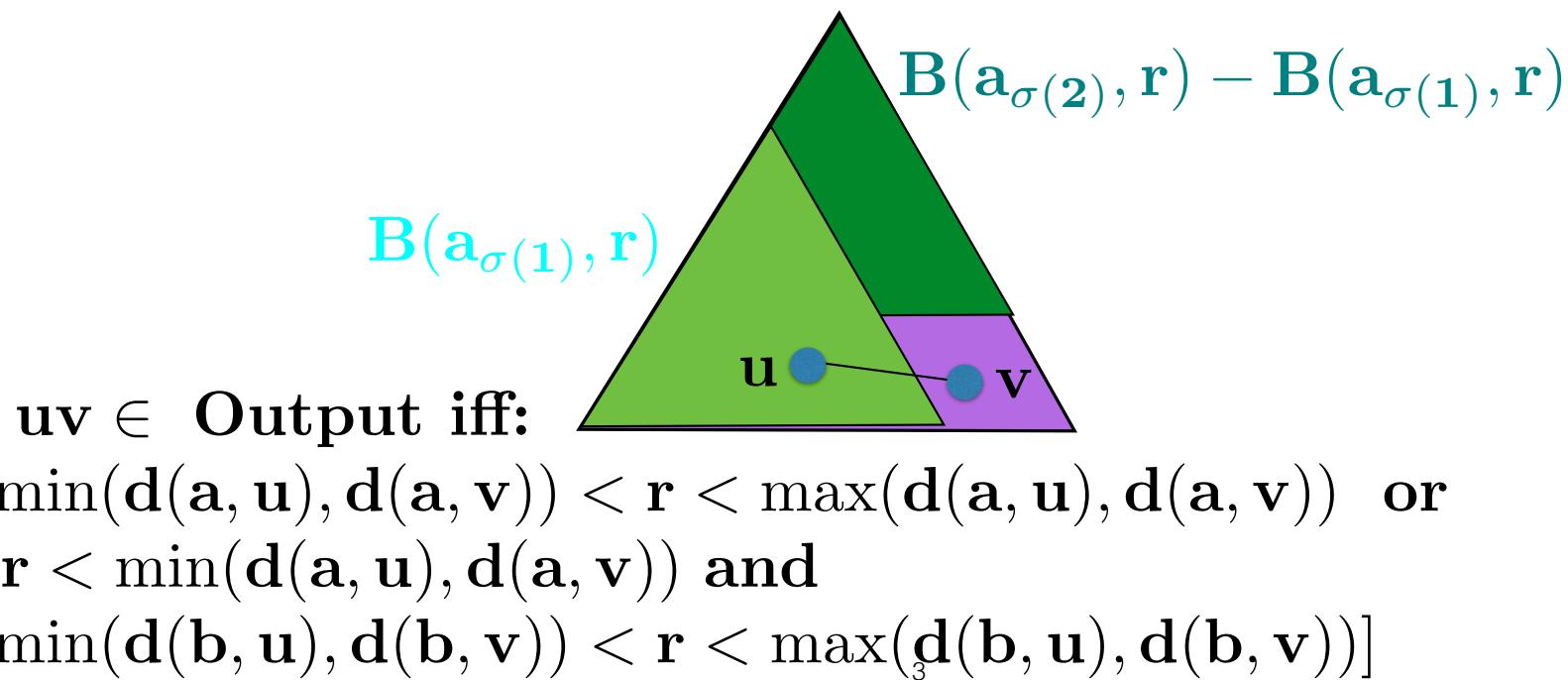
 assign to $a_{\sigma(i)}$ unassigned vertices of $B(a_{\sigma(i)}, r)$

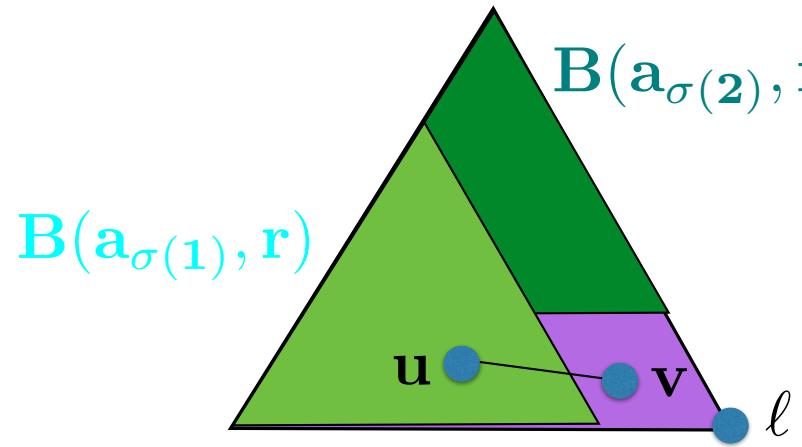
 assign rest to $a_{\sigma(k)}$

Analysis

$$E[Output] = \sum_{uv \in E} c_{uv} \Pr(uv \in Output)$$

$$a = a_{\sigma(1)}, b = a_{\sigma(2)}, c = a_{\sigma(3)}$$





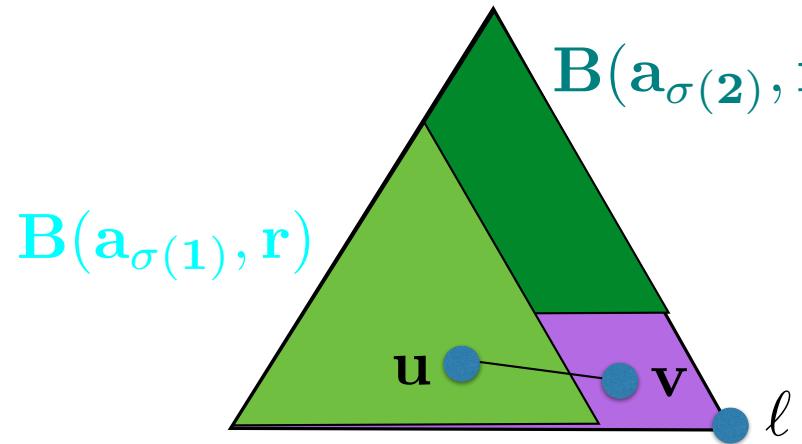
$$B(a_{\sigma(2)}, r) - B(a_{\sigma(1)}, r)$$

uv ∈ Output:
 a_i first to catch u or v
 and catches exactly one

$$\ell = \arg \min \{ \min(d(a_i, u), d(a_i, v)) \}$$

$i \neq \ell : i$ precedes ℓ and $B(a_i, r)$ separates them

$$\frac{1}{2} |d(a_i, u) - d(a_i, v)|$$



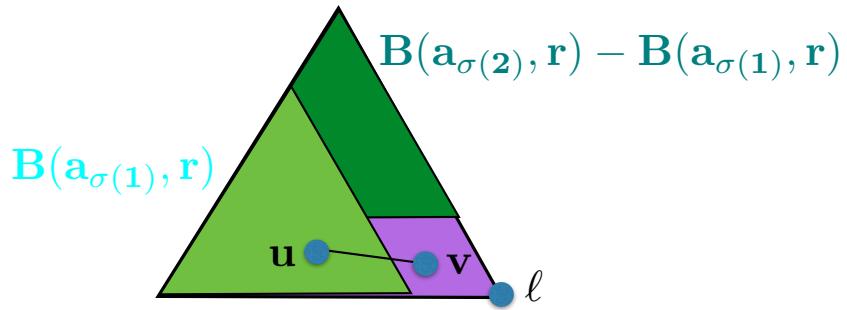
$$B(a_{\sigma(2)}, r) - B(a_{\sigma(1)}, r)$$

$uv \in \text{Output}:$
 a_i first to catch u or v
 and catches exactly one

$$\ell = \arg \min \{ \min(d(a_i, u), d(a_i, v)) \}$$

$i = \ell : \ell$ not last and $B(a_\ell, r)$ separates them

$$(1 - \frac{1}{k}) |d(a_\ell, u) - d(a_\ell, v)|$$

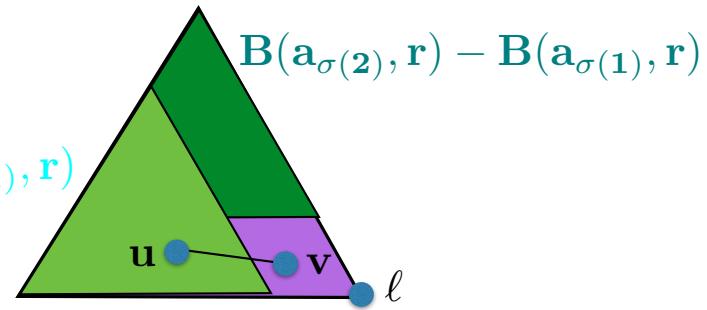


Together

$$(1 - \frac{1}{k})|d(a_\ell, u) - d(a_\ell, v)| + \sum_{i \neq \ell} \frac{1}{2}|d(a_i, u) - d(a_i, v)|$$

$$= \frac{1}{2}(1 - \frac{2}{k})|d(a_\ell, u) - d(a_\ell, v)| + \sum_i \frac{1}{2}|d(a_i, u) - d(a_i, v)|$$

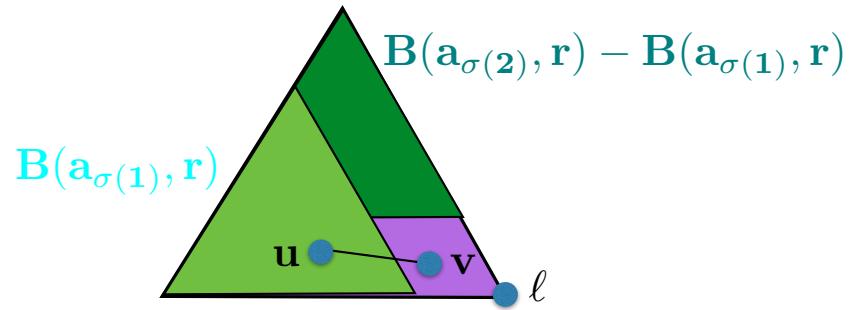
$$= \frac{1}{2}(1 - \frac{2}{k})|u_\ell - v_\ell| + \sum_i \frac{1}{2}|u_i - v_i|$$



$$\sum_{\mathbf{u}} \mathbf{u}_i = \sum_{\mathbf{i}} \mathbf{v}_i = 1$$

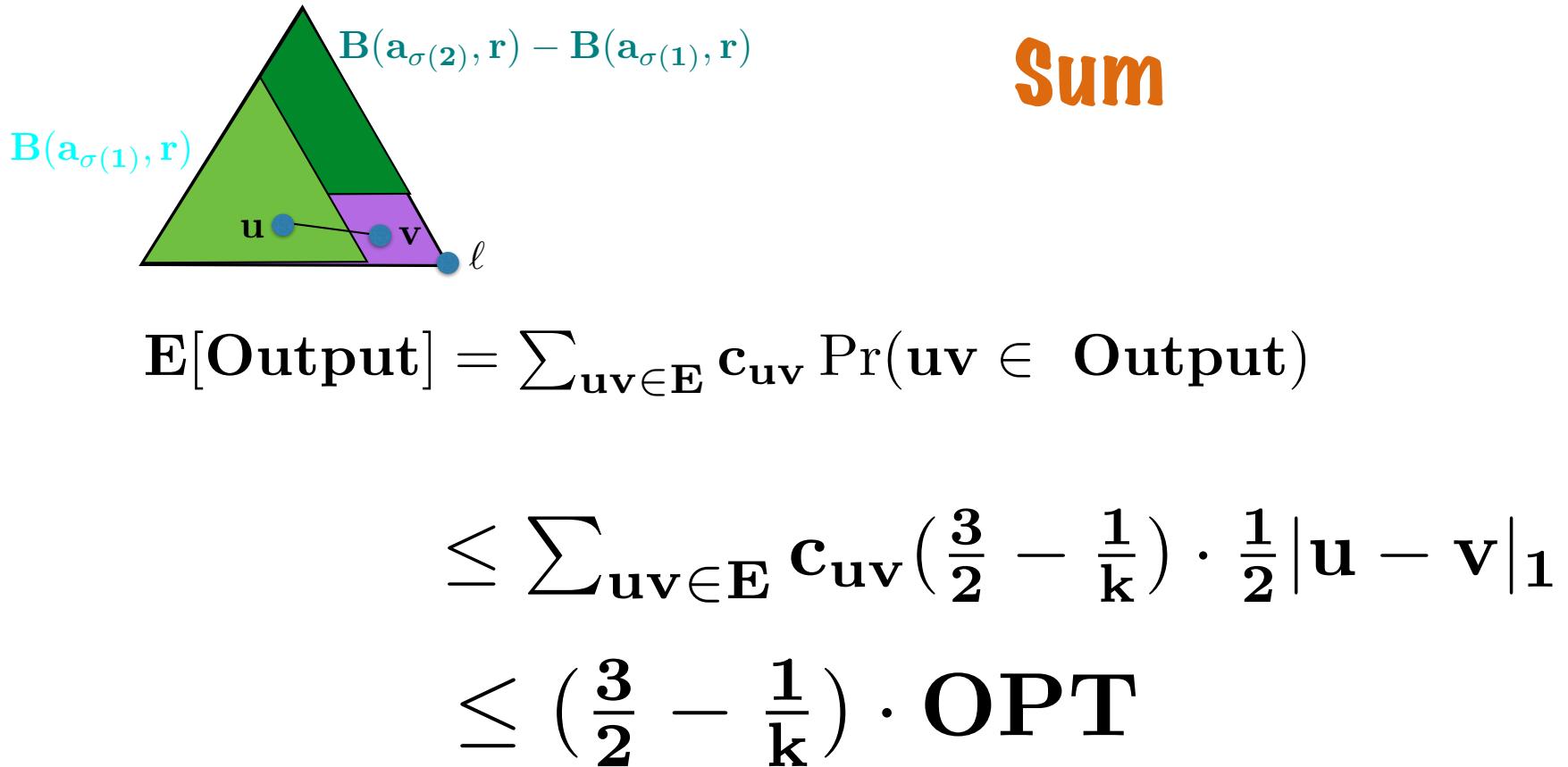
so

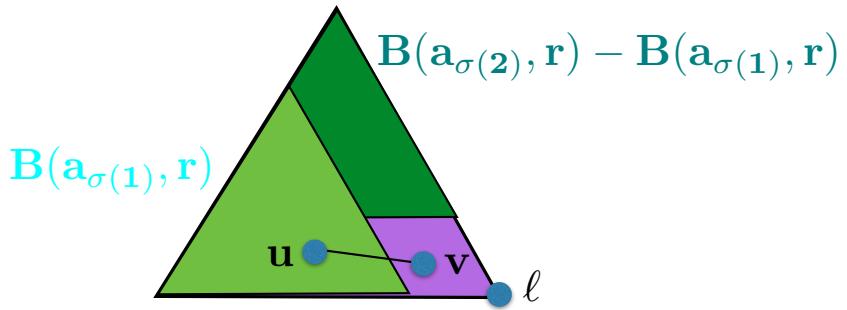
$$|\mathbf{u}_\ell - \mathbf{v}_\ell| \leq \frac{1}{2} \sum_{\mathbf{i}} |\mathbf{u}_i - \mathbf{v}_i|$$



Together

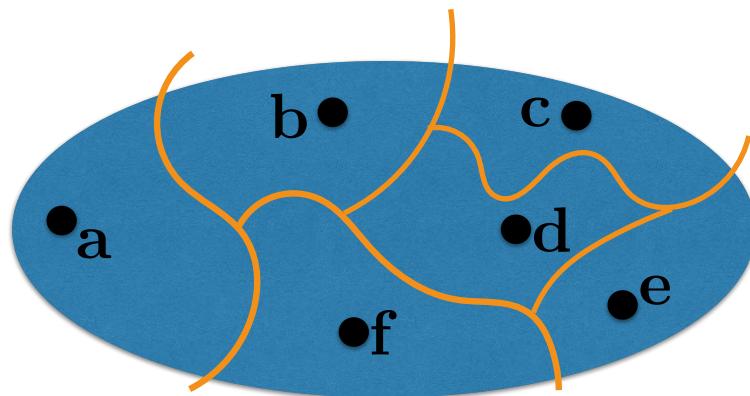
$$\begin{aligned}
 & \left(\frac{1}{2} - \frac{1}{k}\right) |\mathbf{u}_\ell - \mathbf{v}_\ell| + \sum_i \frac{1}{2} |\mathbf{u}_i - \mathbf{v}_i| \\
 & \leq \sum_i \left(\frac{3}{4} - \frac{1}{2k}\right) |\mathbf{u}_i - \mathbf{v}_i| \\
 & = \left(\frac{3}{2} - \frac{1}{k}\right) \cdot \frac{1}{2} |\mathbf{u} - \mathbf{v}|_1
 \end{aligned}$$



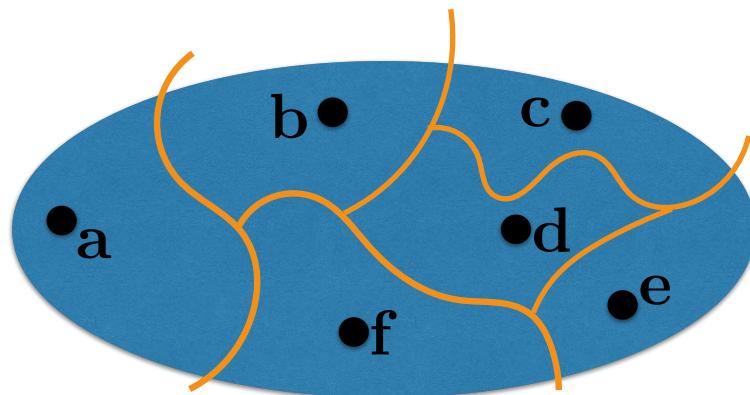


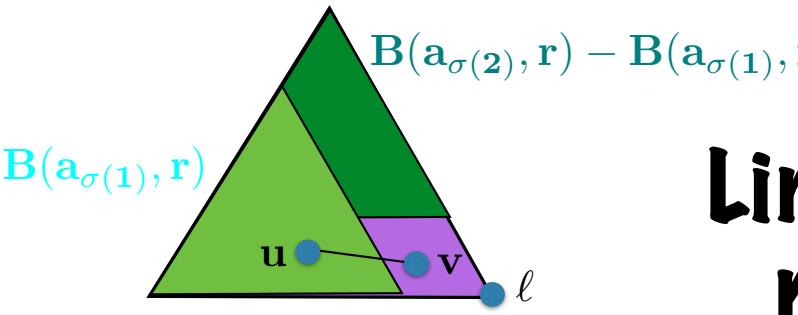
Linear programming and
randomized rounding
give a $3/2 - 1/k$
approximation
for multicut

Multiway cut, linear programming and randomized rounding



Multiway cut, linear programming and randomized rounding





Linear programming and
randomized rounding
give a $3/2 - 1/k$
approximation
for multicut

Can we do better?

k=3: 12/11
by using LP to find the rounding!

APX-hard: cannot get $1 + \epsilon$

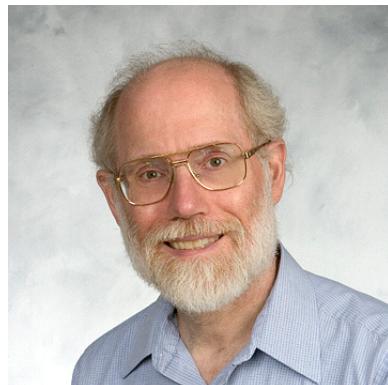
The story behind the story

Applications:

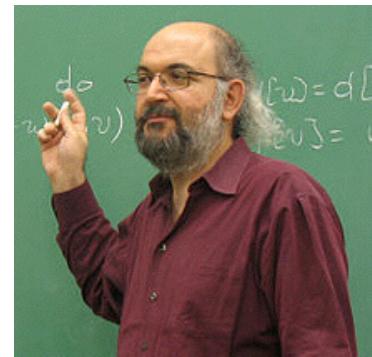
- “minimization of communication costs in parallel computing systems...”
- assigning program modules to processors ...
- partitioning files among the nodes of a network...
- assigning users to base computers in a multicomputer environment...
- partitioning the elements of a circuit into the subcircuits that will go on different chips”



Elias Dalhaus



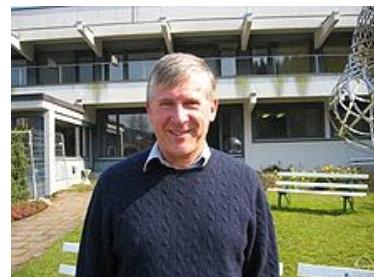
David Johnson



**Mihalis
Yannakakis**



Christos Papadimitriou



Paul Seymour

**APX hardness
approx with min cuts**



Gruia Calinescu



Howard Karloff

Geometric embedding

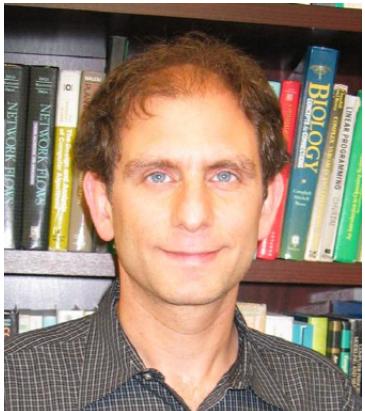
$$3/2 - 1/k$$



Yuval Rabani



David Karger



Cliff Stein



Philip Klein



Neal Young



Mikkel Thorup

**12/11
rounding by
linear programming**

Techniques

rounding input

linear programming relaxation

randomized rounding

probabilistic analysis techniques

geometric interpretation

Problems

Vertex cover

Knapsack

Bin packing

Set cover

Multiway cut

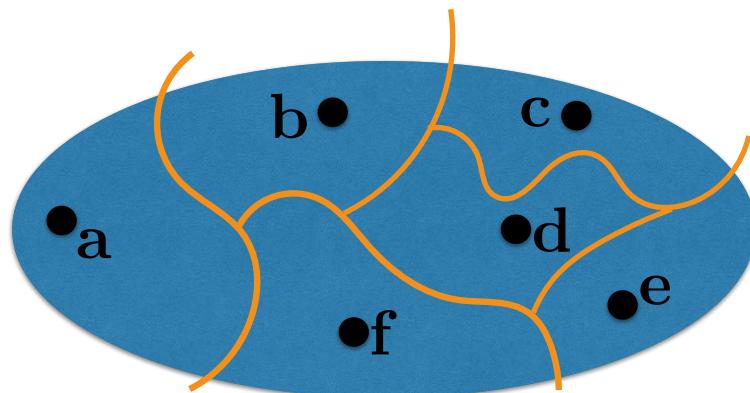
Approximation algorithms, Part II

LP duality

primal dual algorithms

semi-definite programming

Multiway cut, linear programming and randomized rounding



Exercise 5 Solutions

Input: A graph $G = (V, E)$ and k terminals.

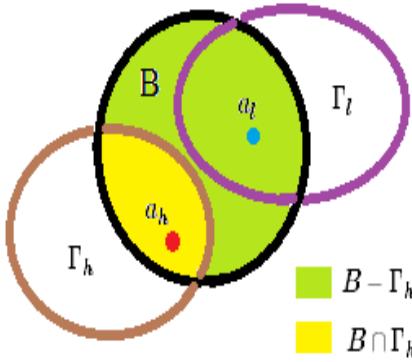
Output: A set of edges Output that separates the k terminals.

1. Output $\leftarrow \emptyset$
2. $\mathcal{P} \leftarrow \{G\}$
3. While \mathcal{P} contains less than k elements:
 - (a) For each element G_i of \mathcal{P} that contains more than 1 terminal:
 - i. $C_i \leftarrow$ minimum cut that separates G_i into G_i^1 and G_i^2 such that both G_i^1 and G_i^2 contains at least one terminal.
 - (b) $a \leftarrow \arg \min_i \text{value}(C_i)$
 - (c) Output \leftarrow Output $\cup C_a$
 - (d) $\mathcal{P} \leftarrow \mathcal{P} \setminus \{G_a\}$
 - (e) $\mathcal{P} \leftarrow \mathcal{P} \cup \{G_a^1, G_a^2\}$
4. Return Output

Approximation ratio.

- We define P_0, \dots, P_{k-1} to be the partitions of the vertices defined by the set P at the different stage of the algorithm. $P_0 = V$ and for any $i > 0$, P_i is the partition of the vertices defined by P at the i^{th} iteration of the while loop.
- We denote by $w(P_i)$ the sum of the weights of the edges that connects vertices lying in different parts of the partition induced by P_i .
- Moreover, for any partition Γ of the vertices of G , we say that the partition is *valid* if each part of the partition contains at least one terminal.
- For any part $\Gamma_j \in \Gamma$, we define $w(\Gamma_j)$ as the sum of the weights of the edges having exactly one endpoint in Γ_j .
- We aim at showing the following Lemma:
- Lemma 1. For any $i \leq k$, for any partition $\Gamma = \{\Gamma_1, \dots, \Gamma_i\}$ with i parts, $w(P_{i-1}) \leq \sum_{j=1}^{i-1} w(\Gamma_j)$.
- Question 1. Argue that the case $i = 1$ holds.
 - For $i = 1$, $w(P_{i-1}) = w(P_0) = w(V) = 0 = \sum_{j=1}^0 w(\Gamma_j)$, since P_0 (having only one part and hence no cut) does not have any edge crossing the cut. Hence the **induction basis** holds.
- We now want to show that the statement is true for any i . We assume that it holds up to $i - 1$. We consider an arbitrary valid partition $\Gamma = \{\Gamma_1, \dots, \Gamma_i\}$ with i parts.
- Question 2. Consider the partition P_{i-2} . How many parts does it contain?
 - P_{i-2} (where $i \geq 2$) is obtained by executing the first $i - 2$ iterations of the loop, each loop creates one additional part and initially the graph G had 1 part (which was itself).
 - Hence, P_{i-2} contains $1 + i - 2 = i - 1$ parts.

- Question 3. Argue that there exist two terminals which belong to the same part B of P_{i-2} but to two different parts Γ_h, Γ_l of Γ .
 - The partition Γ is valid \Rightarrow each part of Γ shall contain at least one terminal (by definition).
 - Γ contains i parts $\Rightarrow \Gamma$ contains at least i terminals, each part containing at least 1 terminal.
 - Let the i terminals be a_1, a_2, \dots, a_i so that $a_j \in \Gamma_j, \forall j \in \{1, 2, \dots, i\}$, w.l.o.g.
 - Also, P_{i-2} contains $i - 1$ parts (by question 2).
 - Now, consider the i terminals a_1, a_2, \dots, a_i from Γ to be placed in $i - 1$ parts of P_{i-2} .
 - By pigeon hole principle, \exists a part $B \in P_{i-2}$ that contains at least 2 of these i terminals, say a_h and a_l for some $h, l \in \{1, \dots, i\}$ s.t., $a_h, a_l \in B \wedge h \neq l$ (w.l.o.g.)
 - But, $a_h \in \Gamma_h$ and $a_l \in \Gamma_l$ (they belong to two different parts of Γ by assumption), which completes the proof.
- Question 4. Compare the cost of splitting B into $B \cap \Gamma_h$ and $B - \Gamma_h$ to the difference $w(P_{i-1}) - w(P_{i-2})$



- $a_h \in \Gamma_h \wedge a_h \in B \Rightarrow a_h \in B \cap \Gamma_h$, from previous results.
- Also, $a_l \in \Gamma_l \wedge a_l \in B \Rightarrow a_l \in B \cap \Gamma_l \Rightarrow a_l \in B - \Gamma_h$, since $\Gamma_h \cap \Gamma_l = \emptyset$, both being (disjoint) parts of the partition Γ .
- Again, $(B \cap \Gamma_h) \cup (B - \Gamma_h) = B \wedge (B \cap \Gamma_h) \cap (B - \Gamma_h) = \emptyset \Rightarrow B \cap \Gamma_h$ and $B - \Gamma_h$ form 2 parts that B can be partitioned into (the edges crossing the cut will not necessarily be the mincut).
- Since $B \in P_{i-2}$ and it contains 2 terminal elements a_h and a_l , on the $(i - 1)^{th}$ iteration, the algorithm will select $G_i = B$ and compute a minimum cut C_i that separates B into 2 parts B^1 and B^2 , when the line 3.(a) gets executed.
- By line 3.(b) of the algorithm, the size of the minimum cut chosen will be the smallest of all such minimum cuts for all available elements G_i of P containing more than 1 terminal element.
- By lines 3.(c) – (e) of the algorithm, the value of $w(P_{i-1}) - w(P_{i-2})$ will be size of the smallest of all such minimum cuts $\Rightarrow w(P_{i-1}) - w(P_{i-2}) \leq$ the cost of splitting B into $B \cap \Gamma_h$ and $B - \Gamma_h$.

- Question 5. Conclude the proof of the Lemma using Question 4 and the induction hypothesis.
 - By the induction hypothesis, we have, $w(P_{i-2}) \leq \sum_{j=1}^{i-2} w(\Gamma_j)$
 - By the Lemma (question 4), we have, $w(P_{i-1}) \leq w(P_{i-2}) +$ the cost of splitting B into $B \cap \Gamma_h$ and $B - \Gamma_h$.
 - Combining the above two, we have, $w(P_{i-1}) \leq \sum_{j=1}^{i-2} w(\Gamma_j) +$ the cost of splitting B into $B \cap \Gamma_h$ and $B - \Gamma_h$.
 - Now from question 4, we can see that $B \cap \Gamma_h$ and $B - \Gamma_h$ is an arbitrary partition of B
 - Moreover, the partition of B into the parts $B \cap \Gamma_h$ and $B - \Gamma_h$ is a valid partition, since each of them contains at least one terminal element, a_h and a_l respectively (by definition of valid partition).
 - Hence, w.l.o.g. we can choose $T_{i-1} = B$ and by definition $w(T_{i-1}) = w((B \cap \Gamma_h) \cup (B - \Gamma_h)) =$ the cost of splitting B into $B \cap \Gamma_h$ and $B - \Gamma_h$, since the edges that have exactly one endpoint in T_{i-1} are exactly those edges that appear in the cut.
 - Hence, we have, $w(P_{i-1}) \leq \sum_{j=1}^{i-2} w(\Gamma_j) + w(T_{i-1})$
 - OR, $w(P_{i-1}) \leq \sum_{j=1}^{i-1} w(\Gamma_j)$ (proved)
- Consider an optimal solution for the problem, namely a partition $F = \{F_1, F_2, \dots, F_k\}$ of V into k parts that all contain exactly one terminal, where the ordering is such that the weight of edges leaving F_k is at least the weight of edges leaving any other set in the partition.

- Question 6. What is the approximation ratio? Conclude using the lectures and Lemma 1.
 - First we notice that the cost of the algorithm is $= \text{cost}(\text{Output}) = w(P_{k-1})$ since P_{k-1} contains k parts.
 - Using the Lemma 1, for OPT, we have, $w(P_{i-1}) \leq \sum_{j=1}^{i-1} w(F_j)$, for any $i \leq k$, since F is a valid partition, each part containing one terminal (by definition).
 - Hence, $w(P_{k-1}) \leq \sum_{j=1}^{k-1} w(F_j)$.
 - Also, given, for OPT, $w(F_1) \leq w(F_2) \dots \leq w(F_{k-1}) \leq w(F_k)$.
 - $\Rightarrow \sum_{j=1}^{k-1} w(F_j) \leq (k-1).w(F_k) = \left(\frac{1-\frac{1}{k}}{\frac{1}{k}}\right).w(F_k)$, (we have $k \geq 2$).
 - $\Rightarrow \left(\frac{1}{k}\right).\sum_{j=1}^{k-1} w(F_j) \leq \left(1 - \frac{1}{k}\right).w(F_k)$.
 - $\Rightarrow (1 - (1 - \frac{1}{k}))\sum_{j=1}^{k-1} w(F_j) \leq \left(1 - \frac{1}{k}\right).w(F_k)$.
 - $\Rightarrow \sum_{j=1}^{k-1} w(F_j) - (1 - \frac{1}{k}).\sum_{j=1}^{k-1} w(F_j) \leq \left(1 - \frac{1}{k}\right).w(F_k)$.
 - $\Rightarrow \sum_{j=1}^{k-1} w(F_j) \leq \left(1 - \frac{1}{k}\right).\sum_{j=1}^{k-1} w(F_j) + (1 - \frac{1}{k}).w(F_k) = \left(1 - \frac{1}{k}\right).\left(\sum_{j=1}^{k-1} w(F_j) + w(F_k)\right)$.
 - $\Rightarrow \sum_{j=1}^{k-1} w(F_j) \leq \left(1 - \frac{1}{k}\right).\sum_{j=1}^k w(F_j)$.
 - Also, from lecture, we have $2 \times \text{OPT} = \sum_{j=1}^k w(F_j)$, since every edge separating some terminal from the others is counted twice.
 - Combining, we have, $\text{cost}(\text{Output}) = w(P_{k-1}) \leq \sum_{j=1}^{k-1} w(F_j) \leq \left(1 - \frac{1}{k}\right).\sum_{j=1}^k w(F_j) = \left(1 - \frac{1}{k}\right).(2 \times \text{OPT})$
 - $\Rightarrow \text{cost}(\text{Output}) \leq 2\left(1 - \frac{1}{k}\right).\text{OPT}$.
 - $\Rightarrow \text{Approximation ratio} = \frac{\text{cost}(\text{Output})}{\text{OPT}} = 2\left(1 - \frac{1}{k}\right)$.

Correctness.

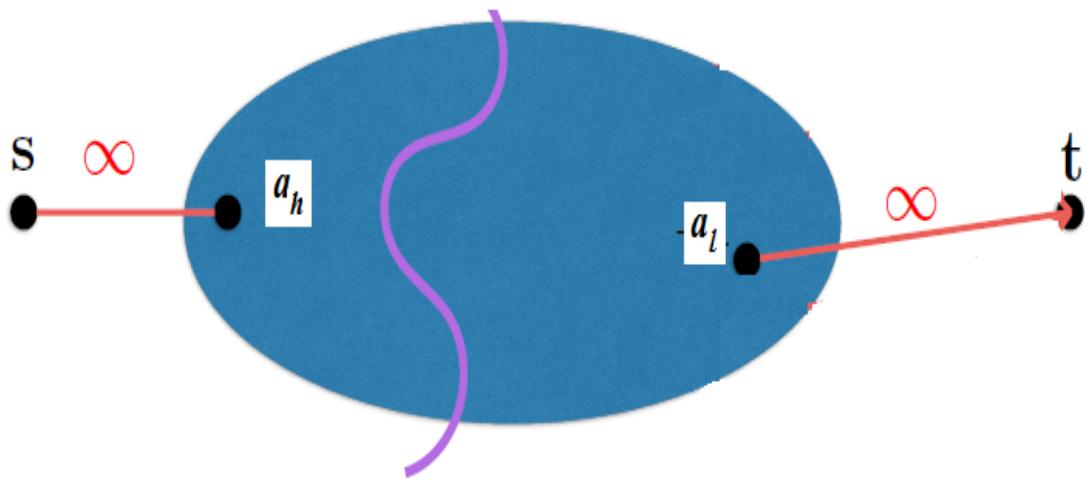
- Question 7. Argue that the output is a solution to the multiway cut problem.
 - Claim: At the end of the iteration $i = k - 1$, the algorithm Output is a partition of G with k parts, each part containing exactly 1 terminal.
 - Let's first prove the following lemma by induction on the number of iterations i .
 - Lemma 2: At the end of the iteration i of the while loop steps 3.(a – e), the algorithm Output is a collection of $i + 1$ vertex-disjoint elements, with each element containing at least one terminal.
 - Basis ($i = 1$): Initially the Output is empty and P contains exactly 1 element, namely the entire graph G .
 - Steps 3.(a) – (e) choose the mincut with the smallest value that separates G into 2 vertex-disjoint elements G_a^1 and G_a^2 , with $G_a^1 \cap G_a^2 = \emptyset$ ensured by the mincut algorithm, each containing at least one terminal.
 - Hence, the base case trivially holds. Output is a collection of $2 = 1 + 1$ elements at the end of the iteration $i = 1$.
 - Induction hypothesis: let the lemma hold for all iterations $\leq i - 1 \Rightarrow$ at the end of the iteration $i - 1$, the algorithm Output is a collection of i vertex-disjoint elements of G , with each element containing at least one terminal.
 - Induction Step: Let's prove the lemma holds for i as well, then we are done.
 - On the i^{th} iteration, the step 3.(a). (i) chooses an element in G that contains more than 1 terminal and uses a minimum cut to separate the element into 2 vertex-disjoint elements, each element containing at least one terminal. Next step 3.(b) chooses the smallest of all such mincuts.
 - Hence, in this iteration, one particular element with at least 2 terminals in G is chosen and split into 2 vertex-disjoint elements, thereby increasing the number of elements in G by one, number of elements (the algorithm output) becomes $i + 1$, by induction hypothesis.
 - Also, the induction hypothesis and the splitting by mincut at iteration i ensures that each of the $i + 1$ elements contain at least one terminal, at the end of iteration i .
 - Hence, the lemma 2 holds $\forall i \in \{1 \dots k - 1\} \Rightarrow$ at the end of the iteration $i = k - 1$, the algorithm Output is a collections of $k - 1 + 1 = k$ vertex-disjoint elements, with each element containing at least one terminal.
 - But, the graph G contains exactly k terminals \Rightarrow each element in the output must contain exactly one terminal (no element can contain more than 1 terminal, otherwise the total number of terminals will exceed k) at the end of the iteration $i = k - 1$.
 - The elements are also vertex-disjoint (ensured by mincuts), which implies that at the end of the iteration $i = k - 1$, the algorithm Output is a partition of G with k parts, each part containing exactly 1 terminal, proving the claim.

Complexity.

- Question 8. Assuming that a minimum cut between two vertices of a graph can be computed in time T , give an upper bound on the complexity for performing line 3.a.i?

Mincut(s,t)

G_i



- As can be seen from the above figure (borrowed from the lecture), if the mincut for an element G_i (that contains more than 1 terminal) can be computed in time T , the complexity of the line 3.a.i will also be $O(T)$, once 2 of the terminals a_h and a_l have been identified.
- In order to find 2 terminals a_h and a_l in G_i , we need $O(|G_i|)$ time, where $|G_i|$ = number of vertices in G_i .
- Combined time complexity for 3.a.i is $O(T + |G_i|) = O(T + |V|)$, for a given i .

- Question 9. What is the maximum cardinality of P ?

- The maximum cardinality of P is k , it's achieved when the algorithm partitions the graph into k parts and the while loop terminates.

- Question 10. What is the overall complexity of the algorithm?

- Lines 1, 2, 4 has complexity $O(1)$.
- While loop at line 3 will be executed $k - 1$ times.
- Now let's consider the worst case time complexity of 3.(a) – (e) for just one single iteration of the while loop 3 first.
 - At any of the iteration, number of elements G_i of P with more than 1 terminal will always be $\leq \frac{k}{2}$, since there are exactly k terminals in G . Hence 3.(a) will at most execute for $\frac{k}{2}$ times for all while loop iterations. (Since G_i contains at least 2 terminals)
 - The total time taken to execute 3.(a) will be $O(\sum_i (T + |G_i|)) = O(\frac{k}{2} \cdot T) + O(\sum_i |G_i|) = O(kT + |V|)$.
 - 3.(b) is $O(k)$.
 - 3.(c) is $O(1)$.
 - 3.(d) and 3.(e) are $O(|E|)$.
- Hence, the total time complexity for all the $O(k)$ iterations of the while loop is $O(k \cdot (kT + |V| + k + |E|)) = O(k^2 T + k(|V| + |E|))$.
- Hence, the overall complexity is $O(k^2 T + k(|V| + |E|))$, which is polynomial in V, E , since T is also polynomial in V, E .
- Treating k as a constant, the overall complexity is $O(T + |V| + |E|)$.

Peer-graded Assignment 5

Clark Grubb

January 10, 2016

Input: A graph $G = (V, E)$ and k terminals.
Output: A set of edges Output that separates the k terminals.

1. $\text{Output} \leftarrow \emptyset$
2. $\mathcal{P} \leftarrow \{G\}$
3. While \mathcal{P} contains less than k elements:
 - (a) For each element G_i of \mathcal{P} that contains more than 1 terminal:
 - i. $C_i \leftarrow$ minimum cut that separates G_i into G_i^1 and G_i^2 such that both G_i^1 and G_i^2 contains at least one terminal.
 - (b) $a \leftarrow \arg \min_i \text{value}(C_i)$
 - (c) $\text{Output} \leftarrow \text{Output} \cup C_a$
 - (d) $\mathcal{P} \leftarrow \mathcal{P} \setminus \{G_a\}$
 - (e) $\mathcal{P} \leftarrow \mathcal{P} \cup \{G_a^1, G_a^2\}$
4. Return Output

Approximation ratio.

We define P_0, \dots, P_{k-1} to be the partitions of the vertices defined by the set \mathcal{P} at the different stage of the algorithm. $P_0 = V$ and for any $i > 0$, P_i is the partition of the vertices defined by \mathcal{P} at the i th iteration of the while loop.

We denote by $w(P_i)$ the sum of the weights of the edges that connects vertices lying in different parts of the partition induced by P_i .

Moreover, for any partition Γ of the vertices of G , we say that the partition is valid if each part of the partition contains at least one terminal.

For any part $\Gamma_i \in \Gamma$, we define $w(\Gamma_j)$ as the sum of the weights of the edges having exactly one endpoint in Γ_j .

We aim at showing the following Lemma:

Lemma 1. For any $i \leq k$, for any partition $\Gamma = \{\Gamma_1, \dots, \Gamma_i\}$ with i parts,

$$w(P_{i-1}) \leq \sum_{j=1}^{i-1} w(\Gamma_j)$$

Q1. Argue that the case $i = 1$ holds.

A1. $w(P_0) = 0$, since there is a single part in the partition and no edges cross a boundary. The summation is over the empty set and is also zero.

We now want to show that the statement is true for any i . We assume that it holds up to $i - 1$. We consider an arbitrary valid partition $\Gamma = \{\Gamma_1, \dots, \Gamma_i\}$ with i parts.

Q2. Consider the partition P_{i-2} . How many parts does it contain?

A2. If there is only one part, then there is no P_{i-2} . Otherwise the iteration replaced one part with two, so the number of parts in P_{i-2} is $i - 1$.

Q3. Argue that there exist two terminals which belong to the same part B of P_{i-2} but to two different parts Γ_h, Γ_l of Γ .

A3. The iteration of the algorithm which created P_{i-1} from P_{i-2} split one of the parts of P_{i-2} . Let G_j be this part. According to the algorithm, it was split into two parts G_j^1 and G_j^2 , each containing at least one terminal. Then B is the vertices of G_j , Γ_h is the vertices of G_j^1 , and Γ_l is the vertices of G_j^2 . B is the union of the disjoint sets Γ_h and Γ_l and contains at least two terminals.

Q4. Compare the cost of splitting B into $B \cap \Gamma_h$ and $B \setminus \Gamma_h$ to the difference $w(P_{i-1}) - w(P_{i-2})$.

A4. The difference of $w(P_{i-1}) - w(P_{i-2})$ is the weight of the added edges, all of which connect vertices in Γ_h to vertices in Γ_l . Because we count each of these edges twice, once in $w(\Gamma_h)$ and again in $w(\Gamma_l)$, it follows that:

$$w(\Gamma_h) + w(\Gamma_l) - w(B) = 2(w(P_{i-1}) - w(P_{i-2}))$$

Q5. Conclude the proof of the Lemma using Question 4 and the induction hypothesis.

A5. Let $\{\Gamma_1, \dots, \Gamma_{i-1}\}$ be P_{i-1} . Let $B = \Gamma_{i-2} \cup \Gamma_{i-1}$. Observe that the result demonstrated in **A4.** applies here with Γ_{i-2} and Γ_{i-1} replacing Γ_h and Γ_l , even if B is not the part that the algorithm would select to split. Thus

$$\begin{aligned} w(P_{i-1}) &= w(P_{i-1}) - w(P_{i-2}) + w(P_{i-2}) \\ &= \frac{w(\Gamma_{i-2}) + w(\Gamma_{i-1}) - w(B)}{2} + w(P_{i-2}) \\ &\leq \frac{w(\Gamma_{i-2}) + w(\Gamma_{i-1}) - w(B)}{2} + \sum_{j=1}^{i-3} w(\Gamma_j) + w(B) \\ &\leq w(\Gamma_{i-2}) + w(\Gamma_{i-1}) - w(B) + \sum_{j=1}^{i-2} w(\Gamma_j) + w(B) \\ &= \sum_{j=1}^{i-1} w(\Gamma_j) \end{aligned}$$

Consider an optimal solution for the problem, namely a partition $F = \{F_1, F_2, \dots, F_k\}$ of V into k parts that all contain exactly one terminal, where the ordering is such that the weight of edges leaving F_k is at least the weight of edges leaving any other set in the partition.

Q6. What is the approximation ratio? Conclude using the lectures and Lemma 1.

A6. Recall that

$$OPT = \frac{1}{2} \sum_{j=1}^k w(\Gamma_j)$$

The cost of the partition discovered by the algorithm is

$$\begin{aligned} w(P_{k-1}) &\leq \sum_{j=1}^{k-1} w(\Gamma_j) \\ &\leq \frac{k-1}{k} \sum_{j=1}^k w(\Gamma_j) \\ &= 2\left(1 - \frac{1}{k}\right) OPT \end{aligned}$$

The algorithm yields a solution no more than $2(1 - \frac{1}{k})$ as expensive as the optimal solution.

Correctness.

Q7. Argue that the output is a solution to the multiway cut problem.

A7. We know that the algorithm terminates after at most $k - 1$ iterations of the loop. By the termination condition of the while loop, we know that there are at least k parts in the partition. However, since each part contains a terminal and there are k terminals, there must be exactly k parts. Since each part contains a terminal, the partition is a solution to the multiway cut problem.

Complexity.

Q8. Assuming that a minimum cut between two vertices of a graph can be computed in time T , give an upper bound on the complexity for performing line 3.a.i?

A8. There are $\binom{k}{2}$ terminals, so we never need to look for more than that many minimum cuts in an iteration.

Q9. What is the maximum cardinality of \mathcal{P} ?

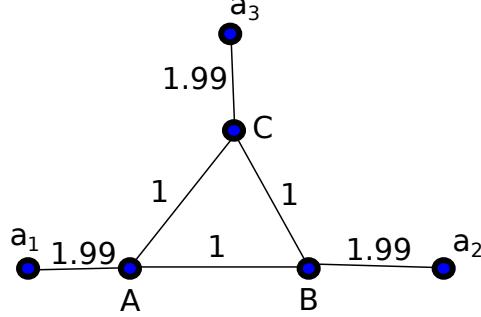
A9. Every part must contain a terminal, so $|\mathcal{P}| \leq k$.

Q10. What is the overall complexity of the algorithm?

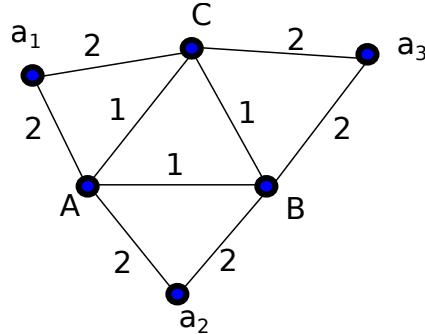
A10. There are $k - 1$ iterations, and $\binom{k}{2}$ pairs of terminals, so we wouldn't need to look for minimum cuts more than $\frac{k(k-1)^2}{2}$ times. The reasoning does not prove that the bound is tight, but the number of minimum cuts that must be searched for can be $O(k^3)$ in some cases. Consider a case where after each iteration, the largest part of the partition has $k - j$ terminals in it where j is the iteration number, and all the other parts have a single terminal. Then the number of minimum cuts we search for is:

$$\sum_{j=2}^k \binom{j}{2} = \frac{(k-1)k(k+1)}{6} = O(k^3)$$

1. Consider the simple algorithm described at the first lecture of session 5. Consider the following graph, what is the cut output by the algorithm? What is the optimal cut? Conclude.



2. Consider the linear program relaxation described during the lectures. Consider the following graph, what is the value of the fractional optimal solution? What is the optimal cut? Conclude.



3. Consider the linear program for multiway cut described during the lectures and the distances defined by a fractional optimal solution. We describe a new rounding procedure:

- (a) $\text{Output} \leftarrow \emptyset$.
- (b) Pick at random $r \in [0, 1/2]$.
- (c) For each edge (u, v) , add (u, v) to Output if there exists a terminal a_i such that $d(u, a_i) \leq r \leq d(v, a_i)$.

Show that the cut that is output by the algorithm is of cost at most 2OPT.

4. We present a different relaxation for the multiway cut problem. Consider an undirected graph $G = (V, E)$ and a set of terminal $\{a_1, \dots, a_k\}$. Compute the following directed graph $H = (V, E')$ where $E' = \{\langle u, v \rangle \mid (u, v) \in E\}$. Namely for each edge $(u, v) \in E$, there are two arcs $\langle u, v \rangle$ and $\langle v, u \rangle$ in E' . Additionally, each arc $\langle u, v \rangle$ has the same weight

$w_{(u,v)}$ as the edge (u, v) . Let \mathcal{P} be the collection of all the directed paths starting at a terminal a_i and ending at a terminal a_j with $i < j$.

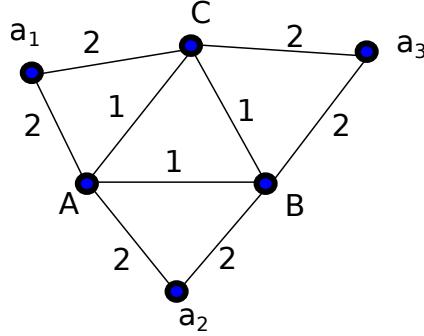
We now define the linear program. Each variable d_e correspond to an arc of E' .

$$\min \sum_{e \in E'} d_e w_e$$

subject to:

$$\begin{aligned} \sum_{e \in P} d_e &\geq 1, \forall P \in \mathcal{P} \\ d_e &\in \{0, 1\} \end{aligned}$$

- (a) Argue that any (integral) solution to this linear program can be converted into a solution to the multiway cut problem of same cost.
- (b) Argue that any multiway cut of G can be converted into a solution to this linear program of same cost.
- (c) Consider the following graph. What is the value of the optimal fractional solution on this graph? What is the value of the optimal multiway cut? Conclude.



5. We define a slightly different way of rounding the fractional optimal solution of the LP described during the lecture.

- (a) Pick at random $r \in (0, 1)$.
- (b) Pick at random σ in the two following permutations: $(a_1, a_2, \dots, a_{k-1}, a_k)$ and $(a_{k-1}, a_{k-2}, \dots, a_1, a_k)$.
- (c) For each i in $1, \dots, k-1$:
 - i. Add each vertex v such that $x_{v,\sigma(i)} \geq r$ to the cluster of $a_{\sigma(i)}$.
- (d) Add the remaining vertices to the cluster of $a_{\sigma(k)}$.
- (e) Output \leftarrow the edges between the vertices of different clusters.

We aim at showing that this rounding procedure provides the same approximation guarantee than the one described during the lectures.



([https://accounts.coursera.org/i/zendesk/courserahelp?
return_to=https://learner.coursera.help/hc](https://accounts.coursera.org/i/zendesk/courserahelp?return_to=https://learner.coursera.help/hc))

Congratulations! [View Final Grade](#)

i You've completed the course. (/learn/approximation-algorithms-part-1/home/welcome)

Multiway Cut and Randomized Rounding



Claire Mathieu

This module deepens the understanding of randomized rounding by developing a sophisticated variant and applying it to another basic problem, the Multiway Cut problem. (This is a more

What is the Multiway Cut problem?

● Lecture: definition 14 min

(/learn/approximation-algorithms-part-1/lecture/kleLz/lecture-definition)

● Slides

(/learn/approximation-algorithms-part-1/supplement/Jf0oD/slides)

Help Center

Practice Quiz: Quiz 1 :

● Some context on cuts 2 questions

(/learn/approximation-algorithms-part-1/quiz/4jwNy/quiz-1-some-context-on-cuts)

A linear programming relaxation

Lecture: linear

 programming
relaxation 20 min

(/learn/approximation-algorithms-part-1/lecture/ce4ya/lecture-linear-programming-relaxation)

 Slides

(/learn/approximation-algorithms-part-1/supplement/6Smjw/slides)

 Practice Quiz: Quiz
2 3 questions

(/learn/approximation-algorithms-part-1/quiz/KvC1N/quiz-2)

Randomized rounding

 Lecture: randomized
rounding 14 min

(/learn/approximation-algorithms-part-1/lecture/c7FEk/lecture-randomized-rounding)

 Slides

(/learn/approximation-algorithms-part-1/supplement/ZV1sS/slides)

 Practice Quiz: Quiz
3 2 questions

(/learn/approximation-algorithms-part-1/quiz/tTtVe/quiz-3)

Analysis

 **Lecture: analysis** 15 min

(/learn/approximation-algorithms-part-1/lecture/lj0Zx/lecture-analysis)

 **Slides**

(/learn/approximation-algorithms-part-1/supplement/MgMgx/slides)

 **Practice Quiz: Quiz**

4 2 questions

(/learn/approximation-algorithms-part-1/quiz/sWQcd/quiz-4)

Conclusion

 **Lecture: conclusion** 7 min

(/learn/approximation-algorithms-part-1/lecture/JIsU8/lecture-conclusion)

 **Slides**

(/learn/approximation-algorithms-part-1/supplement/8ztfq/slides)

 **Practice Quiz: Quiz**

5 2 questions

(/learn/approximation-algorithms-part-1/quiz/jkMPq/quiz-5)

Exercises

 **Practice exercise**

(/learn/approximation-algorithms-part-1/supplement/HfSTi/practice-exercise)

 **Assignment: Peer-graded assignment 5** 30 min

(/learn/approximation-algorithms-part-1/peer/njgfn/peer-graded-assignment-5)

Review Classmates: Peer-

✓ graded assignment

5 30 min

(/learn/approximation-algorithms-part-1/peer/njgfn/peer-graded-assignment-5/give-feedback)

All Chapter Slides together
in one file

(/learn/approximation-algorithms-part-1/supplement/KSWYv/all-chapter-slides-together-in-one-file)

Slides for all chapters of
Approx Algs Part 1 together
in one file

(/learn/approximation-algorithms-part-1/supplement/Ed3TV/slides-for-all-chapters-of-approx-algs-part-1-together-in-one-file)

Input: A graph $G = (V, E)$ and k terminals.

Output: A set of edges Output that separates the k terminals.

1. Output $\leftarrow \emptyset$

2. $\mathcal{P} \leftarrow \{G\}$

3. While \mathcal{P} contains less than k elements:

(a) For each element G_i of \mathcal{P} that contains more than 1 terminal:

i. $C_i \leftarrow$ minimum cut that separates G_i into G_i^1 and G_i^2 such that both G_i^1 and G_i^2 contains at least one terminal.

(b) $a \leftarrow \arg \min_i \text{value}(C_i)$

(c) Output \leftarrow Output $\cup C_a$

(d) $\mathcal{P} \leftarrow \mathcal{P} \setminus \{G_a\}$

(e) $\mathcal{P} \leftarrow \mathcal{P} \cup \{G_a^1, G_a^2\}$

4. Return Output

(https://accounts.coursera.org/i/zendesk/courserahelp?return_to=https://learner.coursera.help/hc)

Assignment: Peer-graded assignment 5

Review classmates to see your grade

Your grade is ready, but you have not reviewed enough classmates yet. Review more classmates to see your grade.

[Review Classmates' Work \(/learn/approximation-algorithms-part-1/peer/njgfn/peer-graded-assignment-5/give-feedback\)](#)

Instructions (</learn/approximation-algorithms-part-1/peer/njgfn/peer-graded-assignment-5>)

My submission (</learn/approximation-algorithms-part-1/peer/njgfn/peer-graded-assignment-5/submit>)

Discussions (</learn/approximation-algorithms-part-1/peer/njgfn/peer-graded-assignment-5/discussions>)

Thank you for completing a peer reviewed assignment! Please take this survey to help us improve your experience.

(<https://www.surveymonkey.com/r/Z8KFSCC>)

c=assignmentId%3DQqaGLJrREeW3phIGMvrfMw%4012&c=courseId%3DbnQLDSclEeWbiBJCM9ziNQ&c=itemId%3Dnjgfn&c=submissionId%3Ds69tnrZSEeWARs

Assignment 5 Solutions

January 9, 2016

Shareable Link (<https://www.coursera.org/learn/approximation-algorithms-part-1/peer/njgfn/peer-graded-assignment-5/review/s69tnrZSEeWARsHojuCw>)

Consider the following algorithm.

(<https://www.di.ens.fr/~vcohen/algo.jpg>)
<https://www.di.ens.fr/~vcohen/algo.jpg> (<https://www.di.ens.fr/~vcohen/algo.jpg>)

Approximation ratio.

We define P_0, \dots, P_{k-1} to be the partitions of the vertices defined by the set P at the different stage of the algorithm. $P_0 = V$ and for any $i > 0$, P_i is the partition of the vertices defined by P at the i th iteration of the while loop.

We denote by $w(P_i)$ the sum of the weights of the edges that connect vertices lying in different parts of the partition induced by P_i .

Moreover, for any partition Γ of the vertices of G , we say that the partition is valid if each part of the partition contains at least one terminal.

For any part $\Gamma_j \in \Gamma$, we define $w(\Gamma_j)$ as the sum of the weights of the edges having exactly one endpoint in Γ_j .

We aim at showing the following Lemma:

Lemma 1. For any $i \leq k$, for any partition $\Gamma = \{\Gamma_1, \dots, \Gamma_i\}$ with i parts,

$$w(P_{i-1}) \leq \sum_{j=1}^{i-1} w(\Gamma_j).$$

Question 1. Argue that the case $i = 1$ holds.

We now want to show that the statement is true for any i . We assume that it holds up to $i - 1$. We consider an arbitrary valid partition $\Gamma = \{\Gamma_1, \dots, \Gamma_{i-1}\}$ with i parts.

Question 2. Consider the partition P_{i-2} . How many parts does it contain?

Question 3. Argue that there exist two terminals which belong to the same part B of P_{i-2} but to two different parts Γ_h, Γ_ℓ of Γ .

Question 4. Compare the cost of splitting B into $B \cap \Gamma_h$ and $B \setminus \Gamma_h$ to the difference $w(P_{i-1}) - w(P_{i-2})$.

Question 5. Conclude the proof of the Lemma using Question 4 and the induction hypothesis.

Consider an optimal solution for the problem, namely a partition $F = \{F_1, F_2, \dots, F_k\}$ of V into k parts that all contain exactly one terminal, where the ordering is such that the weight of edges leaving F_k is at least the weight of edges leaving any other set in the partition.

Question 6. What is the approximation ratio? Conclude using the lectures and Lemma 1.

Correctness.

Question 7. Argue that the output is a solution to the multiway cut problem.

Complexity.

Question 8. Assuming that a minimum cut between two vertices of a graph can be computed in time T , give an upper bound on the complexity for performing line 3.a.i?

Question 9. What is the maximum cardinality of P ?

Question 10. What is the overall complexity of the algorithm?

[Assignment 5 Solutions](https://s3.amazonaws.com/coursera-uploads/peer-review/bnQLDSclEeWbiBJCM9ziNQ/d442a725426837fbac64e304af1408aa/exercise.pdf) (<https://s3.amazonaws.com/coursera-uploads/peer-review/bnQLDSclEeWbiBJCM9ziNQ/d442a725426837fbac64e304af1408aa/exercise.pdf>)

Assignment 5 Solutions

An answer to question 1 has to be of the following form:

The case $i = 1$ holds since $P_{i-1} = P_0 = V$.

- 1 pt
Yes

- 0 pts
No

The partition contains exactly $i - 1$ parts.

- 1 pt
Yes

- 0 pts
No

An answer to question 3 has to be of the following form:

Since Γ is valid and contains i parts and P_{i-2} contains $i - 1$ parts, there must be two terminals in the same part of P_{i-2} but in different parts of Γ .

- 2 pts
Yes

- 0 pts
No

An answer to question 4 has to be of the following form:

The algorithm considers splitting B into $B \cap \Gamma_h$ and $B \setminus \Gamma_h$ when creating part P_i . Since it picks the cut of minimum weight it follows that $w(P_{i-1}) - w(P_{i-2}) \leq w(\Gamma_h)$.

- 3 pts
Yes

- 0 pts
No

From the induction hypothesis, we have $w(P_{i-2}) \leq \left(\sum_{j=1}^{i-1} w(\Gamma_j) \right) - w(\Gamma_h)$.

Moreover, $w(P_i - 1) \leq w(P_{i-2}) + w(\Gamma_h)$ and so,

$$w(P_i - 1) \leq \sum_{j=1}^{i-1} w(\Gamma_j).$$

- 3 pts
Yes

- 0 pts

No

An answer to question 6 has to be of the following form:

From the lectures we have $\sum_{j=1}^{k-1} w(F_j) \leq 2(1 - 1/k)\text{OPT}$

The output of the greedy is the partition P_{k-1} and so, by Lemma 1,

$w(P_{k-1}) \leq \sum_{j=1}^{k-1} w(F_j) \leq 2(1 - 1/k)\text{OPT}$.

3 pts

Yes

0 pts

No

An answer to question 7 has to be of the following form:

Since each part contains at least one terminal and since there are at most k parts, the cut output by the algorithm is a solution to the multiway cut problem.

3 pts

Yes

0 pts

No

An answer to question 8 has to be of the following form:

$O(\text{poly}(k)T)$.

1 pt

Yes

0 pts

No

The cardinality of P is at most k .

1 pt

Yes

0 pts

No

An answer to question 10 has to be of the following form:

$\text{poly}(k, n)$, i.e., a polynomial in n and k (e.g., $n^3 + k^4 + 12k^2$).

2 pts

Yes

0 pts

No

Edit submission

Comments

Visible to classmates



share your thoughts...

Question 1

Since we are using w in 2 different senses, I will use \mathbf{w} for the weight of a partition, and w for the weight of a part in a partition.

For $i = 1$ any partition of the vertices V into i parts is the trivial partition $\Gamma = \{\Gamma_1\} = \{V\}$. Note that this partition is valid.

We have $\sum_{j=1}^{i-1} w(\Gamma_j) = \sum_{j=1}^0 w(\Gamma_j) = 0$, since there are no terms in the sum.

We also have $\mathbf{w}(P_{i-1}) = \mathbf{w}(P_0) = \mathbf{w}(\{V\}) = 0$ since all edges have both vertices in the same part $V = \Gamma_1$ of the partition.

Hence $\mathbf{w}(P_0) \leq \sum_{j=1}^0 w(\Gamma_j)$.

Question 2

If we partition the set V into j parts, where $j < k$, then at least one of the parts contains more than one terminal. Otherwise there would be at most j terminals, leading to a contradiction.

The algorithm starts with the partition $P_0 = \{V\}$ that has one part. At each step i of the algorithm one part of the partition is divided into 2 sub-parts, incrementing the number of parts by 1, as long as there is at least one partition containing more than 1 terminal (i.e. as long as $i < k$). So partition P_i has $i+1$ parts for $i < k$.

So P_{i-2} contains $i - 1$ parts for $i < k + 2$.

Note that the algorithm will produce only valid partitions, since a part will be split into 2 parts only if it contains at least 2 terminals, and the split is done so that each of the 2 resulting subsets of the part contains at least 1 terminal.

Question 3

P_{i-2} is made of $i - 2 + 1 = i - 1$ parts, while Γ is made of i parts.

Since Γ is valid, each part $\Gamma_j, j \in \{1, \dots, i\}$ contains at least one terminal. So for each $j \in \{1, \dots, i\}$ we can choose an associated terminal $t_j \in \Gamma_j$, where all the t_j are distinct.

P_{i-2} is a partition of V into $i - 1$ parts, so denote by B_j the part that contains t_j . We are mapping i terminals onto $i - 1$ parts, so there must be some h and l such that t_h and t_l map to the same part $B = B_h = B_l$. These belong to the same part B but different Γ_h and Γ_l .

Question 4

P_{i-1} differs from P_{i-2} in that one of the parts has been split into 2. Let $P_{i-2} = \{B_1, \dots, B_{i-1}\}$ and let B_m denote the part split in the next step of the algorithm, so that $P_{i-1} = \{B_1, \dots, B_{m-1}, S_1, S_2, B_{m+1}, \dots, B_i\}$, where $S = \{S_1, S_2\}$ is a subpartition of B_m , i.e. $B_m = S_1 \cup S_2$ and $S_1 \cap S_2 = \emptyset$.

The cost $\mathbf{w}(P_{i-2})$ is the sum of the weight of all the edges in G which have their vertices in different parts. All these edges also have their vertices in different parts in P_{i-1} . However, all the edges internal to B_m which have one vertex in S_1 and the other in S_2 will also contribute to $\mathbf{w}(P_{i-1})$. We can write this as follows

$$\mathbf{w}(P_{i-1}) = \mathbf{w}(P_{i-2}) + \mathbf{w}(S)$$

where by $\mathbf{w}(S)$ we mean the cost of the partition S of the subgraph of G which has only the vertices B_m and the edges that have both vertices in B_m .

This is true for the split of any B_m into any 2 subparts, and a similar relation holds for the split of B from Q3 into $S' = \{B \cap \Gamma_h, B \setminus \Gamma_h\}$.

Our algorithm considers all splits into valid partitions and chooses m and S so as to get the minimum possible value of $\mathbf{w}(S)$. Therefore

$$\mathbf{w}(P_{i-1}) - \mathbf{w}(P_{i-2}) = \mathbf{w}(S) \leq \mathbf{w}(S') = \mathbf{w}((P_{i-2} \setminus \{B\}) \cup S') - \mathbf{w}(P_{i-2})$$

Question 5

To complete the induction, for $i > 1$ consider any valid partition $\Gamma = \{\Gamma_1, \dots, \Gamma_i\}$ of V . From Q3 we know that there is a part $B \in P_{i-2}$ such that Γ_h and Γ_l both contain terminals from B . Without loss of generality we can assume $h = i - 1$ and $l = i$. Consider the partition Γ' obtained by merging the last 2 parts, so $\Gamma' = \{\Gamma_1, \dots, \Gamma_{i-2}, \Gamma_{i-1} \cup \Gamma_i\} = \{\Gamma'_1, \dots, \Gamma'_{i-1}\}$ where $\Gamma'_j = \Gamma_j, j \in \{1, \dots, i-2\}$ and $\Gamma'_{i-1} = \Gamma_{i-1} \cup \Gamma_i$. This is also a valid partition (the last part contains at least one terminal) with $i - 1$ parts, and by the induction hypothesis we have

$$\mathbf{w}(P_{i-2}) \leq \sum_{j=1}^{i-2} w(\Gamma'_j)$$

So (recall that $S' = \{B \cap \Gamma_{i-1}, B \setminus \Gamma_{i-1}\}$)

$$\begin{aligned}
\mathbf{w}(P_{i-1}) &\leq \mathbf{w}((P_{i-2} \setminus \{B\}) \cup S') \quad \text{recall that } P_{i-1} \text{ is minimal} \\
&\leq \mathbf{w}(P_{i-2}) + \mathbf{w}(S') \\
&\leq \sum_{j=1}^{i-2} w(\Gamma'_j) + \mathbf{w}(S') \quad \text{by the induction hypothesis} \\
&\leq \sum_{j=1}^{i-2} w(\Gamma_j) + \mathbf{w}(S') \\
&\leq \sum_{j=1}^{i-2} w(\Gamma_j) + \mathbf{w}(\{\Gamma_{i-1}, \Gamma_i\}) \\
&\leq \sum_{j=1}^{i-2} w(\Gamma_j) + w(\Gamma_{i-1}) \\
&\leq \sum_{j=1}^{i-1} w(\Gamma_j)
\end{aligned}$$

because $\mathbf{w}(\{\Gamma_{i-1}, \Gamma_i\}) \leq w(\Gamma_{i-1})$ since the edges connecting Γ_{i-1} and Γ_i are also edges that leave Γ_{i-1} .

This completes the induction.

Question 6

Let an optimal solution be given by a valid partition $F = \{F_1, \dots, F_k\}$, where we use a permutation of these parts which gives the last part the highest w value, i.e. $w(F_k) \geq w(F_j)$. We have, since every edge in the optimal cut is contained in 2 of the F_j ,

$$\begin{aligned}
OPT &= \mathbf{w}(F) = \frac{1}{2} \cdot \sum_{j=1}^k w(F_j) \\
2 \cdot OPT &= \sum_{j=1}^k w(F_j)
\end{aligned}$$

The algorithm produces a valid partition P_{k-1} . Our lemma produces a bound for $\mathbf{w}(P_{k-1})$ for any valid partition, so we apply the lemma to our optimal partition, which gives

$$\begin{aligned}
\mathbf{w}(P_{k-1}) &\leq \sum_{j=1}^{k-1} w(F_j) \\
&\leq \left[\sum_{j=1}^k w(F_j) \right] - w(F_k) \\
&\leq \left(1 - \frac{1}{k}\right) \cdot \sum_{j=1}^k w(F_j)
\end{aligned}$$

since $w(F_k) \geq w(F_j)$ and thereby $w(F_k) \geq \frac{1}{k} \sum_{j=1}^k w(F_j)$.

Clearly $OPT \leq \mathbf{w}(P_{k-1})$ and so

$$\begin{aligned}
OPT &\leq \mathbf{w}(P_{k-1}) \\
&\leq \left(1 - \frac{1}{k}\right) \cdot \sum_{j=1}^k w(F_j) \\
&\leq 2 \cdot \left(1 - \frac{1}{k}\right) \cdot OPT
\end{aligned}$$

So we have a $2 \cdot \left(1 - \frac{1}{k}\right)$ approximation.

Question 7

The algorithm performs $k - 1$ steps, at the end of which it has a partition of V into k parts, such that every part contains at least 1 terminal. We start with 1 part and after iteration i we will have $i + 1$ parts, since at each iteration at least one part exists with more than 1 terminal, and one of these is split into 2 new parts so that the partition is still valid. After $k - 1$ iterations we have k parts, and, since there are k terminals, each part has exactly 1 terminal. The output edges perform the cuts between the parts, and the union of the output edges perform the required multi-way cut of the parts, and therefore of the terminals.

Question 8

We run this step of the algorithm when G_i contains $l > 1$ terminals. To determine the minimum cut, we run the minimum cut algorithm for 2 vertices on all combinations of pairs of terminals in G_i so that we cover all possibilities. There are $\frac{l(l-1)}{2}$ possible (unordered) pairs of terminals in G_i , but we do not have

to run the subroutine that often, as each run will cover multiple combinations (when $l > 2$).

Label the l terminals in G_i as $\{t_1, \dots, t_l\}$. In the minimal cut t_1 will be in one of $\{G_i^1, G_i^2\}$, (w.l.o.g.) say G_i^1 . Now G_i^2 also contains a terminal, say t_m where $m \in \{2, \dots, l\}$. If we run the 2 vertex mincut on the pair (t_1, t_m) , then we will get the minimal sub-partition. So we can compute the minimal cut separating 2 terminals in $(l - 1) \cdot T$ steps by running each of the combinations $(t_1, t_2), \dots, (t_1, t_l)$. This is a worst case only, as each run of the subroutine is likely to cover multiple terminals. I.e. if the run for (t_1, t_m) has t_1 assigned to G_i^1 , and $t_{h_1}, t_{h_2}, \dots, t_{h_n}$ in G_i^2 , then we have covered all the combinations $(t_1, t_{h_1}), \dots, (t_1, t_{h_n})$.

But the worst case running time for this step is $(l - 1) \cdot T$.

As shown in Q10, the overall running time for all iterations of this step is $O(k^2 \cdot T)$.

Question 9

The set \mathcal{P} at the end of the algorithm has k elements, since it starts with 1 element and each step adds 1 element.

So the maximum cardinality of \mathcal{P} is k .

Question 10

Steps 1, 2, and 4 of the algorithm can be covered in linear time.

The algorithm performs $k - 1$ iterations of step 3. For each iteration of step 3, steps (b), (d), (e) can be done in linear time or less. Step (c) depends on the size of the set of edges, but can be done in linear time or less in the number of edges.

Step (a) is repeated over all remaining parts that contain more than 1 terminal. In the first iteration we have k terminals, where step (i) is run once with complexity $(k - 1) \cdot T$. On the next iteration we split V into 2 parts with $k - k_1$ and k_1 terminals respectively. Then step (i) is run twice with total complexity $(k - k_1 - 1) \cdot T + (k_1 - 1) \cdot T = (k - 2) \cdot T$. This continues with each iteration so that the complexity of step (a) for iteration j is $(k - j) \cdot T$, and the overall complexity for step (a) in all iterations is $\sum_{j=1}^{k-1} (k - j) \cdot T = \frac{k(k-1)}{2} \cdot T$.

This suggests that the overall complexity of the algorithm as written is $O(k^2 \cdot T)$. In practice this would do a lot of unnecessary recomputation of step (a)i, because on each iteration we split just one part, and so, if we remember the results of the computations in step (a)i from the previous step, we need to do a new

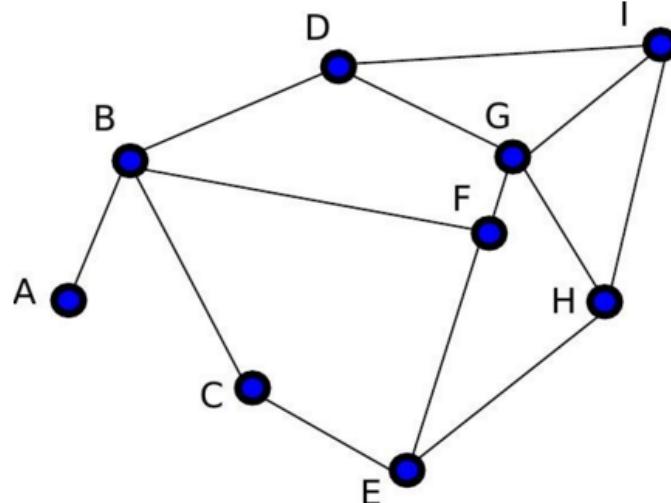
computation only for the 2 new sets that are the result of the last split. However, we may in the worst case (where the split is always such that one of the new sets has just one terminal) still require the same complexity.

So the overall complexity of the algorithm is $O(k^2 \cdot T)$. Since the number of terminals k may be much smaller than the number of vertices and edges, the complexity of T may dominate other parts of the algorithm. Note that T can be done in polynomial time, so the overall complexity is polynomial time.

1. This is a solution for the linear program for vertex cover on the following graph.

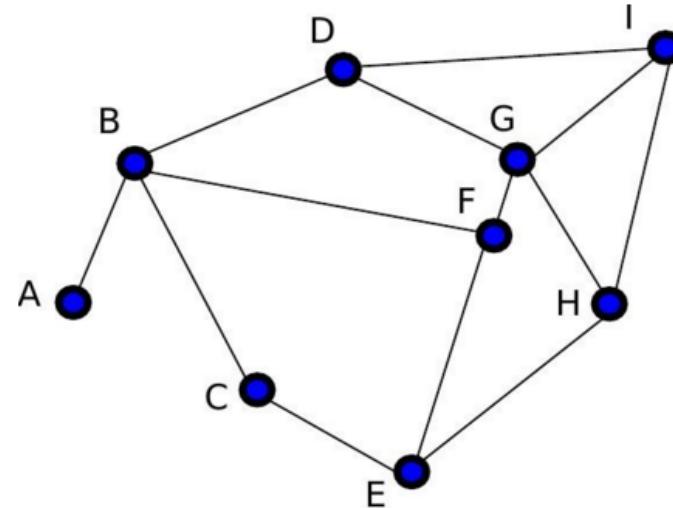
Pick the sets of vertices output by the rounding step of the algorithm, based on this solution.

$$\begin{array}{llll} x_A = 0.3, & x_B = 0.8, & x_C = 0.3, & x_D = 1, \\ x_E = 0.7, & x_F = 0.7, & x_G = 0.3, & x_H = 0.7, & x_I = 1 \end{array}$$



- {D,I}
- {B,E,G,I}
- {B,D,E,F,H,I}
- {B,E,F,H}

2. Consider the following graph.



This is the corresponding integer program:

$$\begin{aligned} \text{Minimize } & x_A + x_B + x_C + x_D + x_E + x_F + x_G + x_H + x_I \\ \text{subject to } & x_A + x_B \geq 1, \quad x_B + x_D \geq 1, \quad x_I + x_G \geq 1, \\ & x_B + x_C \geq 1, \quad x_D + x_G \geq 1, \quad x_E + x_F \geq 1, \\ & x_B + x_F \geq 1, \quad x_C + x_E \geq 1, \quad x_E + x_H \geq 1, \\ & x_H + x_G \geq 1, \quad x_H + x_I \geq 1, \quad x_I + x_D \geq 1, \quad x_F + x_G \geq 1. \\ & x_A, x_B, x_C, x_D, x_E, x_F, x_G, x_H, x_I \in \{0, 1\} \end{aligned}$$

What is the output of the algorithm presented during the lecture on this input?

- {B,E,G,I}
- {B,C,D,H}
- {B,E,D,G,H}
- {A,C,D,F,H,I}

1. Consider this linear program:

$$\text{Minimize } x_A + x_B + x_C + x_D + x_E + x_F + x_G + x_H + x_I$$

$$x_A + x_B \geq 1, \quad x_B + x_D \geq 1, \quad x_D + x_G \geq 1,$$

$$x_B + x_C \geq 1, \quad x_F + x_G \geq 1, \quad x_E + x_F \geq 1,$$

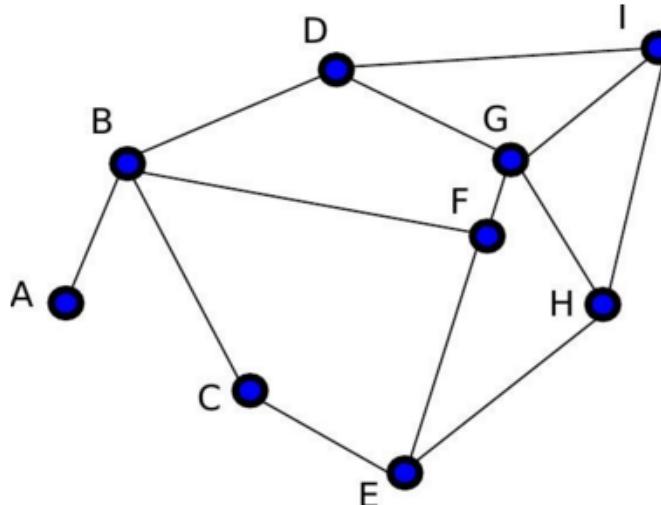
$$x_B + x_F \geq 1, \quad x_C + x_E \geq 1, \quad x_E + x_H \geq 1,$$

$$x_H + x_G \geq 1, \quad x_H + x_I \geq 1, \quad x_I + x_D \geq 1,$$

$$x_I + x_G \geq 1,$$

$$x_A, x_B, x_C, x_D, x_E, x_F, x_G, x_H, x_I \in \{0,1\}$$

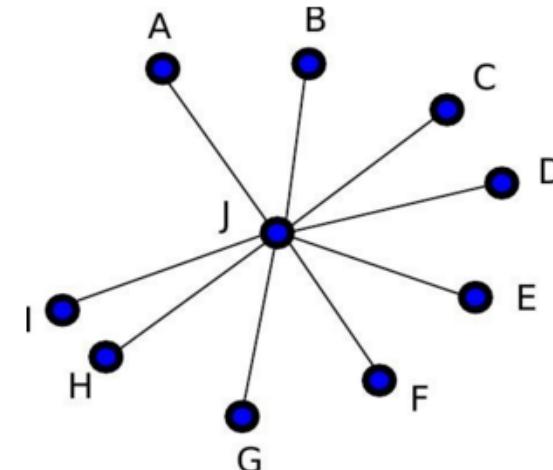
It is a linear program for the vertex cover problem on this graph:



Pick all the assignments that are a solution to the linear program and that lead the algorithm to obtain an optimal vertex cover for this graph.

- $x_A = 0.3, x_B = 0.8, x_C = 0.3, x_D = 1,$
 $x_E = 0.7, x_F = 0.7, x_G = 0.3, x_H = 0.7, x_I = 1.$
- $x_A = 0, x_B = 1, x_C = 0, x_D = 0, x_E = 1,$
 $x_F = 0, x_G = 1, x_H = 0, x_I = 1.$
- $x_A = 0.3, x_B = 0.8, x_C = 0.3, x_D = 0,$
 $x_E = 0.7, x_F = 0.4, x_G = 0.8, x_H = 0.4, x_I = 1.$
- $x_A = 0.1, x_B = 0.8, x_C = 0.3, x_D = 0.2, x_E = 0.7,$
 $x_F = 0.3, x_G = 1, x_H = 0.3, x_I = 0.8.$

2. Consider the following graph:



This is the linear program for the vertex cover problem on this graph.

$$\text{Minimize } x_A + x_B + x_C + x_D + x_E + x_F + x_G + x_H + x_I + x_J$$

$$x_A + x_J \geq 1, \quad x_B + x_J \geq 1, \quad x_D + x_J \geq 1,$$

$$x_C + x_J \geq 1, \quad x_F + x_J \geq 1, \quad x_E + x_J \geq 1,$$

$$x_G + x_J \geq 1, \quad x_H + x_J \geq 1, \quad x_I + x_J \geq 1,$$

$$x_A, x_B, x_C, x_D, x_E, x_F, x_G, x_H, x_I, x_J \geq 0$$

Consider the following assignment A:

$$x_A = 0.6, x_B = 0.6, x_C = 0.6, x_D = 0.6, x_E = 0.6, x_F = 0.6, x_G = 0.6, x_H = 0.6, \\ x_I = 0.6, x_J = 0.4$$

The rounding step on this assignment outputs the set $\{A, B, C, D, E, F, G, H, I\}$ but there exists an optimal vertex cover of size 1: $\{J\}$. The ratio between the two values of the two solutions is 7 but we proved in the lecture an approximation ratio of 2, what is the contradiction?

- The assignment A is not a solution to the linear program.
- The rounding step based on A does not output the set $\{A, B, C, D, E, F, G, H, I\}$.
- The assignment A is not an optimal solution to the linear program.
- The algorithm presented in the lectures uses the integer linear program instead of the linear program.

Quiz 7

2 questions

Submit Quiz

1. Pick all the algorithms that can solve linear programs with a polynomial-time worst-case running time.

- The Ellipsoid Method
- The Simplex Algorithm
- The Criss-Cross Algorithm
- The Projective Algorithm

2. What would be the implications of Unique Game Conjecture (UGC)?

- There is no polynomial-time algorithm that achieves a better approximation than 2 for vertex cover.
- $P = NP$
- $P \neq NP$
- There is no polynomial-time algorithm that achieves a better approximation than 0.878... for the maximum cut problem.

Submit Quiz



- Linus from Peanuts
 - Mickey Mouse
 - Harry Potter
 - The Smurfs
 - Linux
-

2. Pick all the correct statements.

If a problem H is NP-Hard, then...

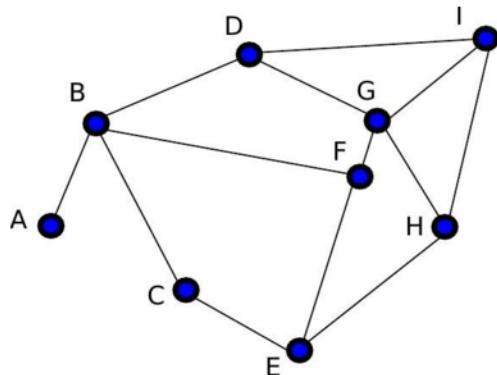
- For any problem L in P, there is a polynomial-time reduction from L to H.
 - H belongs to the class P.
 - H belongs to the class NP.
 - If there exists a polynomial-time algorithm for H then there exists a polynomial-time algorithm for the Hamiltonian path problem.
 - If there exists a polynomial-time algorithm for the Hamiltonian path problem then there exists a polynomial-time algorithm for H.
 - An output to problem H is "yes" or "no".
 - For any problem L in NP, there is a polynomial-time reduction from L to H.
-

3. Pick all the correct statements.

If a problem H is NP-Complete, then...

- For any problem L in NP, there is a polynomial-time reduction from L to H.
- For any problem L in P, there is a polynomial-time reduction from L to H.
- H belongs to the class P.
- H belongs to the class NP.
- If there exists a polynomial-time algorithm for the Hamiltonian path problem then there exists a polynomial-time algorithm for H.
- An output to problem H is "yes" or "no".
- If there exists a polynomial-time algorithm for H then there exists a polynomial-time algorithm for the Hamiltonian path problem.

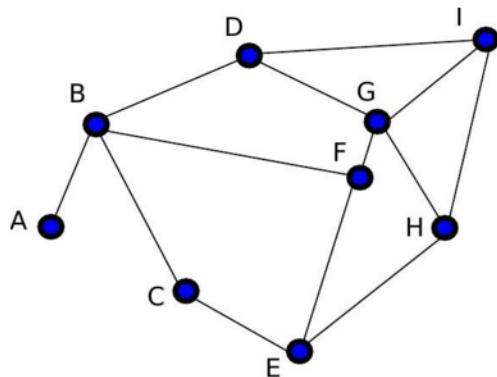
1. Consider the following graph.



Which sets of vertices induce a Vertex Cover? Pick all the correct answers.

- {B,E,G,I}
- {A,C,D,H}
- {D,F,C,A,G,H}
- {B,E,D,H}

2. Consider the following graph.



Which sets of vertices induce an optimal Vertex Cover? Pick all the correct answers.

- {B,E,G,I}
- {D,H,F,B}
- {D,F,C,A,G,H}
- {B,E,D,H}

3. Consider the complete graph with n vertices.

What is the value of the optimal vertex cover?

- n-1
- n
- 1
- n/2

4. Consider the complete bipartite graph with A vertices on one side and B vertices on the other side.

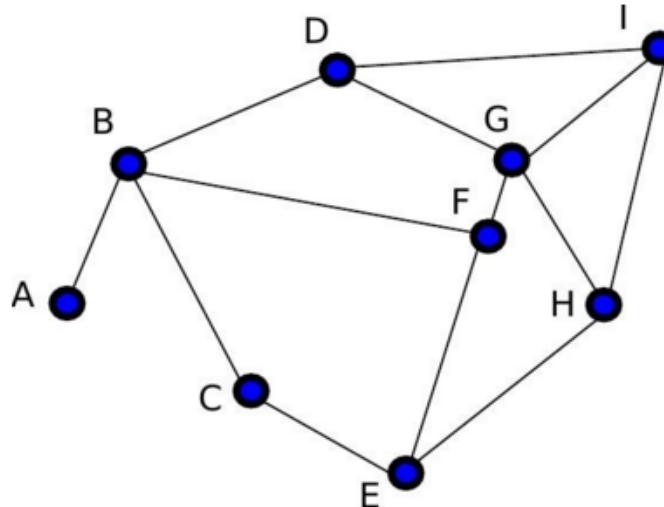
What is the value of the optimal vertex cover?

- A
- A/2 + B/2
- max(A,B)
- min(A,B)

[Submit Quiz](#)



1. Is the following integer program for vertex cover on this graph correct? Pick all the correct statements.



Minimize $x_A + x_B + x_C + x_D + x_E + x_F + x_G + x_H + x_I$

$$x_A + x_B \geq 1, \quad x_B + x_D \geq 1, \quad x_I + x_G \geq 1,$$

$$x_B + x_C \geq 1, \quad x_D + x_G \geq 1, \quad x_E + x_F \geq 1,$$

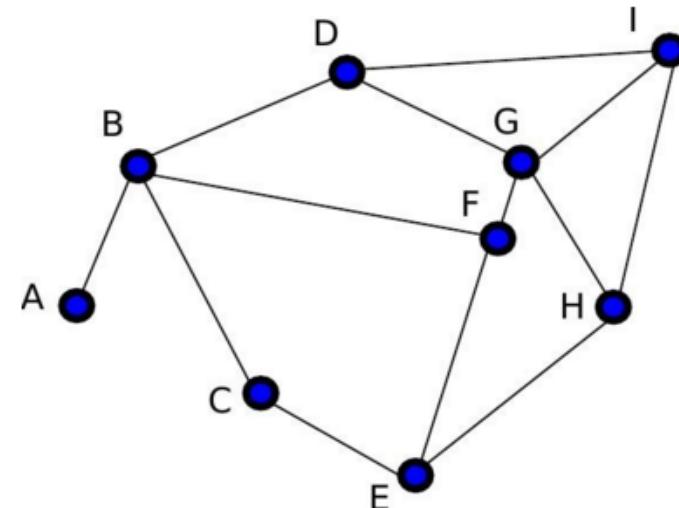
$$x_B + x_F \geq 1, \quad x_C + x_E \geq 1, \quad x_E + x_H \geq 1,$$

$$x_H + x_G \geq 1, \quad x_H + x_I \geq 1, \quad x_I + x_D \geq 1,$$

$$x_A, x_B, x_C, x_D, x_E, x_F, x_G, x_H, x_I \in \{0,1\}$$

- No missing constraint
- $x_B + x_C = 1$ is a missing constraint
- $x_F + x_G \geq 1$ is a missing constraint
- $x_C + x_A \geq 1$ is a missing constraint

2. Consider the following graph.



What is the assignment to the variables $x_A, x_B, x_C, x_D, x_E, x_F, x_G, x_H, x_I$ that corresponds to the vertex cover $\{A, C, F, D, H, I\}$.

- $x_A = 1, x_B = 0, x_C = 0.8, x_D = 0.2, x_E = 0, x_F = 0, x_G = 1, x_H = 1, x_I = 1$
- $x_A = 0.7, x_B = 0.3, x_C = 1, x_D = 0.2, x_E = 1, x_F = 0, x_G = 1, x_H = 0, x_I = 1$
- $x_A = 1, x_B = 0, x_C = 1, x_D = 1, x_E = 0, x_F = 1, x_G = 0, x_H = 1, x_I = 1$
- $x_A = 1, x_B = 0, x_C = 1, x_D = 0, x_E = 1, x_F = 1, x_G = 0, x_H = 0, x_I = 1$

Quiz 4

3 questions

Submit Quiz

1. Is this assignment a solution to the following linear program? Pick all the constraints of the LP that are not satisfied.

Assignment:

$$\begin{array}{llll} x_A = 0.3, & x_B = 0.8, & x_C = 0.2, & x_D = 1, \\ x_E = 0.7, & x_F = 0.7, & x_G = 0.3, & x_H = 0.1, & x_I = 1. \end{array}$$

Linear Program:

$$\text{Minimize } x_A + x_B + x_C + x_D + x_E + x_F + x_G + x_H + x_I$$

$$x_A + x_B \geq 1, \quad x_B + x_D \geq 1, \quad x_D + x_G \geq 1,$$

$$x_B + x_C \geq 1, \quad x_F + x_G \geq 1, \quad x_E + x_F \geq 1,$$

$$x_B + x_F \geq 1, \quad x_C + x_E \geq 1, \quad x_E + x_H \geq 1,$$

$$x_H + x_G \geq 1, \quad x_H + x_I \geq 1, \quad x_I + x_D \geq 1,$$

$$x_I + x_G \geq 1,$$

$$x_A, x_B, x_C, x_D, x_E, x_F, x_G, x_H, x_I \geq 0$$

- $x_A + x_B \geq 1$
- $x_E + x_G \geq 1$
- $x_A + x_C \geq 1$
- $x_H + x_G \geq 1$

2. Consider the following integer linear program.

$$\text{maximize } x_A + x_B + x_C$$

subject to:

$$x_A + x_B \leq 1,$$

$$x_A + x_C \leq 1,$$

$$x_B + x_C \leq 1,$$

$$x_A, x_B, x_C \in \{0,1\}$$

What is the value of the optimal solution?

1

3. Consider the following linear program.

$$\text{maximize } x_A + x_B + x_C$$

subject to:

$$x_A + x_B \leq 1,$$

$$x_A + x_C \leq 1,$$

$$x_B + x_C \leq 1,$$

$$x_A, x_B, x_C \geq 0$$

What is the value of the optimal solution?

1.5

Submit Quiz





Quiz 4

1 question

Submit Quiz

1. How long does it take to execute the following code.

$$\max_v \{v : A[i, v] \leq B\}$$

$O(N)$

$O(n^2)$

Submit Quiz

Quiz 5

1 question

Submit Quiz

1. Consider the Algorithm Scale (slide 5.1, page 9 and 10). What's the approximation factor if $N = 2n$.

2/3

3/2

2

42

200

1

100

Submit Quiz



Quiz 6

1 question

Submit Quiz

1. Run Algorithm Scale (slide 5.1. page 9 and 10) with $N = 100n$. What happens if you don't deleted the items which don't fit. Suppose the largest item has size $4B$.



The approximation factor becomes worse than $1/0.99$.



The asymptotic running time becomes worse.

Submit Quiz



Quiz 7

1 question

Submit Quiz

1. Consider the Algorithm Scale (slide 5.1, page 9 and 10).

What's the approximation factor if $N = n^2$.

- $1/(1 - 1/n)$
- $1/(1 - 1/n^2)$
- $1/\log(n)$
- n
- n^2
- 42

Submit Quiz



Quiz 1

1 question

Submit Quiz

1. Consider a knapsack with capacity $B = 13$ and the following items $\{4, 11, 7, 5, 3\}$, where the weight for all items equals their value, i.e., $v_i = s_i$ for all items i . What's the value of the optimal packing?

- 11
- 12
- 13
- 14

Submit Quiz

Quiz 2

1/1 question correct



[Retake](#)

Excellent!

[Next](#)

- ✓ 1. Consider the setting where the size equals the value ($s_i = v_i$).

Let $1, 2, \dots, k$ be the items packed in an optimal solution where $v_{i-1} \geq v_i$ for $1 \leq i \leq k$.

- If an algorithm packs any $k/2$ of these items, then it gives a 2-approximation.

Well done!

- If an algorithm packs at least the items $1, 2, \dots, \lceil k/2 \rceil$, then it is a 2-approximation.

Well done!

- Any optimal algorithm must pack all items $1, 2, \dots, k$

Well done!

For example if the item $k+1$ has the same value and size as item k , then packing item $1, 2, \dots, k-1$ and item $k+1$ is also optimal.

Quiz 3

2/2 questions correct

Excellent!

[Retake](#)

[Next](#)

- ✓ 1. Consider the array A where $A[i, v] = \min$ size among subsets of the first i items with total value v . Let B be the size of the knapsack. Is it possible to obtain the optimal solution, i.e., the items in the optimal packing, and not only the value of the optimal packing OPT ?

- This can be achieved by backtracking starting at
 $A[i, B]$.

Well done!

- This can be achieved by backtracking starting at
at the maximum entry of A .

Well done!

- The optimal solution can't be recovered and has to be recomputed from scratch.

Well done!

- A simple solution requiring only polynomial time is to try all possibilities of items and check whether they yield the value of OPT .

Well done!

- ✓ 2. $A[i, s] = \max$ value among subsets of the first i items with total size s . Which options are correct?

- This algorithm works nice in the setting where the sizes are small integers.

Well done!

- This algorithm works nice in the setting where the sizes are small numbers.

Well done!

If the size have varying sizes, e.g., between $1/e^{100}$ and $1/e^{10}$, then no computer can store the dynamic program.

Quiz 2

3/3 questions correct

Excellent!

Retake

Next

- ✓ 1. Consider the instance of bin-packing with 3 items, each one having a size of $2/3$ and bins of capacity one. For this instance, select the constraint that is not part of the linear program for two bins:

- $x_{11} + x_{12} = 1$
- $x_{21} + x_{22} = 1$
- $x_{31} + x_{32} = 1$
- $x_{11} + x_{21} + x_{31} \geq 2$

Well done!

- $\frac{2}{3}x_{11} + \frac{2}{3}x_{21} + \frac{2}{3}x_{31} \leq 1$
- $\frac{2}{3}x_{12} + \frac{2}{3}x_{22} + \frac{2}{3}x_{32} \leq 1$
- $x \geq 0$

- ✓ 2. Is the linear program, for two bins, feasible?

- Yes

Well done!

- No

- ✓ 3. What is the number of bins used in the optimal packing for this instance?

- 2
- 3

Well done!

- 3.5



Quiz 3

1/1 question correct



Retake

Excellent!

Next

- ✓ 1. In this lecture was shown that Next-Fit returns a solution using $\frac{3}{2} \text{opt} + 1$ bins if the size of every item is at most $1/3$. Suppose now that every item has size at most $1/4$. If opt is the number of bins used in the optimal packing, then, how many bins are used, at most, in the solution returned by Next-Fit?

$\frac{4}{3} \text{opt} + 1$ bins.

Well done!

$\frac{3}{2} \text{opt} + 1$ bins.

Well done!

$\text{opt} + 1$ bins.

Well done!

Quiz 4

3/3 questions correct

Excellent!

Retake

Next

- ✓ 1. Consider an instance with n items, all of them with size strictly greater than $1/2$. We run the Next-Fit algorithm for some order of the items. Which of the following statements are true?

The number of items in the packing returned by Next-Fit depends on the order of the items.

Well done!

No matter what order we consider, the value is always n .

Well done!

The algorithm returns an optimal packing.

Well done!

- ✓ 2. In the slides was shown a formulation of the Bin-Packing problem in terms of *configurations*. In particular, the instance considered have items of size 3 and 4. If the capacity of the bins is 12, what are the missing configurations in the slides?

{3, 3, 3, 3}

Well done!

{4, 3, 4, 1}

Well done!

The empty configuration.

Well done!

{4}

Well done!

- ✓ 3. Consider the instance of bin-packing with 3 items, each one having a size of $2/3$ and bins of capacity one. The possible configurations for this instance are: the empty configuration and {2}. Select the true statements:

The minimum of the configuration LP relaxation is less than 3.

Well done!

The minimum of the configuration LP relaxation is exactly 3.

Well done!

The number of bins used in the optimal solution is 3.

Well done!



Quiz 5

2 questions

Submit Quiz

1. Consider the following instance for Bin-Packing: there are eight items of size 16, seven items of size 20, one item of size 31, one item of size 32, seven items of size 35, one item of size 41, one item of size 42 and one item of size 45. Suppose $\epsilon = 1/3$, and construct the new instance as shown in the slides, that is, we sort the items in non-decreasing size, we create groups of cardinality $n\epsilon^2$, and we round-up the size values in every group to the maximum size in it. How many different sizes are there in the new instance?

- 4
- 5
- 6

2. Suppose that the items are packed into bins of capacity 100. Select the true statements:

- In the new instance there are nine items with size 35.
- In the new instance we have 30 items.
- The largest size in the new instance is 45.
- The optimal number of bins for the new instance is at least 8 and at most 9.

Submit Quiz

Quiz 6

3 questions

Submit Quiz

1. Consider the following instance for Bin-Packing: there are eight items of size 16, seven items of size 20, one item of size 31, one item of size 32, seven items of size 35, one item of size 41, one item of size 42 and one item of size 45. Suppose $\epsilon = 1/3$, and construct the new instance as follows: we sort the items in non-decreasing size, we create groups of cardinality $n\epsilon^2$, and we round-down the size values in every group to the minimum size in it. How many different sizes are there in the new instance?

- 4
- 5
- 6

2. Suppose that the items are packed into bins of capacity 100. Select the true statements:

- In the new instance there are nine items with size 35.
- In the new instance we have 27 items.
- The largest size in the new instance is 41.
- The optimal number of bins for the new instance is at least 7 and at most 8.

3. The difference between the optimal number of items used in the rounded-up solution and the rounded-down solution is not greater than ...

- 2
- 3
- 4

Submit Quiz

Quiz 7

3/3 questions correct



Retake

Excellent!

Next

- ✓ 1. Consider the following instance for Bin-Packing: there are eight items of size 16, seven items of size 20, one item of size 31, one item of size 32, seven items of size 35, one item of size 41, one item of size 42 and one item of size 45. Suppose $\epsilon = 1/3$, the bins have capacity $B = 100$ and we run the general algorithm shown in the slide 2. Select the true statements.

When the items with size less than ϵB are set aside, it is obtained an instance with 10 items.

Well done!

Each group has cardinality 1.

Well done!

After rounding-up there are 2 different values.

Well done!

- ✓ 2. The optimal number of bins for the rounded problem U is equal to ...

4

5

Well done!

6

- ✓ 3. Suppose that the small items are sorted according to non-decreasing size, and they are packed greedily starting from the bins already opened by U . Select the true statements.

It is obtained a solution for the original instance using 7 bins.

Well done!

It is obtained a solution for the original instance using 8 bins.

Well done!

In addition to the bins opened because of U , it is necessary to open at least 3 new bins.

Well done!

- ✓ 1. Select the true statements about the Next-Fit algorithm:

The Next-Fit algorithm always returns an optimal solution.

Well done!

In the Next-Fit algorithm, if a bin was closed it is never reused later.

Well done!

The Next-Fit algorithm is an asymptotic 2-approximation.

Well done!

The output of the Next-Fit algorithm depends on the order of the items.

Well done!

The Next-Fit algorithm is a 3-approximation.

Well done!

- ✓ 2. Consider the bin-packing instance with six items and sizes $s_1 = 0.7$, $s_2 = 0.6$, $s_3 = 0.4$, $s_4 = 0.3$, $s_5 = 0.45$ and $s_6 = 0.5$. We run the Next-Fit algorithm with items in the order 2, 5, 3, 1, 6, 4. How many bins are used in the packing returned by the algorithm?

3

Well done!

4

Well done!

5

Well done!

6

Well done!



Quiz 5

1/1 question correct

Excellent!

Retake

Next

- ✓ 1. What if you repeat the randomized rounding algorithm $\ln(n) + 6$ times instead of $\ln(n) + 3$ times using the same arguments as before. What is the probability that the output is a cover.

- $Pr(\text{cover}) \approx 0.002$
- $Pr(\text{cover}) \approx 0.03$
- $Pr(\text{cover}) \approx 0.95$
- $Pr(\text{cover}) \approx 0.998$

Well done!

Quiz 6

1/1 question correct

Excellent!

Retake

Next

- ✓ 1. Guarantee Set Cover: Iterated randomized rounding gives collection of sets that is a set cover with probability 95 %. Assume that the expected running time is t .

How can you convert this algorithm into an algorithm that returns a set cover with probability 100%?

- Check if the output is a set cover (assume this can be done instantly). If not repeat until the output is a proper set cover. The expected running time is bounded by $2t$.

Well done!

- Run the mentioned algorithm. Check if the output is a set cover. If not go through all sets having items not covered yet and add them. The output will be a valid set cover (the approximation ratio might become worse).

Well done!

Quiz 7

1/1 question correct

Retake

Excellent!

Next

- ✓ 1. Stopping time: A stopping time with respect to a sequence of random variables X_1, X_2, \dots is a random variable τ with values in $\{1, 2, 3, \dots\}$ and the property that for each $t \in \{1, 2, \dots\}$, the occurrence or non-occurrence of the event $\tau = t$ depends only on the values of X_1, X_2, \dots, X_t . Furthermore, it must hold that $\Pr(\tau < \infty) = 1$.

Consider the following example. You play in a casino starting with \$100 and you bet \$1 on red in every game. Which of the following rules are valid stopping rules?

(Borrowed from https://en.wikipedia.org/wiki/Stopping_time)

Playing exactly 5 rounds.

Well done!

Playing until you run out of money or you have \$500.

Well done!

Playing until you are the maximum ahead you will ever be.

Well done!

The event $t = \tau$ depends on X_{t+1}, X_{t+2}, \dots

Playing until the money doubles.

Well done!

There is a positive probability that you run out of money before you ever reach \$200.
Hence $\Pr(\tau < \infty) < 1$

Playing until the money doubles or you run out of money.

Well done!

Quiz 8

2/2 questions correct



Retake

Excellent!

Next



- ✓ 1. Set Cover is a special case of Vertex Cover.

- Yes
 No

Well done!

Vertex Cover is a special case of Set Cover, but SC is not a special case of VC.

- ✓ 2. Let A and B be two events. Then $\Pr(A \wedge B) \leq \Pr(A) \cdot \Pr(B)$.

- Yes
 No

Well done!

Consider the event A the number shown by the die is at most 3 and the event B the number shown by the die is at most 1.

Quiz 1

1/1 question correct



Retake

Excellent!

Next



- ✓ 1. Rounding: Why do we need to round $x_S = 0.9$ to 1 or 0 instead of keeping it at 0.9?

Because

rounding will give a 'cheaper' solution. We can beat the value of the fractional solution.

Well done!

we don't want to store floats.

Well done!

either the output cover includes set S or it doesn't; the model doesn't allow taking 90% of it.

Well done!



Quiz 2

1/1 question correct



Retake

Excellent!

Next

- ✓ 1. Which of the following statements is correct?(multiple)

After rounding, the value of the objective function can increase.

Well done!

The rounded solution might not be a valid solution.

Well done!

The rounding might take 'too' long.

Well done!

(Of course, If one tries really hard, then one can come up with a rounding scheme which takes super polynomial time)

Quiz 3

1/1 question correct



Retake

Excellent!

Next

- ✓ 1. Say you have a random variable X which only takes two values: $b \in \mathbb{N}$ and 0.

How do you have to choose b such that the following holds: $E[X] = \Pr(X = b)$.

$b = 2$

Well done!

$b = -1$

Well done!

$b = 42$

Well done!

$b = 1$

Well done!

Quiz 4

2/2 questions correct



[Retake](#)

Excellent!

[Next](#)

- ✓ 1. You throw a die. Let A be the event that

the die shows an even number. Let B be the event that the die shows a number which is divisible by 3.

A and B are independent.

Well done!

A and B are correlated.

- ✓ 2. You throw a die. Let A be the event that

the die shows an even number. Let B be the event that the die shows a number which is smaller or equal 3.

A and B are independent.

A and B are correlated.

Well done!

Quiz 4

2/2 questions correct

Excellent!

Retake

Next

- ✓ 1. Consider the multiway cut problem on a graph $G = (V, E)$ with k terminals.

Suppose the edge (u, v) belongs to the output of the algorithm.

Suppose further that $r < \min_t d(a_{\sigma(t)}, u)$.

Pick the correct implications.

Node u belongs to the cluster of $a_{\sigma(k)}$.

Well done!

Node v belongs to the cluster of $a_{\sigma(k)}$.

Well done!

Node u belongs to the cluster of $a_{\sigma(j)}, j \neq k$.

Well done!

Node u doesn't belong to any cluster.

Well done!

- ✓ 2. Assume $k = 4$. Let a be the approximation ratio achieved by the simple algorithm for multiway cut presented at the beginning of the lecture. Let b be the approximation guarantee achieved by the algorithm presented in the last slide of lecture 3.

What is the ratio a/b ?

1.2

Well done!

Quiz 5

2 questions

Submit Quiz

1. Pick all the constraints that can be formulated as linear constraints.

$x \geq |y|$

$x \leq |y|$

-
2. Which of the following problems can be formulated as an instance of the multiway cut problem?

Assigning users to base computers in a multicomputer environment.

Partitioning files among the nodes of a network.

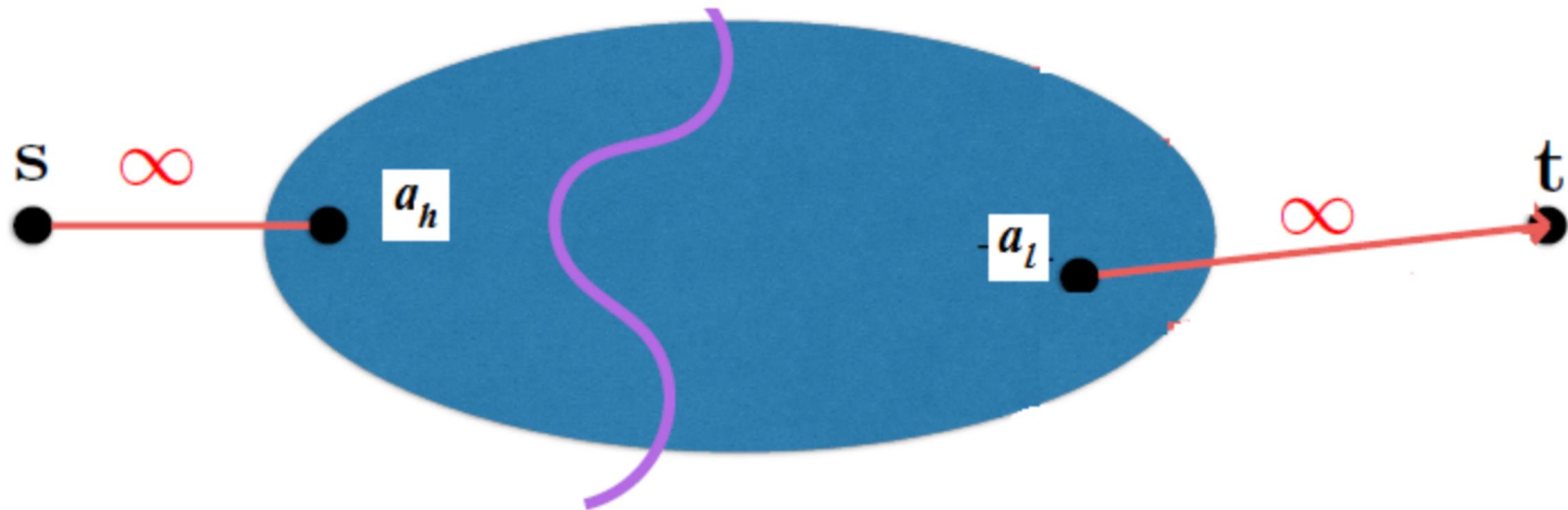
Designing a schedule for a university.

Finding the shortest way to deliver pizzas on mars.

Submit Quiz

Mincut(s,t)

G_i



Quiz 1 : Some context on cuts

2/2 questions correct

Excellent!

Retake

Next

- ✓ 1. Computing a cut for 2 terminals can be done in polynomial time. What is the running time of the Stoer-Wagner Minimum Cut Algorithm mentioned on Wikipedia? (below, n is the number of vertices and m is the number of edges in the graph.)

$O(nm + n^2 \log n)$

Well done!

$O(m + n \log n)$

Well done!

$O(n^2 m)$

Well done!

Not bounded since the value of the cut can be irrational.

Well done!

- ✓ 2. The minimum cut for two terminals can be computed in polynomial time. What about the maximum cut? Pick all the correct statements.

It can be solved in time $O(n + m)$ in general graphs.

Well done!

It is NP-Hard for general graphs.

Well done!

It is APX-Hard for general graphs.

Well done!

It can be solved in polynomial time in planar graphs.

Well done!

It can be solved in polynomial time in general graphs.

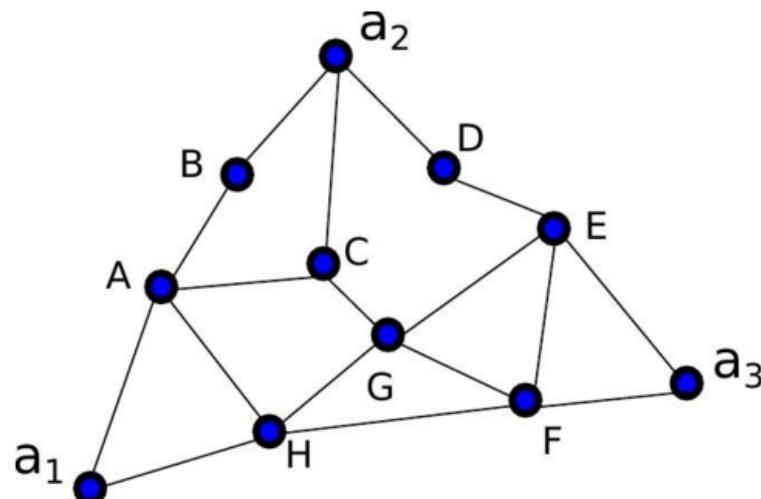
Well done!

Quiz 2

3 questions

[Submit Quiz](#)

1. Consider the IP formulation of the multiway cut problem for the following graph $G = (V, E)$ with 3 terminals a_1, a_2, a_3 .



Pick all the constraints that belong to the IP.

- $z_{(A,H),2} \geq x_{A,2} - x_{H,2}$
- $z_{(A,H),2} \geq 1 - x_{A,2} + x_{A,3}$
- $z_{(A,H),2} \geq x_{A,2} - x_{H,2}$
- $\sum_{v \in V} x_{v,2} = 1$

2. Consider the IP formulation of the multiway cut problem seen during the lecture.

For each edge (u, v) and terminal a_i , there are two constraints $z_{(u,v),i} \geq x_{u,i} - x_{v,i}$ and $z_{(u,v),i} \geq x_{v,i} - x_{u,i}$.

Why are the constraints $z_{(u,v),i} \leq x_{u,i} - x_{v,i}$ and $z_{(u,v),i} \leq x_{v,i} - x_{u,i}$ not part of the formulation?

- They are not needed.
- They make the formulation incorrect.
- There would be too many constraints.

3. Consider the IP formulation of the multiway cut problem seen during the lecture.

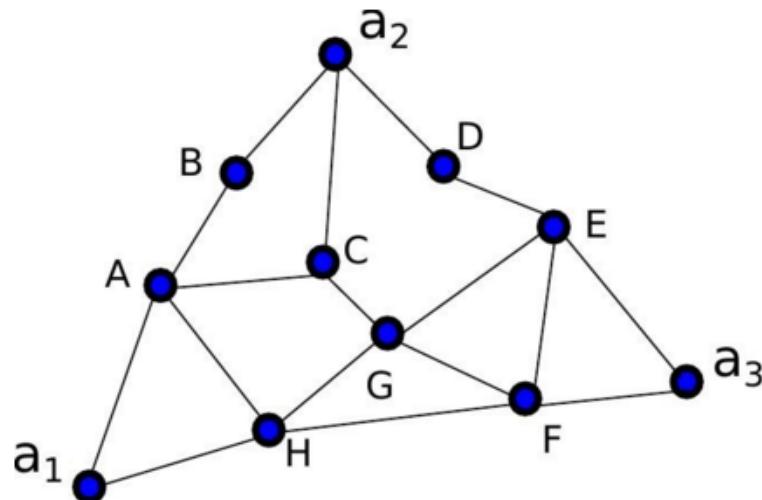
Suppose there are k terminals, n vertices and m edges in the input graph.

How many dimensions are involved in the geometric interpretation?

- n
- 3
- k
- $k(m+n)$

[Submit Quiz](#)

1. Consider the following graph and the IP relaxation for the multiway cut problem described in the lectures.



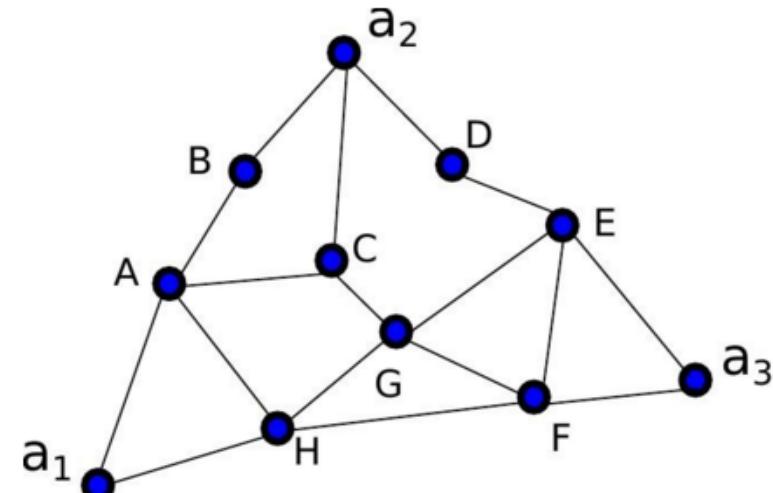
Consider the solution whose positive variables are the following:

$$x_{A1} = 1 \ x_{B2} = 1 \ x_{C3} = 1 \ x_{D2} = 1 \ x_{E3} = 1 \ x_{F3} = 1 \ x_{H1} = 1 \ x_{G1} = 1$$

What are the edges in the cut after running the rounding process described in the lectures?

- (A,B), (A,C), (C,G), (E,G), (F,G), (D,E)
 - (A,H), (H,G), (H,F), (F,G), (F,E)
 - (A,B), (A,C), (C,G), (E,G), (F,G), (F,H), (D,E)

2. Consider the following graph and the IP relaxation for the multiway cut problem described in the lectures.



Consider the solution whose positive variables are the following:

$$\begin{aligned}x_{A,1} &= 0.4, x_{A,2} = 0.3, x_{A,3} = 0.3, x_{B,2} = 1, x_{C,2} = 0.6, x_{C,1} = 0.4, x_{D,2} = 0.2, \\x_{D,3} &= 0.8, x_{E,3} = 1, x_{F,1} = 0.15, x_{F,2} = 0.15, x_{F,3} = 0.7, x_{H,1} = 1, x_{G,1} = 1/3, \\x_{G,2} &= 1/3, x_{C,2} = 1/3\end{aligned}$$

Consider the rounding process described during the lectures and suppose that the random order obtained is w_2, w_3, w_1 , and $r = 0.6$.

Pick all the correct statements

- Vertices A, F, G are in the cluster of a_1 .
 - Vertices E, F are in the cluster of a_3 .
 - Vertices C, G are in the cluster of a_2 .
 - Vertices A, B, H are in the cluster of a_1 .



Approximation Algorithms Part I

by École normale supérieure

Course Home

[Week 1](#)[Week 2](#)[Week 3](#)[Week 4](#)[Week 5](#)[Grades](#)[Discussion Forums](#)[Course Info](#)

Claire Mathieu

Welcome to Approximation Algorithms Part I! You're joining thousands of learners currently enrolled in the course. I'm excited to have you in!

▼ More

Congratulations!

You've successfully completed **Approximation Algorithms Part I**



✓ WEEK 1

✓ WEEK 2

✓ WEEK 3

✓ WEEK 4

✓ WEEK 5



Course Home

5/5

Assignments Passed

97%

Final Course Grade

Grades

Discussion Forums

Course Info

| | | Due | Weight | Passed | Grade |
|---|----|--------|--------|---|-------|
| Vertex cover and Linear Programming | | | | | |
|  Peer-graded Assignment: Peer Graded Assignment 1 | 2h | Dec 13 | 20% |  | 100% |
|  Review Your Peers: Peer Graded Assignment 1 | | Dec 16 | 20% |  | -- |
| Knapsack and Rounding | | | | | |
|  Peer-graded Assignment: Peer Assignment Knapsack | 2h | Dec 20 | 20% |  | 100% |
|  Review Your Peers: Peer Assignment Knapsack | | Dec 23 | 20% |  | -- |
| Bin Packing, Linear Programming and Rounding | | | | | |
|  Peer-graded Assignment: Peer Assignment: Bin-Packing | 2h | Dec 27 | 20% |  | 100% |
|  Review Your Peers: Peer Assignment: Bin-Packing | | Dec 30 | 20% |  | -- |
| Set Cover and Randomized Rounding | | | | | |
|  Peer-graded Assignment: Peer Assig Set Cover | 2h | Jan 3 | 20% |  | 100% |
|  Review Your Peers: Peer Assig Set Cover | | Jan 6 | 20% |  | -- |
| Multiway Cut and Randomized Rounding | | | | | |
|  Peer-graded Assignment: Peer-graded assignment 5 | 2h | Jan 10 | 20% |  | 85% |
|  Review Your Peers: Peer-graded assignment 5 | | Jan 13 | 20% |  | -- |