

Advanced Algorithms (Fall 2025)

Linear Programming Rounding Algorithms

Lecturers: 尹一通, 刘景铄, 栗师

Nanjing University

Approximation Algorithm based on LP Rounding

- Opti. Problem $X \iff 0/1$ Integer Program (IP) $\xRightarrow{\text{relax}}$ LP

Approximation Algorithm based on LP Rounding

- Opti. Problem $X \iff 0/1$ Integer Program (IP) $\xrightarrow{\text{relax}}$ LP

0/1 Integer Program

$$\begin{aligned} \min \quad & c^T x \\ Ax & \geq b \\ x & \in \{0, 1\}^n \end{aligned}$$

Linear Program Relaxation

$$\begin{aligned} \min \quad & c^T x \\ Ax & \geq b \\ x & \in [0, 1]^n \end{aligned}$$

Approximation Algorithm based on LP Rounding

- Opti. Problem $X \iff$ 0/1 Integer Program (IP) $\xrightarrow{\text{relax}}$ LP

0/1 Integer Program

$$\begin{aligned} \min \quad & c^T x \\ Ax & \geq b \\ x & \in \{0, 1\}^n \end{aligned}$$

Linear Program Relaxation

$$\begin{aligned} \min \quad & c^T x \\ Ax & \geq b \\ x & \in [0, 1]^n \end{aligned}$$

- $LP \leq IP$

Approximation Algorithm based on LP Rounding

- Opti. Problem $X \iff$ 0/1 Integer Program (IP) $\xrightarrow{\text{relax}}$ LP

0/1 Integer Program

$$\begin{aligned} \min \quad & c^T x \\ Ax &\geq b \\ x &\in \{0, 1\}^n \end{aligned}$$

Linear Program Relaxation

$$\begin{aligned} \min \quad & c^T x \\ Ax &\geq b \\ x &\in [0, 1]^n \end{aligned}$$

- $LP \leq IP$
- Integer programming is NP-hard, linear programming is in P

Approximation Algorithm based on LP Rounding

- Opti. Problem $X \iff$ 0/1 Integer Program (IP) $\xrightarrow{\text{relax}}$ LP

0/1 Integer Program

$$\begin{aligned} \min \quad & c^T x \\ Ax & \geq b \\ x & \in \{0, 1\}^n \end{aligned}$$

Linear Program Relaxation

$$\begin{aligned} \min \quad & c^T x \\ Ax & \geq b \\ x & \in [0, 1]^n \end{aligned}$$

- $LP \leq IP$
- Integer programming is NP-hard, linear programming is in P
- Solve LP to obtain a fractional $x \in [0, 1]^n$.

Approximation Algorithm based on LP Rounding

- Opti. Problem $X \iff$ 0/1 Integer Program (IP) $\xrightarrow{\text{relax}}$ LP

0/1 Integer Program

$$\begin{aligned} \min \quad & c^T x \\ Ax & \geq b \\ x & \in \{0, 1\}^n \end{aligned}$$

Linear Program Relaxation

$$\begin{aligned} \min \quad & c^T x \\ Ax & \geq b \\ x & \in [0, 1]^n \end{aligned}$$

- $LP \leq IP$
- Integer programming is NP-hard, linear programming is in P
- Solve LP to obtain a fractional $x \in [0, 1]^n$.
- Round** it to an integral $\tilde{x} \in \{0, 1\}^n \iff$ solution for X

Approximation Algorithm based on LP Rounding

- Opti. Problem $X \iff$ 0/1 Integer Program (IP) $\xrightarrow{\text{relax}}$ LP

0/1 Integer Program

$$\begin{aligned} \min \quad & c^T x \\ Ax & \geq b \\ x & \in \{0, 1\}^n \end{aligned}$$

Linear Program Relaxation

$$\begin{aligned} \min \quad & c^T x \\ Ax & \geq b \\ x & \in [0, 1]^n \end{aligned}$$

- LP $\leq$ IP
- Integer programming is NP-hard, linear programming is in P
- Solve LP to obtain a fractional $x \in [0, 1]^n$.
- Round** it to an integral $\tilde{x} \in \{0, 1\}^n \iff$ solution for X
- Prove $c^T \tilde{x} \leq \alpha \cdot c^T x$, then $c^T \cdot \tilde{x} \leq \alpha \cdot \text{LP} \leq \alpha \cdot \text{IP} = \alpha \cdot \text{opt}$

Approximation Algorithm based on LP Rounding

- Opti. Problem $X \iff$ 0/1 Integer Program (IP) $\xrightarrow{\text{relax}}$ LP

0/1 Integer Program

$$\begin{aligned} \min \quad & c^T x \\ Ax & \geq b \\ x & \in \{0, 1\}^n \end{aligned}$$

Linear Program Relaxation

$$\begin{aligned} \min \quad & c^T x \\ Ax & \geq b \\ x & \in [0, 1]^n \end{aligned}$$

- LP $\leq$ IP
- Integer programming is NP-hard, linear programming is in P
- Solve LP to obtain a fractional $x \in [0, 1]^n$.
- Round it to an integral $\tilde{x} \in \{0, 1\}^n \iff$ solution for X
- Prove $c^T \tilde{x} \leq \alpha \cdot c^T x$, then $c^T \cdot \tilde{x} \leq \alpha \cdot \text{LP} \leq \alpha \cdot \text{IP} = \alpha \cdot \text{opt}$
- $\implies \alpha$ -approximation

IP

$$\min \quad c^T x$$

$$Ax \geq b$$

$$x \in \{0, 1\}^n$$

LP Relaxation

$$\min \quad c^T x$$

$$Ax \geq b$$

$$x \in [0, 1]^n$$



IP

$$\begin{aligned} \min \quad & c^T x \\ Ax \geq & b \\ x \in & \{0, 1\}^n \end{aligned}$$

LP Relaxation

$$\begin{aligned} \min \quad & c^T x \\ Ax \geq & b \\ x \in & [0, 1]^n \end{aligned}$$



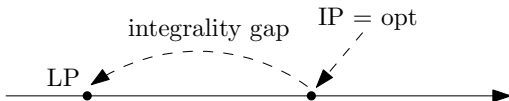
Def. The ratio between $\text{IP} = \text{opt}$ and LP is called the **integrality gap** of the LP relaxation.

IP

$$\begin{aligned} \min \quad & c^T x \\ Ax \geq & b \\ x \in & \{0, 1\}^n \end{aligned}$$

LP Relaxation

$$\begin{aligned} \min \quad & c^T x \\ Ax \geq & b \\ x \in & [0, 1]^n \end{aligned}$$



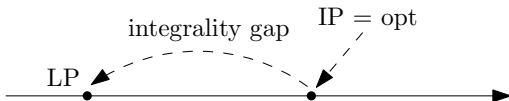
Def. The ratio between $\text{IP} = \text{opt}$ and LP is called the **integrality gap** of the LP relaxation.

IP

$$\begin{aligned} \min \quad & c^T x \\ Ax \geq & b \\ x \in & \{0, 1\}^n \end{aligned}$$

LP Relaxation

$$\begin{aligned} \min \quad & c^T x \\ Ax \geq & b \\ x \in & [0, 1]^n \end{aligned}$$



Def. The ratio between $\text{IP} = \text{opt}$ and LP is called the **integrality gap** of the LP relaxation.

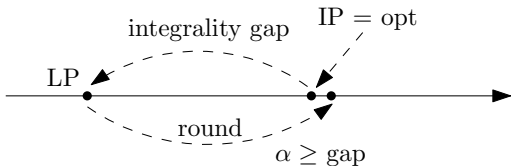
- The approximation ratio based on this analysis can not be better than the worst integrality gap.

IP

$$\begin{aligned} \min \quad & c^T x \\ Ax \geq b \\ x \in \{0, 1\}^n \end{aligned}$$

LP Relaxation

$$\begin{aligned} \min \quad & c^T x \\ Ax \geq b \\ x \in [0, 1]^n \end{aligned}$$



Def. The ratio between $\text{IP} = \text{opt}$ and LP is called the **integrality gap** of the LP relaxation.

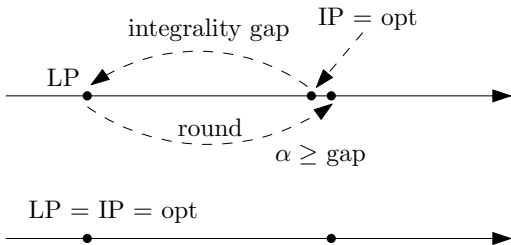
- The approximation ratio based on this analysis can not be better than the worst integrality gap.

IP

$$\begin{aligned} \min \quad & c^T x \\ Ax \geq b \\ x \in \{0, 1\}^n \end{aligned}$$

LP Relaxation

$$\begin{aligned} \min \quad & c^T x \\ Ax \geq b \\ x \in [0, 1]^n \end{aligned}$$



Def. The ratio between $\text{IP} = \text{opt}$ and LP is called the **integrality gap** of the LP relaxation.

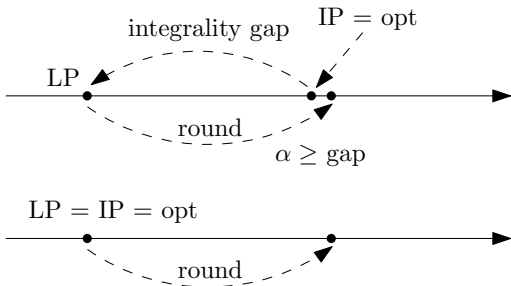
- The approximation ratio based on this analysis can not be better than the worst integrality gap.

IP

$$\begin{aligned} \min \quad & c^T x \\ Ax \geq b \\ x \in \{0, 1\}^n \end{aligned}$$

LP Relaxation

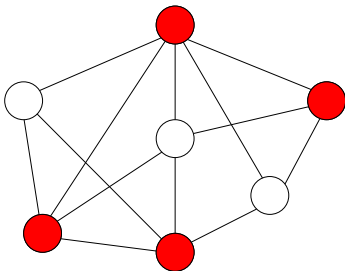
$$\begin{aligned} \min \quad & c^T x \\ Ax \geq b \\ x \in [0, 1]^n \end{aligned}$$



Def. The ratio between $\text{IP} = \text{opt}$ and LP is called the **integrality gap** of the LP relaxation.

- The approximation ratio based on this analysis can not be better than the worst integrality gap.

- 1 2-Approximation Algorithm for Weighted Vertex Cover
- 2 2-Approximation Algorithm for Unrelated Machine Scheduling
- 3 Congestion Minimization *



Weighted Vertex Cover Problem

Input: graph $G = (V, E)$, **vertex weights** $w \in \mathbb{Z}_{>0}^V$

Output: vertex cover S of G , to minimize $\sum_{v \in S} w_v$

- $x_v \in \{0, 1\}, \forall v \in V$: indicate if we include v in the vertex cover

Integer Program

$$\min \sum_{v \in V} w_v x_v$$

$$x_u + x_v \geq 1 \quad \forall (u, v) \in E$$

$$x_v \in \{0, 1\} \quad \forall v \in V$$

LP Relaxation

$$\min \sum_{v \in V} w_v x_v$$

$$x_u + x_v \geq 1 \quad \forall (u, v) \in E$$

$$x_v \in [0, 1] \quad \forall v \in V$$

- $x_v \in \{0, 1\}, \forall v \in V$: indicate if we include v in the vertex cover

Integer Program

$$\min \sum_{v \in V} w_v x_v$$

$$x_u + x_v \geq 1 \quad \forall (u, v) \in E$$

$$x_v \in \{0, 1\} \quad \forall v \in V$$

LP Relaxation

$$\min \sum_{v \in V} w_v x_v$$

$$x_u + x_v \geq 1 \quad \forall (u, v) \in E$$

$$x_v \in [0, 1] \quad \forall v \in V$$

- $IP :=$ value of integer program, $LP :=$ value of linear program
- $LP \leq IP = \text{opt}$

Rounding Algorithm

1: Solve LP to obtain solution $\{x_u^*\}_{u \in V}$

▷ So, $LP = \sum_{u \in V} w_u x_u^* \leq IP$

Rounding Algorithm

- 1: Solve LP to obtain solution $\{x_u^*\}_{u \in V}$
▷ So, $LP = \sum_{u \in V} w_u x_u^* \leq IP$
- 2: **return** $S := \{u \in V : x_u \geq 1/2\}$

Rounding Algorithm

- 1: Solve LP to obtain solution $\{x_u^*\}_{u \in V}$
▷ So, $LP = \sum_{u \in V} w_u x_u^* \leq IP$
- 2: **return** $S := \{u \in V : x_u \geq 1/2\}$

Lemma S is a vertex cover of G .

Proof.

Rounding Algorithm

- 1: Solve LP to obtain solution $\{x_u^*\}_{u \in V}$
▷ So, $LP = \sum_{u \in V} w_u x_u^* \leq IP$
- 2: **return** $S := \{u \in V : x_u \geq 1/2\}$

Lemma S is a vertex cover of G .

Proof.

- Consider any $(u, v) \in E$: we have $x_u^* + x_v^* \geq 1$

Rounding Algorithm

- 1: Solve LP to obtain solution $\{x_u^*\}_{u \in V}$
▷ So, $\text{LP} = \sum_{u \in V} w_u x_u^* \leq \text{IP}$
- 2: **return** $S := \{u \in V : x_u \geq 1/2\}$

Lemma S is a vertex cover of G .

Proof.

- Consider any $(u, v) \in E$: we have $x_u^* + x_v^* \geq 1$
- So, $x_u^* \geq 1/2$ or $x_v^* \geq 1/2 \implies u \in S$ or $v \in S$. □

Rounding Algorithm

- 1: Solve LP to obtain solution $\{x_u^*\}_{u \in V}$
▷ So, $\text{LP} = \sum_{u \in V} w_u x_u^* \leq \text{IP}$
- 2: **return** $S := \{u \in V : x_u \geq 1/2\}$

Lemma S is a vertex cover of G .

Rounding Algorithm

- 1: Solve LP to obtain solution $\{x_u^*\}_{u \in V}$
▷ So, $\text{LP} = \sum_{u \in V} w_u x_u^* \leq \text{IP}$
- 2: **return** $S := \{u \in V : x_u \geq 1/2\}$

Lemma S is a vertex cover of G .

Lemma $\text{cost}(S) := \sum_{u \in S} w_u \leq 2 \cdot \text{LP}$.

Rounding Algorithm

- 1: Solve LP to obtain solution $\{x_u^*\}_{u \in V}$
 $\triangleright$ So, $\text{LP} = \sum_{u \in V} w_u x_u^* \leq \text{IP}$
- 2: **return** $S := \{u \in V : x_u \geq 1/2\}$

Lemma S is a vertex cover of G .

Lemma $\text{cost}(S) := \sum_{u \in S} w_u \leq 2 \cdot \text{LP}$.

Proof.

$$\begin{aligned} \text{cost}(S) &= \sum_{u \in S} w_u \leq \sum_{u \in S} w_u \cdot 2x_u^* = 2 \sum_{u \in S} w_u \cdot x_u^* \\ &\leq 2 \sum_{u \in V} w_u \cdot x_u^* = 2 \cdot \text{LP}. \end{aligned}$$

□

Rounding Algorithm

- 1: Solve LP to obtain solution $\{x_u^*\}_{u \in V}$
▷ So, $\text{LP} = \sum_{u \in V} w_u x_u^* \leq \text{IP}$
- 2: **return** $S := \{u \in V : x_u \geq 1/2\}$

Lemma S is a vertex cover of G .

Lemma $\text{cost}(S) := \sum_{u \in S} w_u \leq 2 \cdot \text{LP}$.

Theorem The algorithm is a 2-approximation algorithm for weighted vertex cover.

Rounding Algorithm

- 1: Solve LP to obtain solution $\{x_u^*\}_{u \in V}$
 $\triangleright$ So, $\text{LP} = \sum_{u \in V} w_u x_u^* \leq \text{IP}$
- 2: **return** $S := \{u \in V : x_u \geq 1/2\}$

Lemma S is a vertex cover of G .

Lemma $\text{cost}(S) := \sum_{u \in S} w_u \leq 2 \cdot \text{LP}$.

Theorem The algorithm is a 2-approximation algorithm for weighted vertex cover.

Proof.

$$\text{cost}(S) \leq 2 \cdot \text{LP} \leq 2 \cdot \text{IP} = 2 \cdot (\text{optimum value})$$



Outline

- 1 2-Approximation Algorithm for Weighted Vertex Cover
- 2 2-Approximation Algorithm for Unrelated Machine Scheduling
- 3 Congestion Minimization *

Unrelated Machine Scheduling

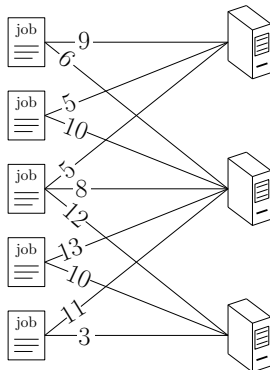
Input: $J, |J| = n$: jobs

$M, |M| = m$: machines

p_{ij} : processing time of
job j on machine i

Output: assignment $\sigma : J \mapsto M$:,
so as to minimize
makespan:

$$\max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_{ij}$$



Unrelated Machine Scheduling

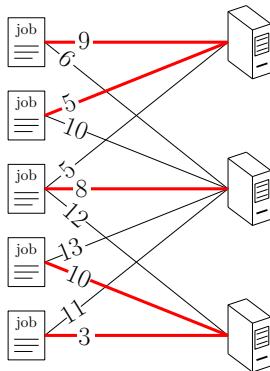
Input: $J, |J| = n$: jobs

$M, |M| = m$: machines

p_{ij} : processing time of
job j on machine i

Output: assignment $\sigma : J \mapsto M$:,
so as to minimize
makespan:

$$\max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_{ij}$$



Unrelated Machine Scheduling

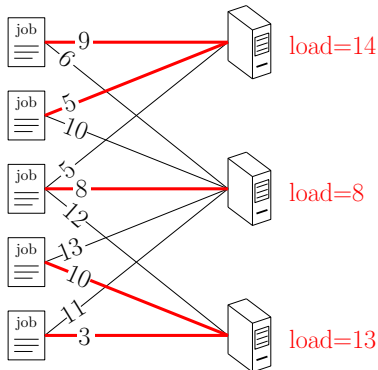
Input: $J, |J| = n$: jobs

$M, |M| = m$: machines

p_{ij} : processing time of
job j on machine i

Output: assignment $\sigma : J \mapsto M$:,
so as to minimize
makespan:

$$\max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_{ij}$$



Unrelated Machine Scheduling

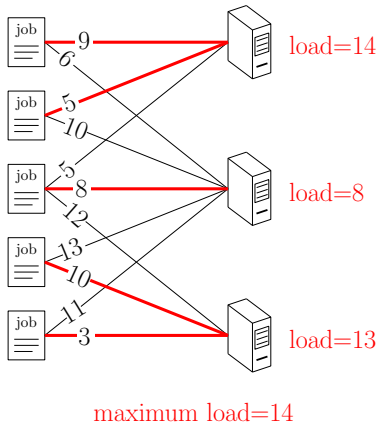
Input: $J, |J| = n$: jobs

$M, |M| = m$: machines

p_{ij} : processing time of
job j on machine i

Output: assignment $\sigma : J \mapsto M$:,
so as to minimize
makespan:

$$\max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_{ij}$$



- Assumption: we are given a target makespan T , and $p_{ij} \in [0, T] \cup \{\infty\}$

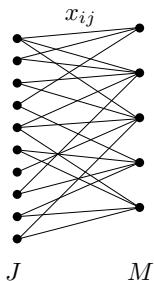
- Assumption: we are given a target makespan T , and $p_{ij} \in [0, T] \cup \{\infty\}$
- x_{ij} : fraction of j assigned to i

$$\begin{aligned}\sum_i x_{ij} &= 1 & \forall j \in J \\ \sum_j p_{ij} x_{ij} &\leq T & \forall i \in M \\ x_{ij} &\geq 0 & \forall ij\end{aligned}$$

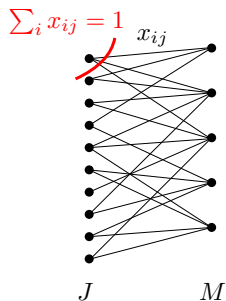
- Assumption: we are given a target makespan T , and $p_{ij} \in [0, T] \cup \{\infty\}$
- x_{ij} : fraction of j assigned to i

$$\begin{aligned}\sum_i x_{ij} &= 1 & \forall j \in J \\ \sum_j p_{ij} x_{ij} &\leq T & \forall i \in M \\ x_{ij} &\geq 0 & \forall ij\end{aligned}$$

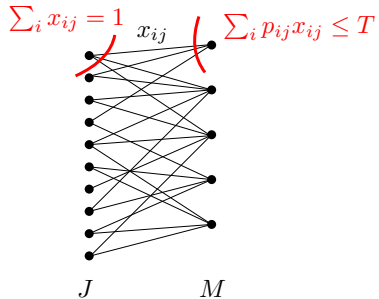
2-Approximate Rounding Algorithm of Shmoys-Tardos



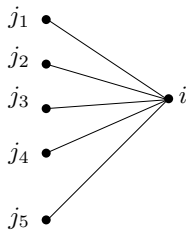
2-Approximate Rounding Algorithm of Shmoys-Tardos



2-Approximate Rounding Algorithm of Shmoys-Tardos

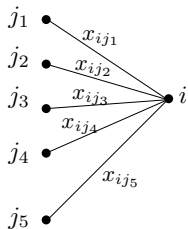


2-Approximate Rounding Algorithm of Shmoys-Tardos



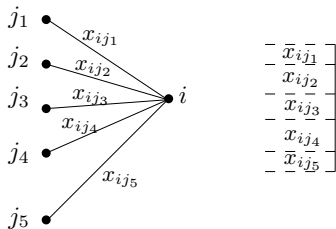
$$p_{ij_1} \geq p_{ij_2} \geq \dots \geq p_{ij_5}$$

2-Approximate Rounding Algorithm of Shmoys-Tardos



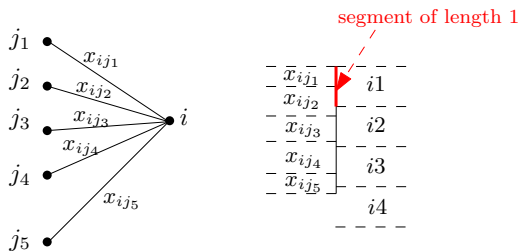
$$p_{ij_1} \geq p_{ij_2} \geq \dots \geq p_{ij_5}$$

2-Approximate Rounding Algorithm of Shmoys-Tardos



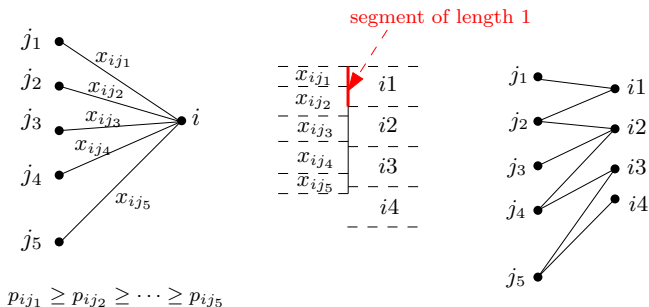
$$p_{ij_1} \geq p_{ij_2} \geq \cdots \geq p_{ij_5}$$

2-Approximate Rounding Algorithm of Shmoys-Tardos

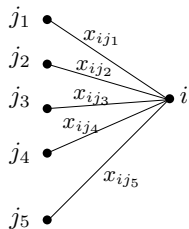


$$p_{ij_1} \geq p_{ij_2} \geq \dots \geq p_{ij_5}$$

2-Approximate Rounding Algorithm of Shmoys-Tardos

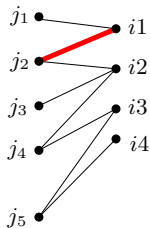


2-Approximate Rounding Algorithm of Shmoys-Tardos

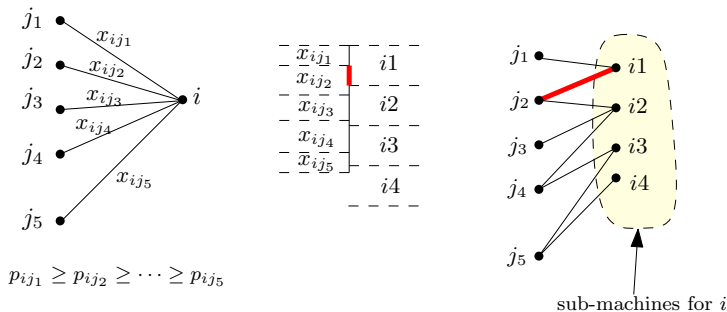


$$p_{ij_1} \geq p_{ij_2} \geq \dots \geq p_{ij_5}$$

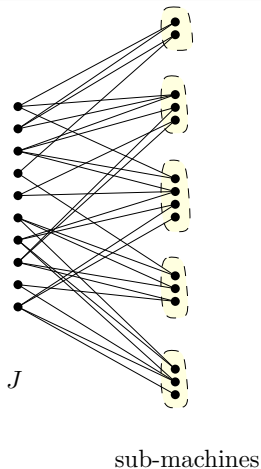
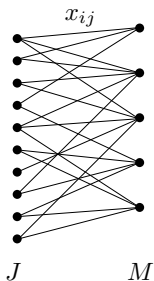
$\bar{x}_{ij_1}$	$i1$
$\bar{x}_{ij_2}$	$i2$
$\bar{x}_{ij_3}$	$i3$
$\bar{x}_{ij_4}$	$i4$
$\bar{x}_{ij_5}$	



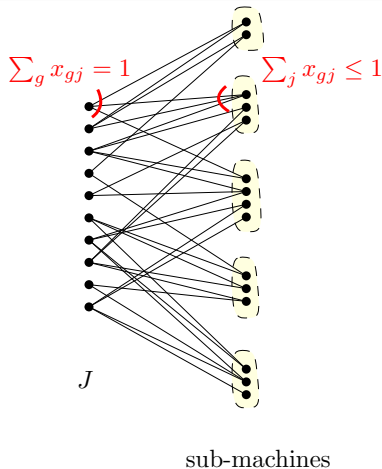
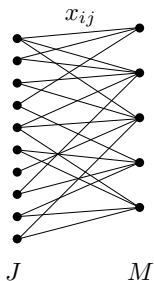
2-Approximate Rounding Algorithm of Shmoys-Tardos



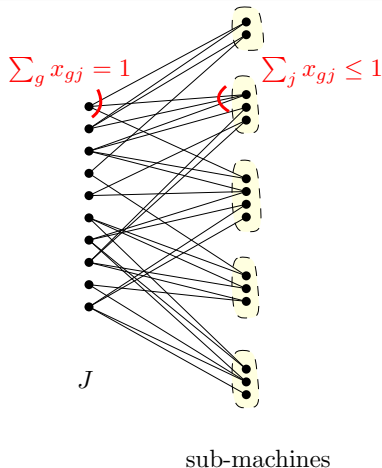
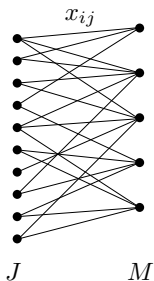
2-Approximate Rounding Algorithm of Shmoys-Tardos



2-Approximate Rounding Algorithm of Shmoys-Tardos



2-Approximate Rounding Algorithm of Shmoys-Tardos



Obs. x between J and sub-machines is a point in the bipartite-matching polytope, where all jobs in J are matched.

- Recall bipartite matching polytope is integral.

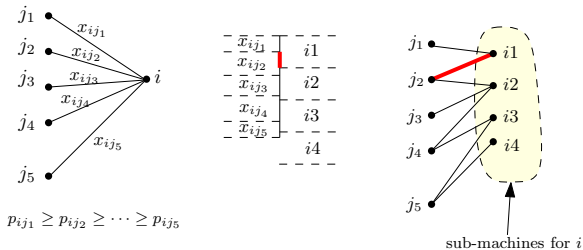
- Recall bipartite matching polytope is integral.
- x is a **convex combination** of matchings.

- Recall bipartite matching polytope is integral.
- x is a **convex combination** of matchings.
- Any matching in the combination covers all jobs J .

- Recall bipartite matching polytope is integral.
- x is a **convex combination** of matchings.
- Any matching in the combination covers all jobs J .

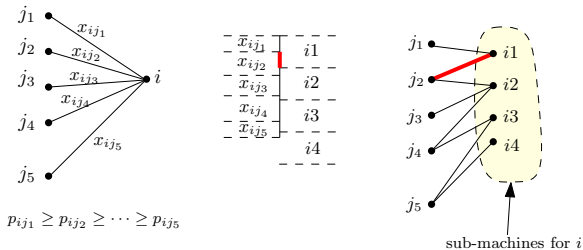
Lemma Any matching in the combination gives an schedule of makespan $\leq 2T$.

Lemma Any matching in the combination gives an schedule of makespan $\leq 2T$.



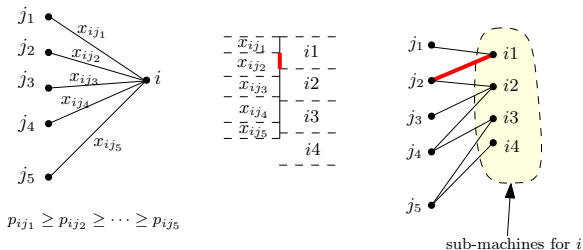
Proof.

Lemma Any matching in the combination gives an schedule of makespan $\leq 2T$.



Proof.

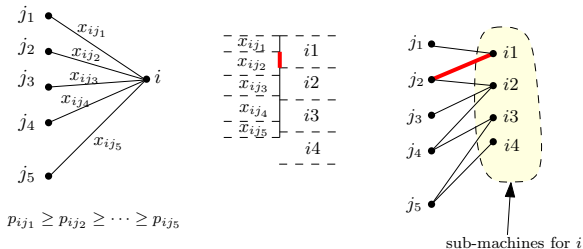
Lemma Any matching in the combination gives an schedule of makespan $\leq 2T$.



Proof.

- focus on machine i , let $i_1, i_2, \dots, i_a$ be the sub-machines for i

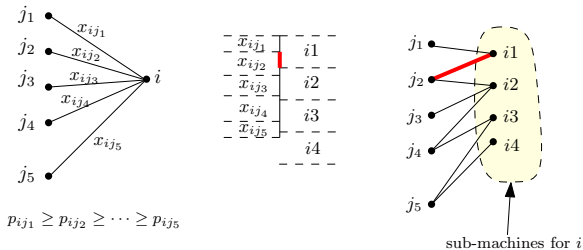
Lemma Any matching in the combination gives an schedule of makespan $\leq 2T$.



Proof.

- focus on machine i , let $i_1, i_2, \dots, i_a$ be the sub-machines for i
- assume job k_t is assigned to sub-machine i_t .

Lemma Any matching in the combination gives an schedule of makespan $\leq 2T$.



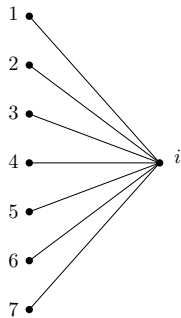
Proof.

- focus on machine i , let $i_1, i_2, \dots, i_a$ be the sub-machines for i
- assume job k_t is assigned to sub-machine i_t .

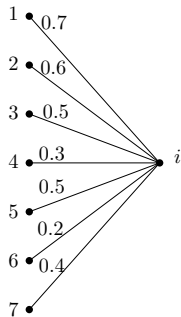
$$\begin{aligned}
 (\text{load on } i) &= \sum_{t=1}^a p_{ik_t} \leq p_{ik_1} + \sum_{t=2}^a \sum_j x_{i_{t-1}j} \cdot p_{ij} \\
 &\leq p_{ik_1} + \sum_j x_{ij} p_{ij} \leq T + T = 2T.
 \end{aligned}$$

□

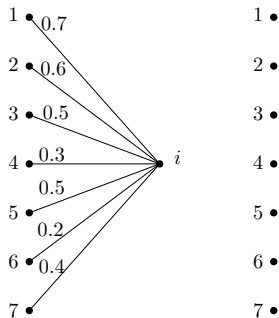
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:



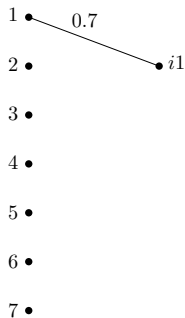
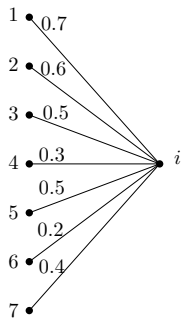
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:



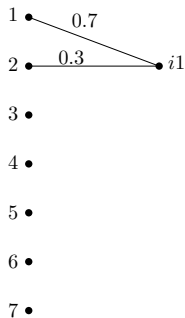
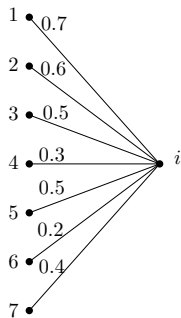
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:



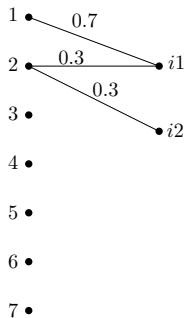
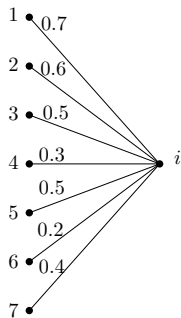
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:



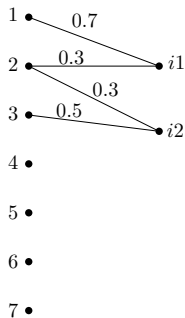
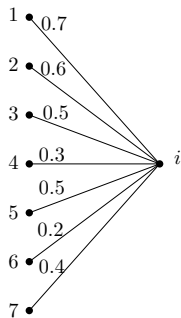
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:



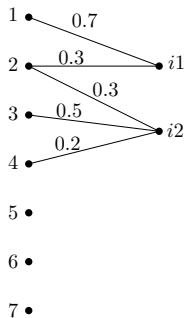
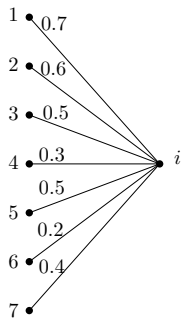
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:



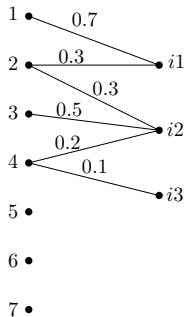
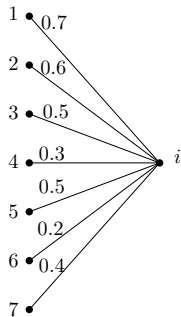
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:



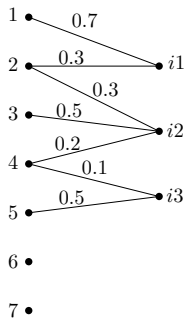
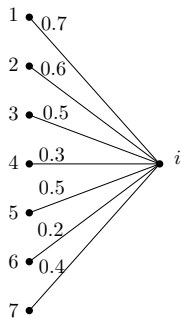
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:



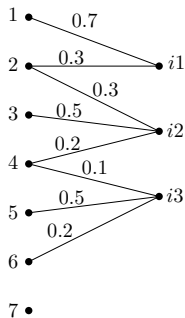
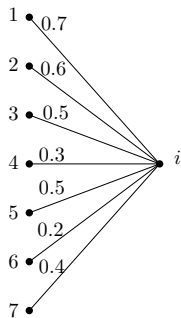
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:



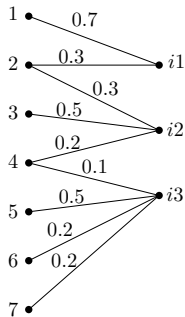
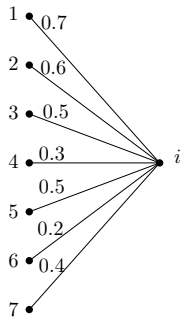
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:



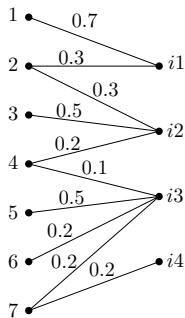
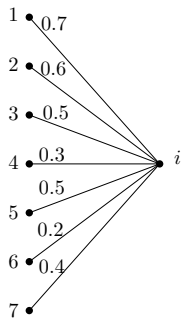
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:



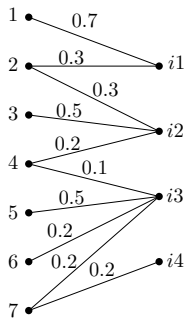
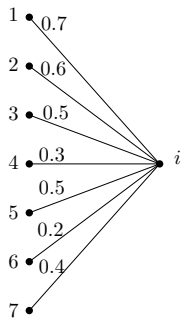
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:



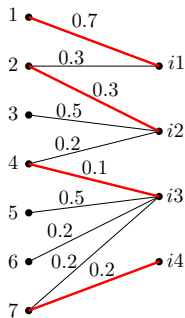
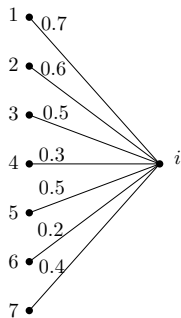
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:



- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:
 - $1 \rightarrow i1, 2 \rightarrow i2$
 - $4 \rightarrow i3, 7 \rightarrow i4$



- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:
 - $1 \rightarrow i1, 2 \rightarrow i2$
 - $4 \rightarrow i3, 7 \rightarrow i4$



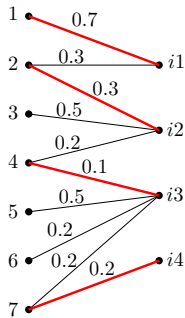
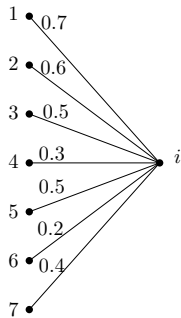
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:
 - $1 \rightarrow i1, 2 \rightarrow i2$
 - $4 \rightarrow i3, 7 \rightarrow i4$

$$p_1 \leq T$$

$$p_2 \leq 0.7p_1 + 0.3p_2$$

$$p_4 \leq 0.3p_2 + 0.5p_3 + 0.2p_4$$

$$p_7 \leq 0.1p_4 + 0.5p_5 + 0.2p_6 + 0.2p_7$$



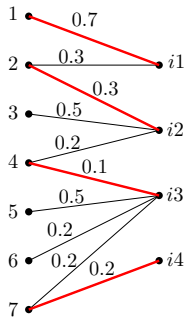
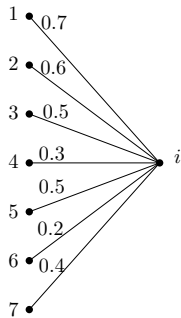
- fix i , use p_j for p_{ij}
- $p_1 \geq p_2 \geq \dots \geq p_7$
- worst case:
 - $1 \rightarrow i1, 2 \rightarrow i2$
 - $4 \rightarrow i3, 7 \rightarrow i4$

$$p_1 \leq T$$

$$p_2 \leq 0.7p_1 + 0.3p_2$$

$$p_4 \leq 0.3p_2 + 0.5p_3 + 0.2p_4$$

$$p_7 \leq 0.1p_4 + 0.5p_5 + 0.2p_6 + 0.2p_7$$



$$\begin{aligned}
 p_1 + p_2 + p_4 + p_7 &\leq T + (0.7p_1 + 0.3p_2) + (0.3p_2 + 0.5p_3 + 0.2p_4) \\
 &\quad + (0.1p_4 + 0.5p_5 + 0.2p_6 + 0.2p_7) \\
 &\leq T + (0.7p_1 + 0.6p_2 + 0.5p_3 + 0.3p_4 + 0.5p_5 + 0.2p_6 + 0.4p_7) \\
 &\leq T + T = 2T
 \end{aligned}$$

Outline

- 1 2-Approximation Algorithm for Weighted Vertex Cover
- 2 2-Approximation Algorithm for Unrelated Machine Scheduling
- 3 Congestion Minimization *

Congestion Minimization

Input: directed graph $G = (V, E)$

k pairs of vertices $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$

Output: find k paths: P_1 from s_1 to t_1 , P_2 from s_2 to t_2 , $\dots$, P_k from s_k to t_k .

$\text{cong}(e) := |\{i \in [k] : e \in P_i\}|.$

goal: minimize $\max_{e \in E} \text{cong}(e)$

Congestion Minimization

Input: directed graph $G = (V, E)$

k pairs of vertices $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$

Output: find k paths: P_1 from s_1 to t_1 , P_2 from s_2 to t_2 , $\dots$, P_k from s_k to t_k .

$\text{cong}(e) := |\{i \in [k] : e \in P_i\}|$.

goal: minimize $\max_{e \in E} \text{cong}(e)$

Q: What if $s_i = s$ for every $i \in [k]$?

Congestion Minimization

Input: directed graph $G = (V, E)$

k pairs of vertices $(s_1, t_1), (s_2, t_2), \dots, (s_k, t_k)$

Output: find k paths: P_1 from s_1 to t_1 , P_2 from s_2 to t_2 , $\dots$, P_k from s_k to t_k .

$\text{cong}(e) := |\{i \in [k] : e \in P_i\}|$.

goal: minimize $\max_{e \in E} \text{cong}(e)$

Q: What if $s_i = s$ for every $i \in [k]$?

A: (Single Source Single Sink) maximum flow problem. Can be solved exactly in polynomial time.

Linear Programming

- $\mathcal{P}_i$: set of paths from s_i to t_i
assume terminals are distinct
- $\mathcal{P} := \bigcup_{i \in [k]} \mathcal{P}_i$

Exponential Size LP

$$\min \quad C$$

$$\sum_{P \in \mathcal{P}_i} x_P = 1 \quad \forall i \in [k]$$

$$C \geq \sum_{P \in \mathcal{P}} x_P \quad \forall e \in E$$

$$x_P \geq 0 \quad \forall P \in \mathcal{P}$$

$$C \geq 1$$

- $x_{i,e}, i \in [k], e \in E$:
whether the path P_i uses
the edge e or not

Compact LP

$$\min \quad C$$

$$C \geq \sum_{i=1}^k x_{i,e} \quad \forall e \in E$$

$$C \geq 1$$

(*): $\forall i \in [k]$: capacities
 $(x_{i,e})_{e \in E}$ support 1 unit flow
from s_i to t_i

Equivalent Polynomial-Sized LP

- (*) can be checked using ellipsoid method, or the following LP network flow

Constraints (*) for a fixed i

$$\sum_{e \in \delta^{\text{out}}(v)} f_{i,e} - \sum_{e \in \delta^{\text{in}}(v)} f_{i,e} = \begin{cases} 1 & v = s_i \\ -1 & v = t_i \\ 0 & v \in V \setminus \{s_i, t_i\} \end{cases}$$
$$f_{i,e} \in [0, x_{i,e}], e \in E$$

Equivalent Polynomial-Sized LP

- (*) can be checked using ellipsoid method, or the following LP network flow

Constraints (*) for a fixed i

$$\sum_{e \in \delta^{\text{out}}(v)} f_{i,e} - \sum_{e \in \delta^{\text{in}}(v)} f_{i,e} = \begin{cases} 1 & v = s_i \\ -1 & v = t_i \\ 0 & v \in V \setminus \{s_i, t_i\} \end{cases}$$
$$f_{i,e} \in [0, x_{i,e}], e \in E$$

Lemma The Exponential-Size LP and the Compact LP for congestion minimization are equivalent.

- Easy direction:
solution for exponential-size LP $\implies$ solution for compact LP

Hard Direction: Solution for Compact LP $\implies$ Solution for Exponential-Size LP

- (*) is feasible: in the digraph G with source s_i , sink t_i and edge capacities $x_{i,e}$, the maximum flow has value at least 1.
- We can find $(y_P \geq 0)_{P \in \mathcal{P}_i}$ such that

$$\sum_{P \in \mathcal{P}_i: P \ni i} y_P \leq x_{i,e}, \forall e \in E \quad \text{and} \quad \sum_{P \in \mathcal{P}_i} y_P = 1$$

- $(y_P)_{P \in \mathcal{P}}$ is a solution for exponential size LP.

Hard Direction: Solution for Compact LP $\implies$ Solution for Exponential-Size LP

- (*) is feasible: in the digraph G with source s_i , sink t_i and edge capacities $x_{i,e}$, the maximum flow has value at least 1.
- We can find $(y_P \geq 0)_{P \in \mathcal{P}_i}$ such that

$$\sum_{P \in \mathcal{P}_i: P \ni i} y_P \leq x_{i,e}, \forall e \in E \quad \text{and} \quad \sum_{P \in \mathcal{P}_i} y_P = 1$$

- $(y_P)_{P \in \mathcal{P}}$ is a solution for exponential size LP.
- We assume we are given $(y_P)_{P \in \mathcal{P}}$, using the sparse representation.

Rounding Algorithm

- 1: **for** every $i \leftarrow 1$ to k **do**
- 2: independently and randomly choose P_i so that

$$\Pr[P_i = P] = x_P, \forall P \in \mathcal{P}_i.$$

- 3: **return** $P_1, P_2, \dots, P_k$

Rounding Algorithm

- 1: **for** every $i \leftarrow 1$ to k **do**
- 2: independently and randomly choose P_i so that

$$\Pr[P_i = P] = x_P, \forall P \in \mathcal{P}_i.$$

- 3: **return** $P_1, P_2, \dots, P_k$

Analysis for a fixed $e \in E$

- $\Pr[e \in P_i] = x_{i,e} := \sum_{P \in \mathcal{P}_i: P \ni e} x_P$
- $\sum_{i \in [k]} x_{i,e} \leq C$
- Let $X_i \in \{0, 1\}$ indicate if $e \in P_i$
- $\mathbb{E}[X_i] = x_{i,e}$
- $\text{cong}(e) = \sum_{i \in [k]} X_i$

Using Chernoff Bound:

$$\begin{aligned}\Pr \left[\sum_{i \in [k]} X_i \geq (1 + \delta)C \right] &\leq \left(\frac{e^\delta}{(1 + \delta)^{1+\delta}} \right)^C \\ &\leq \frac{e^\delta}{(1 + \delta)^{1+\delta}} \quad \text{since } C \geq 1\end{aligned}$$

Using Chernoff Bound:

$$\begin{aligned}\Pr \left[\sum_{i \in [k]} X_i \geq (1 + \delta)C \right] &\leq \left(\frac{e^\delta}{(1 + \delta)^{1+\delta}} \right)^C \\ &\leq \frac{e^\delta}{(1 + \delta)^{1+\delta}} \quad \text{since } C \geq 1\end{aligned}$$

- We need to choose a large enough δ so that $\frac{e^\delta}{(1+\delta)^{1+\delta}} \leq \frac{1}{2n^2}$,
how big should δ be?

Using Chernoff Bound:

$$\begin{aligned}\Pr \left[\sum_{i \in [k]} X_i \geq (1 + \delta)C \right] &\leq \left(\frac{e^\delta}{(1 + \delta)^{1+\delta}} \right)^C \\ &\leq \frac{e^\delta}{(1 + \delta)^{1+\delta}} \quad \text{since } C \geq 1\end{aligned}$$

- We need to choose a large enough δ so that $\frac{e^\delta}{(1+\delta)^{1+\delta}} \leq \frac{1}{2n^2}$, how big should δ be?
- To get an estimate, we replace e^δ with 1, and $1 + \delta$ with δ
- So, we need $\frac{1}{\delta^\delta} = \frac{1}{2n^2}$.
- $\delta = O\left(\frac{\log n}{\log \log n}\right)$ suffices.

- For some $\delta = O(\frac{\log n}{\log \log n})$, we have
 $\Pr[\text{cong}(e) \geq (1 + \delta)C] \leq \frac{1}{2n^2}.$
- Using Union Bound over all edges $e \in E$

$$\Pr[\exists e \in E, \text{cong}(e) \geq (1 + \delta)C] \leq \frac{1}{2n^2} \cdot m \leq \frac{1}{2}$$

$$\Pr[\forall e \in E, \text{cong}(e) < (1 + \delta)C] \geq 1 - \frac{1}{2} = \frac{1}{2}$$

- Remarks: the approximation ratio is as bad as $O(\frac{\log n}{\log \log n})$ only when C is a constant.
- As C becomes bigger, the ratio becomes better.
- If $C = \Theta(\log n)$, then the approximation ratio can be $O(1)$.
- The algorithm can be **derandomized** using the idea of conditional expectation.

Summary

- 2-approximation algorithm for weighted vertex cover
- 2-approximation for unrelated machine scheduling
- $O\left(\frac{\log n}{\log \log n}\right)$ -approximation for congestion minimization