On \((1, \epsilon)\)-Restricted Assignment Makespan Minimization

Shi Li, TTIC

Joint work with Deeparnab Chakrabarty and Sanjeev Khanna

SODA, Jan 5, 2015
Outline

1. Introduction
   - Restricted Makespan Minimization Problem
   - Our Result

2. $(2 - \delta^*)$-Approximation for $(1, \epsilon)$-Restricted Assignment
### Restricted Assignment Makespan Minimization

**Given:**
- $M$: $m$ machines
- $J$: $n$ jobs
- $p_j$: processing time of job $j$
- $M_j \subseteq M$: machines $j$ can be assigned to

**Goal:** assign jobs to machines, minimize makespan: find $\sigma: J \rightarrow M$ such that $\sigma(j) \in M_j$, $\forall j \in J$

**minimize**

$$\max_i \sum_{j \in \sigma^{-1}(i)} p_j$$

**Known Results**
- 2-approximation [LST90]
- 3/2-hardness of approximation

**On $(1, \epsilon)$-Restricted Assignment Makespan Minimization**
Restricted Assignment Makespan Minimization

Given:

- $M$: $m$ machines

Known Results

- 2-approximation [LST90]
- 3/2-hardness of approximation
Restricted Assignment Makespan Minimization

Given:
- $M$: $m$ machines
- $J$: $n$ jobs

Known Results
- 2-approximation [LST90]
- 3/2-hardness of approximation
Restricted Assignment Makespan Minimization

Given:
- \( M: m \) machines
- \( J: n \) jobs
- \( p_j: \) processing time of job \( j \)

<table>
<thead>
<tr>
<th>Machines</th>
<th>Jobs</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
</tr>
</tbody>
</table>

Known Results
- 2-approximation [LST90]
- \(3/2\)-hardness of approximation
Restricted Assignment Makespan Minimization

Given:

- \( M \): \( m \) machines
- \( J \): \( n \) jobs
- \( p_j \): processing time of job \( j \)
- \( M_j \subseteq M \): machines \( j \) can be assigned to

Our Result

Known Results

- 2-approximation [LST90]
- 3/2-hardness of approximation
Restricted Assignment Makespan Minimization

Given:
- **M**: *m* machines
- **J**: *n* jobs
- **p_j**: processing time of job *j*
- **M_j ⊆ M**: machines *j* can be assigned to

Goal: assign jobs to machines, minimize makespan = max. load:
Restricted Assignment Makespan Minimization

Given:
- \( M: m \) machines
- \( J: n \) jobs
- \( p_j: \) processing time of job \( j \)
- \( M_j \subseteq M: \) machines \( j \) can be assigned to

Goal: assign jobs to machines, minimize makespan = max. load:
- find \( \sigma: J \rightarrow M \) such that \( \sigma(j) \in M_j, \forall j \in J \)
Restricted Assignment Makespan Minimization

Given:
- $M$: $m$ machines
- $J$: $n$ jobs
- $p_j$: processing time of job $j$
- $M_j \subseteq M$: machines $j$ can be assigned to

Goal: assign jobs to machines, minimize makespan $= \max$ load:
- find $\sigma: J \rightarrow M$ such that $\sigma(j) \in M_j$, $\forall j \in J$
- minimize $\max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_j$
Restricted Assignment Makespan Minimization

Given:
- \( \mathcal{M} \): \( m \) machines
- \( \mathcal{J} \): \( n \) jobs
- \( p_j \): processing time of job \( j \)
- \( \mathcal{M}_j \subseteq \mathcal{M} \): machines \( j \) can be assigned to

Goal: assign jobs to machines, minimize makespan = \( \max \) load:
- find \( \sigma: \mathcal{J} \rightarrow \mathcal{M} \) such that \( \sigma(j) \in \mathcal{M}_j, \forall j \in \mathcal{J} \)
- minimize \( \max_{i \in \mathcal{M}} \sum_{j \in \sigma^{-1}(i)} p_j \)
Restricted Assignment Makespan Minimization

Given:
- \( M \): \( m \) machines
- \( J \): \( n \) jobs
- \( p_j \): processing time of job \( j \)
- \( M_j \subseteq M \): machines \( j \) can be assigned to

Goal: assign jobs to machines, minimize makespan = max. load:
- find \( \sigma : J \rightarrow M \) such that \( \sigma(j) \in M_j, \forall j \in J \)
- minimize \( \max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_j \)

Known Results

Known Results

2-approximation [LST90]
3/2-hardness of approximation

Shi Li, TTIC
On (1, \( \epsilon \))-Restricted Assignment Makespan Minimization
Restricted Assignment Makespan Minimization

Given:
- \( M \): \( m \) machines
- \( J \): \( n \) jobs
- \( p_j \): processing time of job \( j \)
- \( M_j \subseteq M \): machines \( j \) can be assigned to

Goal: assign jobs to machines, minimize makespan = max. load:
- find \( \sigma : J \rightarrow M \) such that \( \sigma(j) \in M_j, \forall j \in J \)
- minimize \( \max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_j \)

Known Results
- 2-approximation [LST90]
Restricted Assignment Makespan Minimization

Given:
- \( M \): \( m \) machines
- \( J \): \( n \) jobs
- \( p_j \): processing time of job \( j \)
- \( M_j \subseteq M \): machines \( j \) can be assigned to

Goal: assign jobs to machines, minimize makespan = \( \max \) load:
- find \( \sigma : J \rightarrow M \) such that \( \sigma(j) \in M_j, \forall j \in J \)
- minimize \( \max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_j \)

Known Results
- 2-approximation [LST90]
- 3/2-hardness of approximation
(1, \epsilon)-Restricted Assignment

Given:
- \( M: m \) machines
- \( J: n \) jobs
- \( p_j: \) processing time of job \( j \)
- \( M_j \subseteq M: \) machines \( j \) can be assigned to

Goal: assign jobs to machines, minimize makespan = \( \max \) load:
- find \( \sigma : J \rightarrow M \) such that \( \sigma(j) \in M_j, \forall j \in J \)
- minimize \( \max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_j \)

Known Results
- 2-approximation [LST90]
- 3/2-hardness of approximation
(1, ε)-Restricted Assignment

Given:
- \( M \): \( m \) machines
- \( J \): \( n \) jobs
- \( p_j \): processing time of job \( j \), \( p_j \in \{\epsilon, 1\} \)
- \( M_j \subseteq M \): machines \( j \) can be assigned to

Goal: assign jobs to machines, minimize makespan=\( \max \) load:
- find \( \sigma : J \rightarrow M \) such that \( \sigma(j) \in M_j \), \( \forall j \in J \)
- minimize \( \max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_j \)

Known Results
- 2-approximation [LST90]
- 3/2-hardness of approximation
(1, ε)-Restricted Assignment

Given:
- $M$: $m$ machines
- $J$: $n$ jobs
- $p_j$: processing time of job $j$, $p_j \in \{\epsilon, 1\}$
- $M_j \subseteq M$: machines $j$ can be assigned to

Goal: assign jobs to machines, minimize makespan = max. load:
- find $\sigma : J \rightarrow M$ such that $\sigma(j) \in M_j$, $\forall j \in J$
- minimize $\max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_j$

Known Results
- 2-approximation [LST90]
- 3/2-hardness of approximation
(1, \epsilon)-Restricted Assignment

Given:
- \( M: m \) machines
- \( J: n \) jobs
- \( p_j: \) processing time of job \( j, p_j \in \{\epsilon, 1\} \)
- \( M_j \subseteq M: \) machines \( j \) can be assigned to

Goal: assign jobs to machines, minimize makespan = max. load:
- find \( \sigma: J \rightarrow M \) such that \( \sigma(j) \in M_j, \forall j \in J \)
- minimize \( \max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_j \)

Known Results
- 2-approximation \([LST90]\)
- 3/2-hardness of approximation
(1, ε)-Restricted Assignment

Given:
- \( M: m \) machines
- \( J: n \) jobs
- \( p_j: \) processing time of job \( j, p_j \in \{\epsilon, 1\} \)
- \( M_j \subseteq M: \) machines \( j \) can be assigned to

Goal: assign jobs to machines, minimize makespan = max. load:
- find \( \sigma: J \rightarrow M \) such that \( \sigma(j) \in M_j, \forall j \in J \)
- minimize \( \max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_j \)

Known Results
- 2-approximation [LST90]
- 3/2-hardness of approximation
Given:
- $M$: $m$ machines
- $J$: $n$ jobs
- $p_j$: processing time of job $j$, $p_j \in \{\epsilon, 1\}$
- $M_j \subseteq M$: machines $j$ can be assigned to

Goal: assign jobs to machines, minimize makespan = max. load:
- find $\sigma: J \rightarrow M$ such that $\sigma(j) \in M_j$, $\forall j \in J$
- minimize $\max_{i \in M} \sum_{j \in \sigma^{-1}(i)} p_j$

Known Results
- 2-approximation [LST90]
- $7/6$-hardness of approximation
Estimation vs. Approximation

Estimation: only estimate the optimum makespan
Approximation: estimate the optimum makespan, give correspondent assignment

Estimation algorithm of [Sve11]:
33/17 -factor estimation

(1, ϵ)-restricted assignment:
(5/3 + ϵ) -factor estimation

Used configuration LP relaxation
No efficient algorithms to find the correspondent assignment
Estimation vs. Approximation

- **Estimation**: only estimate the optimum makespan
Estimation vs. Approximation

- **Estimation**: only estimate the optimum makespan
- **Approximation**: estimate the optimum makespan, give correspondent assignment
Estimation vs. Approximation

- **Estimation**: only estimate the optimum makespan
- **Approximation**: estimate the optimum makespan, give correspondent assignment

Estimation algorithm of [Sve11]
Estimation vs. Approximation

- **Estimation**: only estimate the optimum makespan
- **Approximation**: estimate the optimum makespan, give correspondent assignment

Estimation algorithm of [Sve11]

- restricted assignment: 33/17-factor estimation
**Estimation vs. Approximation**

- **Estimation**: only estimate the optimum makespan
- **Approximation**: estimate the optimum makespan, give correspondent assignment

**Estimation algorithm of [Sve11]**

- restricted assignment: 33/17-factor estimation
- \((1, \epsilon)\)-restricted assignment: \((5/3 + \epsilon)\)-factor estimation
Estimation vs. Approximation

- **Estimation**: only estimate the optimum makespan
- **Approximation**: estimate the optimum makespan, give correspondent assignment

Estimation algorithm of [Sve11]

- restricted assignment: \( \frac{33}{17} \)-factor estimation
- \((1, \epsilon)\)-restricted assignment: \( \left( \frac{5}{3} + \epsilon \right) \)-factor estimation
- Used *configuration LP relaxation*
Estimation vs. Approximation

- **Estimation**: only estimate the optimum makespan
- **Approximation**: estimate the optimum makespan, give correspondent assignment

Estimation algorithm of [Sve11]

- restricted assignment: \( \frac{33}{17} \)-factor estimation
- \((1, \epsilon)\)-restricted assignment: \( \left(\frac{5}{3} + \epsilon\right) \)-factor estimation
- Used **configuration LP relaxation**
- No efficient algorithms to find the correspondent assignment
Outline

1. Introduction
   - Restricted Makespan Minimization Problem
   - Our Result

2. $(2 - \delta^*)$-Approximation for $(1, \epsilon)$-Restricted Assignment
Our Result

Theorem

There is a polynomial time $(2 - \delta^*)$-approximation algorithm, for the $(1, \epsilon)$-restricted assignment makespan minimization, where $\delta^* > 0$ is a constant independent of $\epsilon$. 
Our Result

Theorem

There is a polynomial time \((2 - \delta^*)\)-approximation algorithm, for the \((1, \epsilon)\)-restricted assignment makespan minimization, where \(\delta^* > 0\) is a constant independent of \(\epsilon\).

Remark: there is a simple \((2 - \epsilon)\)-approximation
Our Result

Theorem

There is a polynomial time \((2 - \delta^*)\)-approximation algorithm, for the \((1, \epsilon)\)-restricted assignment makespan minimization, where \(\delta^* > 0\) is a constant independent of \(\epsilon\).

- Remark: there is a simple \((2 - \epsilon)\)-approximation
- Think of \(\epsilon = o(1)\)
Our Result

Theorem

There is a polynomial time $(2 - \delta^*)$-approximation algorithm, for the $(1, \epsilon)$-restricted assignment makespan minimization, where $\delta^* > 0$ is a constant independent of $\epsilon$.

Remark: there is a simple $(2 - \epsilon)$-approximation
Think of $\epsilon = o(1)$
Remark: $\delta^*$ is tiny
Our Result

Theorem

There is a polynomial time \((2 - \delta^*)\)-approximation algorithm, for the \((1, \epsilon)\)-restricted assignment makespan minimization, where \(\delta^* > 0\) is a constant independent of \(\epsilon\).

- Remark: there is a simple \((2 - \epsilon)\)-approximation
- Think of \(\epsilon = o(1)\)
- Remark: \(\delta^*\) is tiny

Why is \((1, \epsilon)\)-Restricted Case Interesting?
Our Result

Theorem

There is a polynomial time $(2 - \delta^*)$-approximation algorithm, for the $(1, \epsilon)$-restricted assignment makespan minimization, where $\delta^* > 0$ is a constant independent of $\epsilon$.

- Remark: there is a simple $(2 - \epsilon)$-approximation
- Think of $\epsilon = o(1)$
- Remark: $\delta^*$ is tiny

Why is $(1, \epsilon)$-Restricted Case Interesting?
- Simplest case: we did not know better-than-2-approximation
Our Result

Theorem

There is a polynomial time \((2 - \delta^*)\)-approximation algorithm, for the \((1, \epsilon)\)-restricted assignment makespan minimization, where \(\delta^* > 0\) is a constant independent of \(\epsilon\).

- Remark: there is a simple \((2 - \epsilon)\)-approximation
- Think of \(\epsilon = o(1)\)
- Remark: \(\delta^*\) is tiny

Why is \((1, \epsilon)\)-Restricted Case Interesting?

- Simplest case: we did not know better-than-2-approximation
- Captures difficulties of general problem
Our Result

Theorem

There is a polynomial time \((2 - \delta^*)\)-approximation algorithm, for the \((1, \epsilon)\)-restricted assignment makespan minimization, where \(\delta^* > 0\) is a constant independent of \(\epsilon\).

- Remark: there is a simple \((2 - \epsilon)\)-approximation
- Think of \(\epsilon = o(1)\)
- Remark: \(\delta^*\) is tiny

Why is \((1, \epsilon)\)-Restricted Case Interesting?

- Simplest case: we did not know better-than-2-approximation
- Captures difficulties of general problem
- \([Sve11]\) used this case to deliver ideas
Outline

1 Introduction

2 (2 − δ*)-Approximation for (1, ε)-Restricted Assignment
   - Natural LP and Compact LP Relaxations
   - (p, q)-Canonical Instance
   - Overview of Rounding Algorithm
Outline

1 Introduction

2 (2 − δ*)-Approximation for (1, ε)-Restricted Assignment
   - Natural LP and Compact LP Relaxations
   - (p, q)-Canonical Instance
   - Overview of Rounding Algorithm
Natural LP Relaxation

\[(2 - \delta^*)\text{- Approximation for (1, } \epsilon \text{)-Restricted Assignment}\]

\[\text{Natural LP and Compact LP Relaxations}\]

\[(p, q)\text{- Canonical Instance}\]

\[\text{Overview of Rounding Algorithm}\]
Natural LP Relaxation

- [LST90] based on natural LP relaxation
Natural LP Relaxation

- [LST90] based on natural LP relaxation
- $x_{i,j} \in [0, 1], i \in M, j \in J$: whether $j$ is assigned to $i$
Natural LP Relaxation

- [LST90] based on natural LP relaxation
- $x_{i,j} \in [0, 1], i \in M, j \in J$: whether $j$ is assigned to $i$
- $x_{i,j} = 0$ if $i \notin M_j$
Natural LP Relaxation

- [LST90] based on natural LP relaxation
- \( x_{i,j} \in [0, 1], i \in M, j \in J \): whether \( j \) is assigned to \( i \)
- \( x_{i,j} = 0 \) if \( i \notin M_j \)

minimize \( T \)
Natural LP Relaxation

- [LST90] based on natural LP relaxation
- \( x_{i,j} \in [0, 1], i \in M, j \in J \): whether \( j \) is assigned to \( i \)
- \( x_{i,j} = 0 \) if \( i \notin M_j \)

\[
\text{minimize } \quad T
\]
\[
\sum_{j \in J} p_j x_{i,j} \leq T \quad \forall i \in M
\]
\[
p_j \leq T \quad \forall j \in J
\]
Natural LP Relaxation

- [LST90] based on natural LP relaxation
- $x_{i,j} \in [0, 1], i \in M, j \in J$: whether $j$ is assigned to $i$
- $x_{i,j} = 0$ if $i \notin M$

minimize $T$

$$\sum_{j \in J} p_j x_{i,j} \leq T \quad \forall i \in M$$

$$p_j \leq T \quad \forall j \in J$$

$$\sum_{i \in M} x_{i,j} = 1, \quad \forall j \in J$$
Natural LP Relaxation

- [LST90] based on natural LP relaxation
- $x_{i,j} \in [0, 1], i \in M, j \in J$: whether $j$ is assigned to $i$
- $x_{i,j} = 0$ if $i \notin M_j$

**minimize** $T$

$$\sum_{j \in J} p_j x_{i,j} \leq T \quad \forall i \in M$$

$$p_j \leq T \quad \forall j \in J$$

$$\sum_{i \in M} x_{i,j} = 1, \quad \forall j \in J$$

- Integrality gap $= 2$
Natural LP Relaxation

- [LST90] based on natural LP relaxation
- \( x_{i,j} \in [0, 1], i \in M, j \in J \): whether \( j \) is assigned to \( i \)
- \( x_{i,j} = 0 \) if \( i \notin M_j \)

**minimize** \( T \)

\[
\sum_{j \in J} p_j x_{i,j} \leq T \quad \forall i \in M
\]

\[
p_j \leq T \quad \forall j \in J
\]

\[
\sum_{i \in M} x_{i,j} = 1, \quad \forall j \in J
\]

- Integrality gap = 2
Natural LP Relaxation

- \([\text{LST90}]\) based on natural LP relaxation
- \(x_{i,j} \in [0, 1], i \in M, j \in J\): whether \(j\) is assigned to \(i\)
- \(x_{i,j} = 0\) if \(i \notin M_j\)

\[
\text{minimize} \quad T \\
\sum_{j \in J} p_j x_{i,j} \leq T \quad \forall i \in M \\
p_j \leq T \quad \forall j \in J \\
\sum_{i \in M} x_{i,j} = 1, \quad \forall j \in J
\]

- Integrality gap = 2

\(1/\eta\) machines
Natural LP Relaxation

- [LST90] based on natural LP relaxation
- \( x_{i,j} \in [0, 1], i \in M, j \in J \): whether \( j \) is assigned to \( i \)
- \( x_{i,j} = 0 \) if \( i \notin M_j \)

\[
\text{minimize} \quad T \\
\sum_{j \in J} p_j x_{i,j} \leq T \quad \forall i \in M \\
p_j \leq T \quad \forall j \in J \\
\sum_{i \in M} x_{i,j} = 1, \quad \forall j \in J
\]

- Integrality gap = 2

(1 − \( \eta \))/\( \epsilon \) small jobs
Natural LP Relaxation

- **[LST90]** based on natural LP relaxation
- \( x_{i,j} \in [0, 1], i \in M, j \in J \): whether \( j \) is assigned to \( i \)
- \( x_{i,j} = 0 \) if \( i \notin M_j \)

\[
\begin{align*}
\text{minimize} & \quad T \\
\sum_{j \in J} p_j x_{i,j} & \leq T \quad \forall i \in M \\
p_j & \leq T \quad \forall j \in J \\
\sum_{i \in M} x_{i,j} & = 1, \quad \forall j \in J
\end{align*}
\]

- Integrality gap = 2
- **[Sve11]**: Configuration LP overcomes the integrality gap

\( \frac{(1 - \eta)}{\epsilon} \) small jobs
Compact LP for $(1, \epsilon)$-Restricted Case, \(\OPT = 1\)
Compact LP for \((1, \varepsilon)\)-Restricted Case, \(\text{OPT} = 1\)

- \(x_{i,j} \in [0, 1], i \in M, j \in J\): whether \(j\) is assigned to \(i\)
- \(x_{i,j} = 0\) if \(i \notin M_j\)

\[
\sum_{j \in J} p_j x_{i,j} \leq 1, \quad \forall i \in M
\]

\[
\sum_{i \in M} x_{i,j} = 1, \quad \forall j \in J
\]
Compact LP for $(1, \epsilon)$-Restricted Case, OPT = 1

- $x_{i,j} \in [0, 1], i \in M, j \in J$: whether $j$ is assigned to $i$
- $x_{i,j} = 0$ if $i \notin M_j$

\[
\sum_{j \in J} p_j x_{i,j} \leq 1, \quad \forall i \in M
\]
\[
\sum_{i \in M} x_{i,j} = 1, \quad \forall j \in J
\]
\[
\sum_{j \in J_{\text{big}}} x_{i,j} + x_{i,j'} \leq 1 \quad \forall i \in M, j' \in J_{\text{small}}
\]

$(1 - \eta)/\epsilon$ small jobs

Light job assignment

Integrality gap = $\frac{3}{2}$
Compact LP for $(1, \epsilon)$-Restricted Case, $\text{OPT} = 1$

- $x_{i,j} \in [0, 1], i \in M, j \in J$: whether $j$ is assigned to $i$
- $x_{i,j} = 0$ if $i \notin M$

\[
\sum_{j \in J} p_j x_{i,j} \leq 1, \quad \forall i \in M
\]

\[
\sum_{i \in M} x_{i,j} = 1, \quad \forall j \in J
\]

\[
\sum_{j \in J_{\text{big}}} x_{i,j} + x_{i,j'} \leq 1 \quad \forall i \in M, j' \in J_{\text{small}}
\]

- Light job assignment $\leq 1 - \eta \Rightarrow$ fractional solution invalid
Compact LP for $(1, \epsilon)$-Restricted Case, $\text{OPT} = 1$

- $x_{i,j} \in [0, 1], i \in M, j \in J$: whether $j$ is assigned to $i$
- $x_{i,j} = 0$ if $i \not\in M_j$

\[
\sum_{j \in J} p_j x_{i,j} \leq 1, \quad \forall i \in M
\]

\[
\sum_{i \in M} x_{i,j} = 1, \quad \forall j \in J
\]

\[
\sum_{j \in J_{\text{big}}} x_{i,j} + x_{i,j'} \leq 1 \quad \forall i \in M, j' \in J_{\text{small}}
\]

- Light job assignment $\leq 1 - \eta \Rightarrow$ fractional solution invalid
Compact LP for \((1, \epsilon)\)-Restricted Case, \(\text{OPT} = 1\)

- \(x_{i,j} \in [0, 1], i \in M, j \in J\): whether \(j\) is assigned to \(i\)
- \(x_{i,j} = 0\) if \(i \not\in M_j\)

\[
\sum_{j \in J} p_j x_{i,j} \leq 1, \quad \forall i \in M
\]

\[
\sum_{i \in M} x_{i,j} = 1, \quad \forall j \in J
\]

\[
\sum_{j \in J_{\text{big}}} x_{i,j} + x_{i,j'} \leq 1 \quad \forall i \in M, j' \in J_{\text{small}}
\]

- Light job assignment \(\leq 1 - \eta \Rightarrow\) fractional solution invalid
Compact LP for \((1, \epsilon)\)-Restricted Case, \(\text{OPT} = 1\)

- \(x_{i,j} \in [0, 1], i \in M, j \in J\): whether \(j\) is assigned to \(i\)
- \(x_{i,j} = 0\) if \(i \notin M_j\)

\[
\sum_{j \in J} p_j x_{i,j} \leq 1, \quad \forall i \in M
\]

\[
\sum_{i \in M} x_{i,j} = 1, \quad \forall j \in J
\]

\[
\sum_{j \in J_{\text{big}}} x_{i,j} + x_{i,j'} \leq 1 \quad \forall i \in M, j' \in J_{\text{small}}
\]

- Light job assignment \(\leq 1 - \eta \Rightarrow\) fractional solution invalid
- Integrrality gap = 3/2
Outline

1. Introduction

2. $(2 - \delta^*)$-Approximation for $(1, \epsilon)$-Restricted Assignment
   - Natural LP and Compact LP Relaxations
   - $(p, q)$-Canonical Instance
   - Overview of Rounding Algorithm
(\(p, q\))-Canonical Instance

machines  ■  ■  ■  ■  ■  ■  ■
\((p, q)\)-Canonical Instance

- Big job \(j\): "private" set \(M_j\) of \(p\) machines;
- \(\{M_j\}_{j \in J_{\text{big}}}\) form a partitioning of \(M\)
\( (p, q) \)-Canonical Instance

- Big job \( j \): “private” set \( M_j \) of \( p \) machines;
  - \( \{M_j\}_{j \in J_{\text{big}}} \) form a partitioning of \( M \)
- Small job has size \( 1/q \) (instead of \( \epsilon \))
(p, q)-Canonical Instance

- Big job \( j \): “private” set \( M_j \) of \( p \) machines;
  - \( \{M_j\}_{j \in J_{\text{big}}} \) form a partitioning of \( M \)
- Small job has size \( 1/q \) (instead of \( \epsilon \))
- Small job can be assigned to exactly 2 machines
(p, q)-Canonical Instance

- Big job $j$: "private" set $M_j$ of $p$ machines;
  - $\{M_j\}_{j \in J_{\text{big}}}$ form a partitioning of $M$
- Small job has size $1/q$ (instead of $\epsilon$)
- Small job can be assigned to exactly 2 machines
- Associated fractional assignment:
(p, q)-Canonical Instance

- Big job \( j \): “private” set \( M_j \) of \( p \) machines;
  - \( \{M_j\}_{j \in J_{\text{big}}} \) form a partitioning of \( M \)
- Small job has size \( 1/q \) (instead of \( \epsilon \))
- Small job can be assigned to exactly 2 machines
- Associated fractional assignment:
  - \( j \in J_{\text{big}}, \forall i \in M_j: x_{i,j} = 1/p \)
(p, q)-Canonical Instance

- Big job $j$: “private” set $M_j$ of $p$ machines;
  - $\{M_j\}_{j \in J_{\text{big}}}$ form a partitioning of $M$
- Small job has size $1/q$ (instead of $\epsilon$)
- Small job can be assigned to exactly 2 machines
- Associated fractional assignment:
  - $j \in J_{\text{big}}, \forall i \in M_j$: $x_{i,j} = 1/p$
  - small job $j$ with $M_j = \{i, i'\}$: $x_{i,j} = 1/p$ and $x_{i',j} = 1 - 1/p$
\((p, q)\)-Canonical Instance

- Big job \(j\): “private” set \(M_j\) of \(p\) machines;
  - \(\{M_j\}_{j \in J_{big}}\) form a partitioning of \(M\)
- Small job has size \(1/q\) (instead of \(\epsilon\))
- Small job can be assigned to exactly 2 machines
- Associated fractional assignment:
  - \(j \in J_{big}, \forall i \in M_j: x_{i,j} = 1/p\)
  - small job \(j\) with \(M_j = \{i, i'\}\): \(x_{i,j} = 1/p\) and \(x_{i',j} = 1 - 1/p\)
  - call \(j\) small job of type-\((i, i')\)
(\(p, q\))-Canonical Instance

- big jobs
- machines
- small jobs

- small job \(j\) of type-(\(i, i'\)) \(\Rightarrow\) edge \((i, i') \in E\)
(p, q)-Canonical Instance

- big jobs
- machines
- small jobs

- small job $j$ of type-$(i, i') \Rightarrow$ edge $(i, i') \in E$
\((p, q)\)-Canonical Instance

- big jobs
- machines
- small jobs

- small job \(j\) of type-\((i, i')\) \(\Rightarrow\) edge \((i, i') \in E\)
- Directed multi-graph \(G = (M, E)\) to denote small jobs
(p, q)-Canonical Instance

- small job $j$ of type-$(i, i') \Rightarrow$ edge $(i, i') \in E$
- Directed multi-graph $G = (M, E)$ to denote small jobs
- Instance specified by $(\{M_j\}_{j \in J_{\text{big}}}, G)$
\[(p, q)\]-Canonical Instance

- small job \( j \) of type-\((i, i')\) \( \Rightarrow \) edge \((i, i')\) \( \in E \)
- Directed multi-graph \( G = (M, E) \) to denote small jobs
- Instance specified by \( (\{M_j\}_{j \in J_{\text{big}}}, G) \)

\[
\frac{1}{p} + \frac{1}{q} \left[ \left( 1 - \frac{1}{p} \right) d_{\text{in}}(i) + \frac{1}{p} d_{\text{out}}(i) \right] \leq 1, \quad \forall i \in M
\]
Semi-integral assignment: big jobs integrally assigned, small jobs fractionally assigned
Semi-integral assignment: big jobs integrally assigned, small jobs fractionally assigned

Lemma (follows from [LST90])
Semi-integral assignment of makespan $T$
$\Rightarrow$ integral assignment of makespan $T + \epsilon$
• Semi-integral assignment: big jobs integrally assigned, small jobs fractionally assigned

**Lemma (follows from [LST90])**

Semi-integral assignment of makespan $T$  
\[ \Rightarrow \text{integral assignment of makespan } T + \epsilon \]

• Given assignment of big jobs:  
assigning small jobs = network-flow problem, easy
Good Assignment of big Jobs

- Semi-integral assignment: big jobs integrally assigned, small jobs fractionally assigned

Lemma (follows from [LST90])

**Semi-integral assignment of makespan $T$**

$\Rightarrow$ **integral assignment of makespan $T + \epsilon$**

- Given assignment of big jobs:
  assigning small jobs = network-flow problem, easy

Definition (Good Assignment of big Jobs)

An assignment $f : J_{\text{big}} \rightarrow M$ of big jobs is **good** if small jobs can be fractionally assigned so that the makespan is $2 - \delta$. 
Good Assignment for Canonical Instances

Theorem (From Hall’s Theorem)

An assignment \( f: J_{\text{big}} \rightarrow M \) is good iff

\[
\forall S \subseteq M: |S \cap f(J_{\text{big}})| + 1 \leq (2 - \delta^*) |S|.
\]

\((*)\)

Goal: find \( f: J_{\text{big}} \rightarrow M \) so that \((*)\) holds for every \( S \subseteq M \).
(2 − δ*)-Approximation for (1, ϵ)-Restricted Assignment

Good Assignment for Canonical Instances

- machines
- big jobs
- small jobs

Theorem (From Hall's Theorem)
An assignment \( f : J_{\text{big}} \rightarrow M \) is good iff
\[ |S \cap f(J_{\text{big}})| + 1 \leq (2 - \delta^*) |S| \]

Goal: find \( f : J_{\text{big}} \rightarrow M \) so that (*) holds for every \( S \subseteq M \).
Good Assignment for Canonical Instances

Theorem (From Hall’s Theorem)

An assignment \( f : J_{\text{big}} \rightarrow M \) is good iff \( \forall S \subseteq M \):

\[
|S \cap f(J_{\text{big}})| + \frac{1}{q} E_{S,S} \leq (2 - \delta)|S|.
\]

(*)
Good Assignment for Canonical Instances

Theorem (From Hall’s Theorem)

An assignment $f : J_{\text{big}} \rightarrow M$ is good iff $\forall S \subseteq M$:

$$|S \cap f(J_{\text{big}})| + \frac{1}{q} E_{S,S} \leq (2 - \delta)|S|. \quad (*)$$
Good Assignment for Canonical Instances

**Theorem (From Hall’s Theorem)**

An assignment \( f : J_{\text{big}} \rightarrow M \) is good iff \( \forall S \subseteq M : \)

\[
|S \cap f(J_{\text{big}})| + \frac{1}{q} E_{S,S} \leq (2 - \delta)|S|. \tag{*}
\]

_goal: find \( f : J_{\text{big}} \rightarrow M \) so that \( (*) \) holds for every \( S \subseteq M \).
Outline

1 Introduction

2 (2 − δ*)-Approximation for (1, ε)-Restricted Assignment
   - Natural LP and Compact LP Relaxations
   - (p, q)-Canonical Instance
   - Overview of Rounding Algorithm
Overview of Algorithm

1. Reduce to \((p, q)\)-canonical instance
2. Reduce \(p\) and \(q\) if
   - \(p > \max\{q, q_0\}\), then reduce to a \((p/2, q_0)\)-instance
   - \(q > \max\{p, q_0\}\), then reduce to a \((p, q/2)\)-instance
   Good \(f\) for new instance \(\Rightarrow\) good \(f\) for original instance
3. Solve a \((p, q)\)-canonical instance with \(p \leq q_0, q \leq q_0\)
   Can easily obtain \((2 - 1/q_0)\)-makespan
   Not enough: lost a factor of \(\tilde{\Theta}(1/\sqrt{q_0})\) in reductions
   Our goal: \((2 - \Omega(1/poly \log q_0))\)-makespan

Use Lovasz Local Lemma many times
Overview of Algorithm

1. Reduce to \((p, q)\)-canonical instance
Overview of Algorithm

1. Reduce to \((p, q)\)-canonical instance
2. Reduce \(p\) and \(q\)
Overview of Algorithm

1. Reduce to \((p, q)\)-canonical instance
2. Reduce \(p\) and \(q\)
   - \(q_0 = O(1)\): a large constant
Overview of Algorithm

1. Reduce to \((p, q)\)-canonical instance
2. Reduce \(p\) and \(q\)
   - \(q_0 = O(1)\): a large constant
   - If \(p > \max\{q, q_0\}\), then reduce to a \((p/2, q)\)-instance

\[(2 - \delta^*)\text{-Approximation for } (1, \epsilon)\text{-Restricted Assignment} \]
Overview of Algorithm

1. Reduce to \((p, q)\)-canonical instance

2. Reduce \(p\) and \(q\)
   - \(q_0 = O(1)\): a large constant
   - If \(p > \max\{q, q_0\}\), then reduce to a \((p/2, q)\)-instance
   - If \(q > \max\{p, q_0\}\), then reduce to a \((p, q/2)\)-instance
Overview of Algorithm

1. Reduce to \((p, q)\)-canonical instance
2. Reduce \(p\) and \(q\)
   - \(q_0 = O(1)\): a large constant
   - If \(p > \max\{q, q_0\}\), then reduce to a \((p/2, q)\)-instance
   - If \(q > \max\{p, q_0\}\), then reduce to a \((p, q/2)\)-instance
   - Good \(f\) for new instance \(\Rightarrow\) good \(f\) for original instance
Overview of Algorithm

1. Reduce to \((p, q)\)-canonical instance

2. Reduce \(p\) and \(q\)
   - \(q_0 = O(1)\): a large constant
   - If \(p > \max\{q, q_0\}\), then reduce to a \((p/2, q)\)-instance
   - If \(q > \max\{p, q_0\}\), then reduce to a \((p, q/2)\)-instance
   - Good \(f\) for new instance \(\Rightarrow\) good \(f\) for original instance

3. Solve a \((p, q)\)-canonical instance with \(p \leq q_0, q \leq q_0\)
Overview of Algorithm

1. Reduce to \((p, q)\)-canonical instance
2. Reduce \(p\) and \(q\)
   - \(q_0 = O(1)\): a large constant
   - If \(p > \max\{q, q_0\}\), then reduce to a \((p/2, q)\)-instance
   - If \(q > \max\{p, q_0\}\), then reduce to a \((p, q/2)\)-instance
   - Good \(f\) for new instance \(\Rightarrow\) good \(f\) for original instance
3. Solve a \((p, q)\)-canonical instance with \(p \leq q_0, q \leq q_0\)
   - Can easily obtain \((2 - 1/q_0)\)-makespan

\((2 - \delta^*)\)-Approximation for \((1, \epsilon)\)-Restricted Assignment

Natural LP and Compact LP Relaxations

\((p, q)\)-Canonical Instance

Overview of Rounding Algorithm
Overview of Algorithm

1. Reduce to \((p, q)\)-canonical instance

2. Reduce \(p\) and \(q\)
   - \(q_0 = O(1)\): a large constant
   - If \(p > \max\{q, q_0\}\), then reduce to a \((p/2, q)\)-instance
   - If \(q > \max\{p, q_0\}\), then reduce to a \((p, q/2)\)-instance
   - Good \(f\) for new instance \(\Rightarrow\) good \(f\) for original instance

3. Solve a \((p, q)\)-canonical instance with \(p \leq q_0, q \leq q_0\)
   - Can easily obtain \((2 - 1/q_0)\)-makespan
   - Not enough: lost a factor of \(\tilde{\Theta}(1/\sqrt{q_0})\) in reductions
Overview of Algorithm

1. Reduce to \((p, q)\)-canonical instance
2. Reduce \(p\) and \(q\)
   - \(q_0 = O(1)\): a large constant
   - If \(p > \max\{q, q_0\}\), then reduce to a \((p/2, q)\)-instance
   - If \(q > \max\{p, q_0\}\), then reduce to a \((p, q/2)\)-instance
   - Good \(f\) for new instance \(\Rightarrow\) good \(f\) for original instance
3. Solve a \((p, q)\)-canonical instance with \(p \leq q_0, q \leq q_0\)
   - Can easily obtain \((2 - 1/q_0)\)-makespan
   - Not enough: lost a factor of \(\tilde{\Theta}(1/\sqrt{q_0})\) in reductions
   - Our goal: \((2 - \Omega(1/poly \log q_0))\)-makespan
Overview of Algorithm

1. Reduce to \((p, q)\)-canonical instance
2. Reduce \(p\) and \(q\)
   - \(q_0 = O(1)\): a large constant
   - If \(p > \max\{q, q_0\}\), then reduce to a \((p/2, q)\)-instance
   - If \(q > \max\{p, q_0\}\), then reduce to a \((p, q/2)\)-instance
   - Good \(f\) for new instance \(\Rightarrow\) good \(f\) for original instance
3. Solve a \((p, q)\)-canonical instance with \(p \leq q_0, q \leq q_0\)
   - Can easily obtain \((2 - 1/q_0)\)-makespan
   - Not enough: lost a factor of \(\tilde{\Theta}(1/\sqrt{q_0})\) in reductions
   - Our goal: \((2 - \Omega(1/poly \log q_0))\)-makespan

- Use Lovasz Local Lemma many times
Overview of Algorithm

1. Reduce to \((p, q)\)-canonical instance
2. Reduce \(p\) and \(q\)
   - \(q_0 = O(1)\): a large constant
   - If \(p > \max\{q, q_0\}\), then reduce to a \((p/2, q)\)-instance
   - If \(q > \max\{p, q_0\}\), then reduce to a \((p, q/2)\)-instance
   - Good \(f\) for new instance \(\Rightarrow\) good \(f\) for original instance
3. Solve a \((p, q)\)-canonical instance with \(p \leq q_0, q \leq q_0\)
   - Can easily obtain \((2 - 1/q_0)\)-makespan
   - Not enough: lost a factor of \(\tilde{\Theta}(1/\sqrt{q_0})\) in reductions
   - Our goal: \((2 - \Omega(1/poly \log q_0))\)-makespan

- Use Lovasz Local Lemma many times
Solving \((p, q)\)-canonical instance with \(p, q \leq q_0\)

- A edge in \(G\) is **dense** if it has many parallel edges, and **sparse** otherwise.
Solving \((p, q)\)-canonical instance with \(p, q \leq q_0\)

- A edge in \(G\) is **dense** if it has many parallel edges, and **sparse** otherwise.

Case 1: all edges are sparse:

Randomly assign big jobs to machines

Apply uniform LLL

Case 2: all edges are dense:

Pre-processing, randomly assign big jobs to machines

Apply LLL, related to Galton-Watson process

General case: mixture of two cases
A edge in $G$ is **dense** if it has many parallel edges, and **sparse** otherwise.

Case 1: all edges are sparse:

- Randomly assign big jobs to machines
A edge in $G$ is **dense** if it has many parallel edges, and **sparse** otherwise.

Case 1: all edges are sparse:
- Randomly assign big jobs to machines
- Apply uniform LLL

Case 2: all edges are dense:
- Pre-processing, randomly assign big jobs to machines
- Apply LLL, related to Galton-Watson process

General case: mixture of two cases
Solving \((p, q)\)-canonical instance with \(p, q \leq q_0\)

- A edge in \(G\) is **dense** if it has many parallel edges, and **sparse** otherwise.

  **Case 1:** all edges are sparse:
  - Randomly assign big jobs to machines
  - Apply uniform LLL

  **Case 2:** all edges are dense:
  - Pre-processing, randomly assign big jobs to machines
  - Apply LLL, related to Galton-Watson process

Shi Li, TTIC
Solving \((p, q)\)-canonical instance with \(p, q \leq q_0\)

- A edge in \(G\) is **dense** if it has many parallel edges, and **sparse** otherwise.

Case 1: all edges are sparse:
- Randomly assign big jobs to machines
- Apply uniform LLL

Case 2: all edges are dense:
- Pre-processing, randomly assign big jobs to machines
Solving \((p, q)\)-canonical instance with \(p, q \leq q_0\)

- A edge in \(G\) is **dense** if it has many parallel edges, and **sparse** otherwise.

**Case 1**: all edges are sparse:
- Randomly assign big jobs to machines
- Apply uniform LLL

**Case 2**: all edges are dense:
- Pre-processing, randomly assign big jobs to machines
- Apply LLL, related to Galton-Watson process
A edge in $G$ is dense if it has many parallel edges, and sparse otherwise.

Case 1: all edges are sparse:
- Randomly assign big jobs to machines
- Apply uniform LLL

Case 2: all edges are dense:
- Pre-processing, randomly assign big jobs to machines
- Apply LLL, related to Galton-Watson process

General case: mixture of two cases
Summary
Summary

- $(2 - \delta^*)$-approximation for $(1, \epsilon)$-restricted assignment makespan minimization problem
(2 − δ*)-approximation for (1, ε)-restricted assignment makespan minimization problem

Reduce to (p, q)-canonical instances
Summary

- $(2 - \delta^*)$-approximation for $(1, \epsilon)$-restricted assignment makespan minimization problem
  1. Reduce to $(p, q)$-canonical instances
  2. Repeatedly reduce parameters $p$ and $q$
Summary

- $(2 - \delta^*)$-approximation for $(1, \epsilon)$-restricted assignment makespan minimization problem
  1. Reduce to $(p, q)$-canonical instances
  2. Repeatedly reduce parameters $p$ and $q$
  3. Solve $(p, q)$-canonical instances with $p, q \leq q_0$
Summary

- \((2 - \delta^*)\)-approximation for \((1, \epsilon)\)-restricted assignment makespan minimization problem
  - Reduce to \((p, q)\)-canonical instances
  - Repeatedly reduce parameters \(p\) and \(q\)
  - Solve \((p, q)\)-canonical instances with \(p, q \leq q_0\)

Open Problems
Summary

- $(2 - \delta^*)$-approximation for $(1, \epsilon)$-restricted assignment makespan minimization problem
  1. Reduce to $(p, q)$-canonical instances
  2. Repeatedly reduce parameters $p$ and $q$
  3. Solve $(p, q)$-canonical instances with $p, q \leq q_0$

Open Problems

- Simpler algorithm?
Summary

- $(2 - \delta^*)$-approximation for $(1, \epsilon)$-restricted assignment makespan minimization problem
  1. Reduce to $(p, q)$-canonical instances
  2. Repeatedly reduce parameters $p$ and $q$
  3. Solve $(p, q)$-canonical instances with $p, q \leq q_0$

Open Problems

- Simpler algorithm?
- Better-than-2-approximation for restricted assignment makespan minimization?
Summary

- $(2 - \delta^*)$-approximation for $(1, \epsilon)$-restricted assignment makespan minimization problem
  1. Reduce to $(p, q)$-canonical instances
  2. Repeatedly reduce parameters $p$ and $q$
  3. Solve $(p, q)$-canonical instances with $p, q \leq q_0$

Open Problems

- Simpler algorithm?
- Better-than-2-approximation for restricted assignment makespan minimization?

Thank you!