Grouping Techniques For Scheduling Problems

Tim Hartnack

Theory of Parallelism
Institute of Computer Science
Christian-Albrechts-University of Kiel

October 11, 2007
Overview

1. Introduction
   - Overview

2. Unrelated parallel machines with costs
   - Basic ideas
   - Rounding and profiling jobs
   - Grouping jobs
   - Dynamic programming

3. Outlook and discussion
Problem

- $0 < \varepsilon < 1$ fixed
- $m \geq 2$ fixed
- Given:
  - $n$ independent jobs
  - $m$ unrelated parallel machines
  - jobs without interruption
  - each machine: one job at a moment
  - job $J_j$ on machine $i$ requires $p_{ij} \geq 0$
  - and incurs $c_{ij} \geq 0$ costs, $i = 1, \ldots, m, j = 1, \ldots, n$
Problem

- $0 < \varepsilon < 1$ fixed
- $m \geq 2$ fixed
- Given:
  - $n$ independent jobs
  - $m$ unrelated parallel machines
  - jobs without interruption
  - each machine: one job at a moment
  - job $J_j$ on machine $i$ requires $p_{ij} \geq 0$
  - and incurs $c_{ij} \geq 0$ costs, $i = 1, \ldots, m$, $j = 1, \ldots, n$
Problem

- $0 < \varepsilon < 1$ fixed
- $m \geq 2$ fixed
- Given:
  - $n$ independent jobs
  - $m$ unrelated parallel machines
- jobs without interruption
- each machine: one job at a moment
- job $J_j$ on machine $i$ requires $p_{ij} \geq 0$
- and incurs $c_{ij} \geq 0$ costs, $i = 1, \ldots, m$, $j = 1, \ldots, n$
Problem

- $0 < \varepsilon < 1$ fixed
- $m \geq 2$ fixed

Given:
- $n$ independent jobs
- $m$ unrelated parallel machines
- jobs without interruption
- each machine: one job at a moment
- job $J_j$ on machine $i$ requires $p_{ij} \geq 0$
- and incurs $c_{ij} \geq 0$ costs, $i = 1, \ldots, m, j = 1, \ldots, n$
Problem

- $0 < \varepsilon < 1$ fixed
- $m \geq 2$ fixed
- Given:
  - $n$ independent jobs
  - $m$ unrelated parallel machines
- jobs without interruption
- each machine: one job at a moment
- job $J_j$ on machine $i$ requires $p_{ij} \geq 0$
- and incurs $c_{ij} \geq 0$ costs, $i = 1, \ldots, m, j = 1, \ldots, n$
Problem

- $0 < \varepsilon < 1$ fixed
- $m \geq 2$ fixed
- Given:
  - $n$ independent jobs
  - $m$ unrelated parallel machines
- jobs without interruption
- each machine: one job at a moment
- job $J_j$ on machine $i$ requires $p_{ij} \geq 0$
- and incurs $c_{ij} \geq 0$ costs, $i = 1, \ldots, m, j = 1, \ldots, n$
Problem

- $0 < \varepsilon < 1$ fixed
- $m \geq 2$ fixed
- Given:
  - $n$ independent jobs
  - $m$ unrelated parallel machines
- jobs without interruption
- each machine: one job at a moment
- job $J_j$ on machine $i$ requires $p_{ij} \geq 0$
- and incurs $c_{ij} \geq 0$ costs, $i = 1, \ldots, m, j = 1, \ldots, n$
Unrelated parallel machines with costs

Problem

- $0 < \varepsilon < 1$ fixed
- $m \geq 2$ fixed
- Given:
  - $n$ independent jobs
  - $m$ unrelated parallel machines
- Jobs without interruption
- Each machine: one job at a moment
- Job $J_j$ on machine $i$ requires $p_{ij} \geq 0$
- And incurs $c_{ij} \geq 0$ costs, $i = 1, \cdots, m, j = 1, \cdots, n$
Problem

- $0 < \varepsilon < 1$ fixed
- $m \geq 2$ fixed
- Given:
  - $n$ independent jobs
  - $m$ unrelated parallel machines
- Jobs without interruption
- Each machine: one job at a moment
- Job $J_j$ on machine $i$ requires $p_{ij} \geq 0$
- And incurs $c_{ij} \geq 0$ costs, $i = 1, \ldots, m, j = 1, \ldots, n$
Objective function of unrelated parallel machines with costs

- Objective function

\[ T + \mu \sum_{j=1}^{n} \sum_{i=1}^{n} x_{ij} c_{ij} \]  \hspace{1cm} (1)

- with \( x_{ij} = \begin{cases} 1, & \text{if job } J_j \text{ runs on machine } i \\ 0, & \text{else} \end{cases} \)

- \( T \) makespan, and \( \mu \geq 0 \)

- By multiplying each cost value by \( \mu \) we may assume, w.l.o.g. that \( \mu = 1 \)
Objective function of unrelated parallel machines with costs

- Objective function

\[ T + \mu \sum_{j=1}^{n} \sum_{i=1}^{n} x_{ij} c_{ij} \]  

(1)

- with \( x_{ij} = \begin{cases} 
1, & \text{if job } J_j \text{ runs on machine } i \\
0, & \text{else} 
\end{cases} \)

- \( T \) makespan, and \( \mu \geq 0 \)

- By multiplying each cost value by \( \mu \) we may assume, w.l.o.g. that \( \mu = 1 \)
Objective function of unrelated parallel machines with costs

- Objective function
  \[ T + \mu \sum_{j=1}^{n} \sum_{i=1}^{n} x_{ij} c_{ij} \]  \hspace{1cm} (1)

- with \( x_{ij} = \begin{cases} 
1, & \text{if job } J_j \text{ runs on machine } i \\
0, & \text{else} 
\end{cases} \)

- \( T \) makespan, and \( \mu \geq 0 \)

- By multiplying each cost value by \( \mu \) we may assume, w.l.o.g. that \( \mu = 1 \)
Objective function of unrelated parallel machines with costs

- Objective function

\[ T + \mu \sum_{j=1}^{n} \sum_{i=1}^{n} x_{ij} c_{ij} \]  \hspace{1cm} (1)

- with \( x_{ij} = \begin{cases} 
1, & \text{if job } J_j \text{ runs on machine } i \\
0, & \text{else} 
\end{cases} \)

- \( T \) makespan, and \( \mu \geq 0 \)

- By multiplying each cost value by \( \mu \) we may assume, w.l.o.g. that \( \mu = 1 \)
Notation and scaling factors

Definition (scaling factor)

Define for each job $J_j \in \mathcal{J}$

1. $d_j = \min_{i=1, \ldots, m} (p_{ij} + c_{ij})$
2. $D = \sum_{j=1}^{n} d_j$
Upper and lower bound of the objective function

Lemma
For the objective function, the following inequality holds: \( D \leq \text{OPT} \leq m \)

Proof.

\[
D = \sum_{j=1}^{n} d_j \leq \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^* c_{ij} + \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij}^* p_{ij} \\
\leq C^* + T^* \leq m (C^* + T^*) = m \cdot \text{OPT}
\]
Upper and lower bound of the objective function

- Let $m_j$ indicate a machine such that $d_j = p_{m_j,j} + c_{m_j,j}$
- Assign each job $J_j$ to machine $m_j$
- The objective function is bounded by
  \[
  \sum_{j \in J} c_{m_j,j} + \sum_{j \in J} p_{m_j,j} = D
  \]
- $OPT \in \left[\frac{D}{m}, D\right]$
- By dividing all times and costs by $\frac{D}{m}$ we get:
  \[
  1 \leq OPT \leq m
  \]
Let $m_j$ indicate a machine such that $d_j = p_{m_j,j} + c_{m_j,j}$

Assign each job $J_j$ to machine $m_j$

The objective function is bounded by

$$\sum_{j \in J} c_{m_j,j} + \sum_{j \in J} p_{m_j,j} = D$$

$OPT \in \left[ \frac{D}{m}, D \right]$  

By dividing all times and costs by $\frac{D}{m}$ we get:

$$1 \leq OPT \leq m$$
Upper and lower bound of the objective function

- Let $m_j$ indicate a machine such that $d_j = p_{m,j} + c_{m,j}$
- Assign each job $J_j$ to machine $m_j$
- The objective function is bounded by
  \[
  \sum_{j \in J} c_{m,j} + \sum_{j \in J} p_{m,j} = D
  \]
- $OPT \in \left[ \frac{D}{m}, D \right]$
- By dividing all times and costs by $\frac{D}{m}$ we get:
  \[1 \leq OPT \leq m\]
Unrelated parallel machines with costs

Upper and lower bound of the objective function

- Let $m_j$ indicate a machine such that $d_j = p_{m_j,j} + c_{m_j,j}$
- Assign each job $J_j$ to machine $m_j$
- The objective function is bounded by

$$\sum_{j \in J} c_{m_j,j} + \sum_{j \in J} p_{m_j,j} = D$$

- $OPT \in \left[\frac{D}{m}, D\right]$
- By dividing all times and costs by $\frac{D}{m}$ we get:

$$1 \leq OPT \leq m$$
Upper and lower bound of the objective function

- Let $m_j$ indicate a machine such that $d_j = p_{m_j,j} + c_{m_j,j}$
- Assign each job $J_j$ to machine $m_j$
- The objective function is bounded by

$$\sum_{j \in J} c_{m_j,j} + \sum_{j \in J} p_{m_j,j} = D$$

- $OPT \in \left[ \frac{D}{m}, D \right]$
- By dividing all times and costs by $\frac{D}{m}$ we get:

$$1 \leq OPT \leq m$$
Overview of the algorithm

1. **Rounding** and **profiling** of jobs creates *profiles*
   - constant number of profiles

2. **Grouping** of jobs
   - constant number of jobs

3. Schedule constant number of jobs with dynamic programming
Overview of the algorithm

1. **Rounding** and **profiling** of jobs creates *profiles*
   - constant number of profiles

2. **Grouping** of jobs
   - constant number of jobs

3. Schedule constant number of jobs with **dynamic programming**
Overview of the algorithm

1. **Rounding** and **profiling** of jobs creates *profiles*
   - constant number of profiles

2. **Grouping** of jobs
   - constant number of jobs

3. Schedule constant number of jobs with **dynamic programming**

Observation (Transformation)

We say that a transformation produces $1 + O(\varepsilon)$ loss at the objective function.
Overview of the algorithm

1. **Rounding** and **profiling** of jobs creates *profiles*
   - constant number of profiles

2. **Grouping** of jobs
   - constant number of jobs

3. Schedule constant number of jobs with **dynamic programming**

Observation (Transformation)

We say that a transformation produces $1 + O(\varepsilon)$ loss at the objective function.
Overview of the algorithm

1. **Rounding** and **profiling** of jobs creates *profiles*
   - constant number of profiles

2. **Grouping** of jobs
   - constant number of jobs

3. Schedule constant number of jobs with **dynamic programming**

Observation (Transformation)

*We say that a transformation produces* $1 + O(\varepsilon)$ *loss at the objective function*
Overview of the algorithm

1. **Rounding** and **profiling** of jobs creates *profiles*
   - constant number of profiles

2. **Grouping** of jobs
   - constant number of jobs

3. Schedule constant number of jobs with **dynamic programming**

**Observation (Transformation)**

*We say that a transformation produces* $1 + O(\varepsilon)$ *loss at the objective function*
Sets of machines

For every $J_j$ define:

**fast machines** $p_{ij} \leq \frac{\varepsilon}{m} d_j$

**cheap machines** $c_{ij} \leq \frac{\varepsilon}{m} d_j$

**slow machines** $p_{ij} \geq \frac{m}{\varepsilon} d_j$

**expensive machines** $c_{ij} \geq \frac{d_j}{\varepsilon}$
Sets of machines

For every $J_j$ define:

- **fast machines** $p_{ij} \leq \frac{\varepsilon}{m} d_j$
- **cheap machines** $c_{ij} \leq \frac{\varepsilon}{m} d_j$
- **slow machines** $p_{ij} \geq \frac{m}{\varepsilon} d_j$
- **expensive machines** $c_{ij} \geq \frac{d_i}{\varepsilon}$
For every $J_j$ define:

- **Fast machines** $p_{ij} \leq \frac{\epsilon}{m} d_j$
- **Cheap machines** $c_{ij} \leq \frac{\epsilon}{m} d_j$
- **Slow machines** $p_{ij} \geq \frac{m}{\epsilon} d_j$
- **Expensive machines** $c_{ij} \geq \frac{d_j}{\epsilon}$
Sets of machines

For every $J_j$ define:

- **fast machines** $p_{ij} \leq \frac{\varepsilon}{m} d_j$
- **cheap machines** $c_{ij} \leq \frac{\varepsilon}{m} d_j$
- **slow machines** $p_{ij} \geq \frac{m}{\varepsilon} d_j$
- **expensive machines** $c_{ij} \geq \frac{d_j}{\varepsilon}$
Rounding Jobs

fast machine $i$ of $J_j : p_{ij} = 0$

cheap machine $i$ of $J_j : c_{ij} = 0$

slow machine $i$ of $J_j : p_{ij} = +\infty$

expensive machine $i \in$ of $J_j : c_{ij} = +\infty$

other machine $i$ of $J_j$ round $p_{ij}, c_{ij}$ to the nearest lower value of $\frac{\varepsilon}{m} d_j (1 + \epsilon)^h$, for some $h \in \mathbb{N}$
Rounding Jobs

fast machine \( i \) of \( J_j \) : \( p_{ij} = 0 \)

cheap machine \( i \) of \( J_j \) : \( c_{ij} = 0 \)

slow machine \( i \) of \( J_j \) : \( p_{ij} = +\infty \)

expensive machine \( i \in \) of \( J_j \) : \( c_{ij} = +\infty \)

other machine \( i \) of \( J_j \) round \( p_{ij}, c_{ij} \) to the nearest lower value of \( \frac{\varepsilon}{m} d_j (1 + \varepsilon)^h \), for some \( h \in \mathbb{N} \)
Rounding Jobs

- **fast machine** $i$ of $J_j$: $p_{ij} = 0$
- **cheap machine** $i$ of $J_j$: $c_{ij} = 0$
- **slow machine** $i$ of $J_j$: $p_{ij} = +\infty$
- **expensive machine** $i \in J_j$: $c_{ij} = +\infty$
- **other machine** $i$ of $J_j$ round $p_{ij}$, $c_{ij}$ to the nearest lower value of $\frac{\varepsilon}{m} d_j (1 + \varepsilon)^h$, for some $h \in \mathbb{N}$

**Observation**

For each job $J_j \in \mathcal{J}$, there is always a machine with is neither expensive nor slow.
Rounding Jobs

fast machine \( i \) of \( J_j : p_{ij} = 0 \)

cheap machine \( i \) of \( J_j : c_{ij} = 0 \)

slow machine \( i \) of \( J_j : p_{ij} = +\infty \)

expensive machine \( i \in J_j : c_{ij} = +\infty \)

other machine \( i \) of \( J_j \) round \( p_{ij}, c_{ij} \) to the nearest lower value of \( \frac{\varepsilon}{m} d_j (1 + \varepsilon)^h \), for some \( h \in \mathbb{N} \)

Observation

For each job \( J_j \in J \) there is always a machine which is neither expensive nor slow.
Rounding Jobs

- Fast machine $i$ of $J_j$: $p_{ij} = 0$
- Cheap machine $i$ of $J_j$: $c_{ij} = 0$
- Slow machine $i$ of $J_j$: $p_{ij} = +\infty$
- Expensive machine $i \in J_j$: $c_{ij} = +\infty$
- Other machine $i$ of $J_j$ round $p_{ij}, c_{ij}$ to the nearest lower value of $\varepsilon d_j (1 + \varepsilon)^h$, for some $h \in \mathbb{N}$

Observation

For each job $J_j \in J$ there is always a machine which is neither expensive nor slow
Rounding Jobs

fast machine \( i \) of \( J_j \): \( p_{ij} = 0 \)
cheap machine \( i \) of \( J_j \): \( c_{ij} = 0 \)
slow machine \( i \) of \( J_j \): \( p_{ij} = +\infty \)
expensive machine \( i \in \) of \( J_j \): \( c_{ij} = +\infty \)
other machine \( i \) of \( J_j \) round \( p_{ij}, c_{ij} \) to the nearest lower value of \( \frac{\varepsilon}{m} d_j (1 + \varepsilon)^h \), for some \( h \in \mathbb{N} \)

Observation

For each job \( J_j \in \mathcal{J} \) there is always a machine which is neither expensive nor slow
Results of rounding

Lemma

*Rounding produces* $1 + 4\varepsilon$ *loss*

Proof.

- Start by considering rounding to zero the times and costs of jobs on fast and cheap machines, respectively
  - Let $A$ be an optimal schedule of this
  - The objective function value of $A \leq \text{OPT}$

- $F$ and $C$ denote sets of jobs, which are processed on fast and cheap machines according to $A$
- Replace times and costs of the transformed instance by the originals

$$\sum_{j \in F} t_j + \sum_{j \in C} t_j \leq 2 \sum_{j = 1}^{2\varepsilon m} t_j = 2\varepsilon P_m = 2\varepsilon$$
Results of rounding

Lemma

*Rounding produces* $1 + 4\varepsilon$ *loss*

Proof.

- Start by considering rounding to zero the times and costs of jobs on fast and cheap machines, respectively
  - Let $A$ be an optimal schedule of this
  - The objective function value of $A \leq \text{OPT}$
    - we just reduced times and costs
  - $F$ and $C$ denote sets of jobs, which are processed on fast and cheap machines according to $A$
  - Replace times and costs of the transformed instance by the originals

$$
\sum_{j \in F} \frac{\varepsilon}{m} d_j + \sum_{j \in C} \frac{\varepsilon}{m} d_j \leq 2 \sum_{j=1}^{n} \frac{\varepsilon}{m} d_j = 2 \varepsilon \frac{D}{m} = 2\varepsilon
$$
Results of rounding

Lemma

*Rounding produces* $1 + 4\varepsilon$ *loss*

Proof.

- Start by considering rounding to zero the times and costs of jobs on fast and cheap machines, respectively
  - Let $A$ be an optimal schedule of this
  - The objective function value of $A \leq \text{OPT}$
    - we just reduced times and costs
  - $F$ and $C$ denote sets of jobs, which are processed on fast and cheap machines according to $A$
  - Replace times and costs of the transformed instance by the originals
    \[
    \sum_{j \in F} \frac{\varepsilon}{m} d_j + \sum_{j \in C} \frac{\varepsilon}{m} d_j \leq 2 \sum_{j=1}^{n} \frac{\varepsilon}{m} d_j = 2 \varepsilon \frac{D}{m} = 2\varepsilon
    \]
**Results of rounding**

**Lemma**

*Rounding produces* $1 + 4\varepsilon$ *loss*

**Proof.**

- Start by considering rounding to zero the times and costs of jobs on fast and cheap machines, respectively
  - Let $A$ be an optimal schedule of this
  - The objective function value of $A \leq \text{OPT}$
    - we just reduced times and costs
  - $F$ and $C$ denote sets of jobs, which are processed on fast and cheap machines according to $A$
  - Replace times and costs of the transformed instance by the originals

$$
\sum_{J_j \in F} \frac{\varepsilon}{m} d_j + \sum_{J_j \in C} \frac{\varepsilon}{m} d_j \leq 2 \sum_{j=1}^{n} \frac{\varepsilon}{m} d_j = 2\varepsilon \frac{D}{m} = 2\varepsilon
$$
Results of rounding

Lemma

*Rounding produces 1 + 4\(\epsilon\) loss*

Proof.

- Start by considering rounding to zero the times and costs of jobs on fast and cheap machines, respectively
  - Let \(A\) be an optimal schedule of this
  - The objective function value of \(A \leq OPT\)
    - we just reduced times and costs
  - \(F\) and \(C\) denote sets of jobs, which are processed on fast and cheap machines according to \(A\)
  - Replace times and costs of the transformed instance by the originals

\[
\sum_{J_j \in F} \frac{\epsilon}{m} d_j + \sum_{J_j \in C} \frac{\epsilon}{m} d_j \leq 2 \sum_{j=1}^{n} \frac{\epsilon}{m} d_j = 2\epsilon \frac{D}{m} = 2\epsilon
\]
Results of rounding

Lemma

*Rounding produces $1 + 4\varepsilon$ loss*

Proof.

- Start by considering rounding to zero the times and costs of jobs on fast and cheap machines, respectively
  - Let $A$ be an optimal schedule of this
  - The objective function value of $A \leq OPT$
    - we just reduced times and costs
  - $F$ and $C$ denote sets of jobs, which are processed on fast and cheap machines according to $A$
  - Replace times and costs of the transformed instance by the originals

\[
\sum_{J_j \in F} \frac{\varepsilon}{m} d_j + \sum_{J_j \in C} \frac{\varepsilon}{m} d_j \leq 2 \sum_{j=1}^{n} \frac{\varepsilon}{m} d_j = 2\varepsilon \frac{D}{m} = 2\varepsilon
\]
Results of rounding II

Proof.

- Show: there exists an approximate schedule where jobs are scheduled neither on slow nor on expensive machines
  
  \( p_{ij}, c_{ij} := +\infty \)

- Let \( A \) be an optimal schedule, \( T^* \) Makespan \( C^* \) total costs

- \( S \) and \( E \) sets,

- Assign \( J_j \in S \cup E \) \( m_j \)

- This may increase the objective function value by at most

\[
\sum_{J_j \in S \cup E} d_j \leq \frac{\epsilon}{m} \sum_{J_j \in S} p_{A(j),j} + \epsilon \sum_{J_j \in E} c_{A(j),j} \leq \epsilon T^* + \epsilon C^*
\]

since \( p_{A(j),j} \geq \frac{m}{\epsilon} d_j \) for \( J_j \in S \) and \( c_{A(j),j} \geq \frac{d_j}{\epsilon} \) for \( J_j \in E \)
Results of rounding II

Proof.

- Show: there exists an approximate schedule where jobs are scheduled neither on slow nor on expensive machines
  - \( p_{ij}, c_{ij} := +\infty \)
- Let \( A \) be an optimal schedule, \( T^* \) Makespan \( C^* \) total costs
- \( S \) and \( E \) sets,
  - containing jobs, running on slow and expensive machines
- Assign \( J_j \in S \cup E \ m_j \)
- This may increase the objective function value by at most

\[
\sum_{J_j \in S \cup E} d_j \leq \frac{\varepsilon}{m} \sum_{J_j \in S} p_{A(j),j} + \varepsilon \sum_{J_j \in E} c_{A(j),j} \leq \varepsilon T^* + \varepsilon C^*
\]

since \( p_{A(j),j} \geq \frac{m}{\varepsilon} d_j \) for \( J_j \in S \) and \( c_{A(j),j} \geq \frac{d_j}{\varepsilon} \) for \( J_j \in E \)
Results of rounding II

Proof.

- Show: there exists an approximate schedule where jobs are scheduled neither on slow nor on expensive machines
  - \( p_{ij}, c_{ij} := +\infty \)
- Let \( A \) be an optimal schedule, \( T^* \) Makespan \( C^* \) total costs
- \( S \) and \( E \) sets, containing jobs, running on slow and expensive machines
- Assign \( J_j \in S \cup E \ m_j \)
- This may increase the objective function value by at most

\[
\sum_{J_j \in S \cup E} d_j \leq \frac{\varepsilon}{m} \sum_{J_j \in S} p_{A(j), j} + \varepsilon \sum_{J_j \in E} c_{A(j), j} \leq \varepsilon T^* + \varepsilon C^*
\]

since \( p_{A(j), j} \geq \frac{m}{e} d_j \) for \( J_j \in S \) and \( c_{A(j), j} \geq \frac{d_j}{e} \) for \( J_j \in E \)
Results of rounding II

Proof.

- Show: there exists an approximate schedule where jobs are scheduled neither on slow nor on expensive machines
  - \( p_{ij}, c_{ij} := +\infty \)
- Let \( A \) be an optimal schedule, \( T^* \) Makespan \( C^* \) total costs
- \( S \) and \( E \) sets,
  - containing jobs, running on slow and expensive machines
- Assign \( J_j \in S \cup E \) \( m_j \)
- This may increase the objective function value by at most

\[
\sum_{J_j \in S \cup E} d_j \leq \frac{\varepsilon}{m} \sum_{J_j \in S} p_{A(j), j} + \varepsilon \sum_{J_j \in E} c_{A(j), j} \leq \varepsilon T^* + \varepsilon C^*
\]

since \( p_{A(j), j} \geq \frac{m}{\varepsilon} d_j \) for \( J_j \in S \) and \( c_{A(j), j} \geq \frac{d_j}{\varepsilon} \) for \( J_j \in E \)
Results of rounding II

Proof.

- Show: there exists an approximate schedule where jobs are scheduled neither on slow nor on expensive machines
  - \( p_{ij}, c_{ij} := +\infty \)
- Let \( A \) be an optimal schedule, \( T^* \) Makespan \( C^* \) total costs
- \( S \) and \( E \) sets,
  - containing jobs, running on slow and expensive machines
- Assign \( J_j \in S \cup E \ m_j \)
- This may increase the objective function value by at most

\[
\sum_{J_j \in S \cup E} d_j \leq \frac{\varepsilon}{m} \sum_{J_j \in S} p_{A(j),j} + \varepsilon \sum_{J_j \in E} c_{A(j),j} \leq \varepsilon T^* + \varepsilon C^*
\]

since \( p_{A(j),j} \geq \frac{m}{\varepsilon} d_j \) for \( J_j \in S \) and \( c_{A(j),j} \geq \frac{d_j}{\varepsilon} \) for \( J_j \in E \)
Results of rounding II

Proof.

- Show: there exists an approximate schedule where jobs are scheduled neither on slow nor on expensive machines
  
  \[ p_{ij}, c_{ij} := +\infty \]

- Let \( A \) be an optimal schedule, \( T^* \) Makespan \( C^* \) total costs

- \( S \) and \( E \) sets,
  
  containing jobs, running on slow and expensive machines

- Assign \( J_j \in S \cup E \ m_j \)

- This may increase the objective function value by at most

\[
\sum_{J_j \in S \cup E} d_j \leq \frac{\varepsilon}{m} \sum_{J_j \in S} p_{A(j),j} + \varepsilon \sum_{J_j \in E} c_{A(j),j} \leq \varepsilon T^* + \varepsilon C^*
\]

since \( p_{A(j),j} \geq \frac{m}{\varepsilon} d_j \) for \( J_j \in S \) and \( c_{A(j),j} \geq \frac{d_j}{\varepsilon} \) for \( J_j \in E \)
Results of rounding II

Proof.

- Show: there exists an approximate schedule where jobs are scheduled neither on slow nor on expensive machines
  
  \[ p_{ij}, c_{ij} := +\infty \]

- Let \( A \) be an optimal schedule, \( T^* \) Makespan \( C^* \) total costs
- \( S \) and \( E \) sets,
  
  containing jobs, running on slow and expensive machines
- Assign \( J_j \in S \cup E \) \( m_j \)
- This may increase the objective function value by at most

\[
\sum_{J_j \in S \cup E} d_j \leq \frac{\varepsilon}{m} \sum_{J_j \in S} p_{A(j),j} + \varepsilon \sum_{J_j \in E} c_{A(j),j} \leq \varepsilon T^* + \varepsilon C^*
\]

since \( p_{A(j),j} \geq \frac{m}{\varepsilon} d_j \) for \( J_j \in S \) and \( c_{A(j),j} \geq \frac{d_j}{\varepsilon} \) for \( J_j \in E \)
Summary & Outlook

- up to now
  - All jobs rounded
- next
  - Create profiles of jobs
Summary & Outlook

- up to now
  - All jobs rounded
- next
  - Create profiles of jobs
Summary & Outlook

- **up to now**
  - All jobs rounded
- **next**
  - Create profiles of jobs
Summary & Outlook

- up to now
  - All jobs rounded
- next
  - Create profiles of jobs
Profiles for jobs

Definition (Execution profile)
The execution profile of a job \( J_j \) is a \( m \)-tuple

\[
\langle \Pi_{1,j}, \ldots, \Pi_{m,j} \rangle,
\]

so that \( p_{ij} = \frac{\varepsilon}{m} d_j (1 + \varepsilon)^{\Pi_{i,j}} \)

Definition (Cost profile)
The cost profile of a job \( J_j \) is a \( m \)-tuple

\[
\langle \Gamma_{1,j}, \ldots, \Gamma_{m,j} \rangle,
\]

so that \( c_{ij} = \frac{\varepsilon}{m} d_j (1 + \varepsilon)^{\Gamma_{i,j}} \)
Profiles for jobs

Definition (Execution profile)
The execution profile of a job $J_j$ is a $m$-tuple

$$\langle \Pi_{1,j}, \ldots, \Pi_{m,j} \rangle,$$

so that $p_{ij} = \frac{\epsilon m d_j}{m} (1 + \epsilon)^{\Pi_{i,j}}$

Definition (Cost profile)
The cost profile of a job $J_j$ is a $m$-tuple

$$\langle \Gamma_{1,j}, \ldots, \Gamma_{m,j} \rangle,$$

so that $c_{ij} = \frac{\epsilon m d_j}{m} (1 + \epsilon)^{\Gamma_{i,j}}$
Special cases in the profile

- For $p_{ij} = +\infty$ put $\Pi_{i,j} := +\infty$
- For $p_{ij} = 0$ put $\Pi_{i,j} := -\infty$
- For $c_{ij} = +\infty$ put $\Gamma_{i,j} := +\infty$
- For $c_{ij} = 0$ put $\Gamma_{i,j} := -\infty$
Special cases in the profile

- For $p_{ij} = +\infty$ put $\Pi_{i,j} := +\infty$
- For $p_{ij} = 0$ put $\Pi_{i,j} := -\infty$
- For $c_{ij} = +\infty$ put $\Gamma_{i,j} := +\infty$
- For $c_{ij} = 0$ put $\Gamma_{i,j} := -\infty$

Observation
Two jobs have the same profile, if they have the same execution profile as well as the same cost profile.
Special cases in the profile

- For $p_{ij} = +\infty$ put $\Pi_{i,j} := +\infty$
- For $p_{ij} = 0$ put $\Pi_{i,j} := -\infty$
- For $c_{ij} = +\infty$ put $\Gamma_{i,j} := +\infty$
- For $c_{ij} = 0$ put $\Gamma_{i,j} := -\infty$

Observation

Two jobs have the same profile, if they have the same execution profile as well as the same cost profile.
Special cases in the profile

- For $p_{ij} = +\infty$ put $\Pi_{i,j} := +\infty$
- For $p_{ij} = 0$ put $\Pi_{i,j} := -\infty$
- For $c_{ij} = +\infty$ put $\Gamma_{i,j} := +\infty$
- For $c_{ij} = 0$ put $\Gamma_{i,j} := -\infty$

Observation

Two jobs have the same profile, if they have the same execution profile as well as the same cost profile.
Special cases in the profile

- For $p_{ij} = +\infty$ put $\Pi_{i,j} := +\infty$
- For $p_{ij} = 0$ put $\Pi_{i,j} := -\infty$
- For $c_{ij} = +\infty$ put $\Gamma_{i,j} := +\infty$
- For $c_{ij} = 0$ put $\Gamma_{i,j} := -\infty$

Observation

*Two jobs have the same profile, if they have the same execution profile as well as the same cost profile*
Number of profiles

Lemma

The number of different profiles is at most

\[ l := \left( 3 + 2 \log_{1+\varepsilon} \frac{m}{\varepsilon} \right)^{2m} \]
Summary & Outlook

- up to now
  - All jobs rounded
  - Every job has a profile
  - Number of profiles is constant

next: Group jobs $\implies$ Number of jobs constant
Summary & Outlook

- up to now
  - All jobs rounded
  - Every job has a profile
  - Number of profiles is constant

next: Group jobs $\implies$ Number of jobs constant
Summary & Outlook

- **up to now**
  - All jobs rounded
  - Every job has a profile
  - Number of profiles is constant

  next: Group jobs ⇒ Number of jobs constant
Summary & Outlook

- up to now
  - All jobs rounded
  - Every job has a profile
  - Number of profiles is constant

- next: Group jobs $\implies$ Number of jobs constant
Summary & Outlook

- up to now
  - All jobs rounded
  - Every job has a profile
  - Number of profiles is constant

- next: Group jobs $\implies$ Number of jobs constant
Grouping Jobs

1. Make a partition of the jobs

\[ L = \{ J_j : d_j > \frac{\varepsilon}{m} \} \]

and

\[ S = \{ J_j : d_j \leq \frac{\varepsilon}{m} \} \]

2. \( L \) set of big jobs
3. \( S \) set of small jobs
4. Partition \( S \) in \( S_i, i = 1, \cdots, l \) based on the profile

5. Use the above grouping on all \( S_i \) of \( S \)
Grouping Jobs

1. Make a partition of the jobs

\[ L = \{ J_j : d_j > \frac{\varepsilon}{m} \} \]

and

\[ S = \{ J_j : d_j \leq \frac{\varepsilon}{m} \} \]

2. \( L \) set of big jobs

3. \( S \) set of small jobs

4. Partition \( S \) in \( S_i, i = 1, \ldots, l \) based on the profile

\[ \text{Create } J_c \text{ from } J_a \text{ and } J_b \]

Continue this step until there is only one job \( J_j \in S_i \) with \( d_j \leq \frac{\varepsilon}{m} \) left

5. use the above grouping on all \( S_i \) of \( S \)
Grouping Jobs

1. Make a partition of the jobs

\[ L = \{ J_j : d_j > \frac{\varepsilon}{m} \} \]

and

\[ S = \{ J_j : d_j \leq \frac{\varepsilon}{m} \} \]

2. \( L \) set of big jobs

3. \( S \) set of small jobs

4. Partition \( S \) in \( S_i, i = 1, \cdots, l \) based on the profile

   - \( J_a, J_b \in S_i \) with \( d_a, d_b \leq \frac{\varepsilon}{2} \)
   - Create \( J_c \) from \( J_a \) and \( J_b \)
   - Continue this step until there is only one job \( J_j \in S_i \) with \( d_j \leq \frac{\varepsilon}{2} \) left

5. Use the above grouping on all \( S_i \) of \( S \)
Grouping Jobs

1. Make a partition of the jobs

\[ L = \{ J_j : d_j > \frac{\varepsilon}{m} \} \]

and

\[ S = \{ J_j : d_j \leq \frac{\varepsilon}{m} \} \]

2. $L$ set of big jobs

3. $S$ set of small jobs

4. Partition $S$ in $S_i, i = 1, \ldots, l$ based on the profile
   - $J_a, J_b \in S_i$ with $d_a, d_b \leq \frac{\varepsilon}{m}$
   - Create $J_c$ from $J_a$ and $J_b$
   - Continue this step until there is only one job $J_j \in S_i$ with $d_j \leq \frac{\varepsilon}{m}$ left

5. Use the above grouping on all $S_i$ of $S$
Grouping Jobs

1. Make a partition of the jobs

\[ L = \{ J_j : d_j > \frac{\varepsilon}{m} \} \]

and

\[ S = \{ J_j : d_j \leq \frac{\varepsilon}{m} \} \]

2. \( L \) set of big jobs
3. \( S \) set of small jobs
4. Partition \( S \) in \( S_i, i = 1, \cdots, l \) based on the profile

- \( J_a, J_b \in S_i \) with \( d_a, d_b \leq \frac{\varepsilon}{m^2} \)
- Create \( J_c \) from \( J_a \) and \( J_b \)
- Continue this step until there is only one job \( J_j \in S_i \) with \( d_j \leq \frac{\varepsilon}{m^2} \) left

5. Use the above grouping on all \( S_i \) of \( S \)
Grouping Jobs

1. Make a partition of the jobs

\[ L = \{ J_j : d_j > \frac{\varepsilon}{m} \} \]

and

\[ S = \{ J_j : d_j \leq \frac{\varepsilon}{m} \} \]

2. \( L \) set of big jobs

3. \( S \) set of small jobs

4. Partition \( S \) in \( S_i, i = 1, \cdots, l \) based on the profile

- \( J_a, J_b \in S_i \) with \( d_a, d_b \leq \frac{\varepsilon}{m} \)
- Create \( J_c \) from \( J_a \) and \( J_b \)
- Continue this step until there is only one job \( J_j \in S_i \) with \( d_j \leq \frac{\varepsilon}{m} \) left

5. Use the above grouping on all \( S_i \) of \( S \)
Grouping Jobs

1. Make a partition of the jobs

\[ L = \{ J_j : d_j > \frac{\varepsilon}{m} \} \]

and

\[ S = \{ J_j : d_j \leq \frac{\varepsilon}{m} \} \]

2. \( L \) set of big jobs

3. \( S \) set of small jobs

4. Partition \( S \) in \( S_i, i = 1, \ldots, l \) based on the profile

   \( J_a, J_b \in S_i \) with \( d_a, d_b \leq \frac{m}{2} \)

   Create \( J_c \) from \( J_a \) and \( J_b \)

   Continue this step until there is only one job \( J_j \in S_i \) with \( d_j \leq \frac{m}{2} \) left

5. use the above grouping on all \( S_i \) of \( S \)
Grouping Jobs

1. Make a partition of the jobs

\[ L = \{ J_j : d_j > \frac{\varepsilon}{m} \} \]

and

\[ S = \{ J_j : d_j \leq \frac{\varepsilon}{m} \} \]

2. \( L \) set of big jobs

3. \( S \) set of small jobs

4. Partition \( S \) in \( S_i, i = 1, \ldots, l \) based on the profile

   - \( J_a, J_b \in S_i \) with \( d_a, d_b \leq \frac{m}{2} \)
   - Create \( J_c \) from \( J_a \) and \( J_b \)
   - Continue this step until there is only one job \( J_j \in S_i \) with \( d_j \leq \frac{m}{2} \) left

5. Use the above grouping on all \( S_i \) of \( S \)
Results of grouping

**Lemma**

*With a loss of $1 + \varepsilon$ the number of jobs can be reduced to*

$k := \min\{n, \left(\log\frac{m}{\varepsilon}\right)^{O(m)}\}$

**Proof.**

- After the grouping there are at most $l$ jobs, one from each subset $S_i$, with $d_j \leq \frac{\varepsilon m}{2}$
- Therefore the number of jobs is bounded to:

$$\frac{2D}{\varepsilon} + l \leq \frac{2m^2}{\varepsilon} + l = \left(\log\frac{m}{\varepsilon}\right)^{O(m)}$$

- Proof of loss will be omitted
Results of grouping

Lemma

With a loss of $1 + \varepsilon$ the number of jobs can be reduced to

$$k := \min\{n, \left(\log \frac{m}{\varepsilon}\right)^{O(m)}\}$$

Proof.

- After the grouping there are at most $l$ jobs, one from each subset $S_i$, with
  $$d_j \leq \frac{\varepsilon}{m}$$
- Therefore the number of jobs is bounded to:
  $$\frac{2D}{\varepsilon} + l \leq \frac{2m^2}{\varepsilon} + l = \left(\log \frac{m}{\varepsilon}\right)^{O(m)}$$

- Proof of loss will be omitted
Results of grouping

**Lemma**

*With a loss of $1 + \varepsilon$ the number of jobs can be reduced to*

$$k := \min\{n, \left(\log \frac{m}{\varepsilon}\right)^{O(m)}\}$$

**Proof.**

- After the grouping there are at most $l$ jobs, one from each subset $S_i$, with
  $$d_j \leq \frac{\varepsilon}{m}$$
- Therefore the number of jobs is bounded to:
  $$\frac{2D}{\varepsilon} + l \leq \frac{2m^2}{\varepsilon} + l = \left(\log \frac{m}{\varepsilon}\right)^{O(m)}$$
- Proof of loss will be omitted
Summary & Outlook

up to now
- All jobs rounded
- Every job has a profile
- Number of profiles constant
- Grouping \(\implies\) Number of jobs constant

next: Create a schedule with dynamic programming
Summary & Outlook

- up to now
  - All jobs rounded
    - Every job has a profile
    - Number of profiles constant
  - Grouping $\rightarrow$ Number of jobs constant

- next: Create a schedule with dynamic programming
Summary & Outlook

- up to now
  - All jobs rounded
  - Every job has a profile
  - Number of profiles constant
  - Grouping $\implies$ Number of jobs constant

- next: Create a schedule with dynamic programming
Summary & Outlook

- **up to now**
  - All jobs rounded
  - Every job has a profile
  - Number of profiles constant
  - Grouping $\implies$ Number of jobs constant

- **next:** Create a schedule with dynamic programming
Summary & Outlook

up to now
- All jobs rounded
- Every job has a profile
- Number of profiles constant
- Grouping $\implies$ Number of jobs constant

next: Create a schedule with dynamic programming
Summary & Outlook

- up to now
  - All jobs rounded
  - Every job has a profile
  - Number of profiles constant
  - Grouping $\implies$ Number of jobs constant

- next: Create a schedule with dynamic programming
Dynamic Programming

1. $J_1, \cdots, J_k$ jobs of the transformed instance
2. A schedule configuration $s = (t_1, \cdots, t_m, c)$ is a $(m+1)$-tuple
   - $t_i$ completion time of machine $i$
   - $c$ total cost
3. $V_j$ a set of these tuples (f.a. $j = 1, \cdots, n$)
4. $T(j, s)$ denote the truth value of: There is a schedule for $J_1, \cdots, J_j$, for which $s$ is the corresponding configuration
Dynamic Programming

1. $J_1, \cdots, J_k$ jobs of the transformed instance
2. A schedule configuration $s = (t_1, \cdots, t_m, c)$ is a $(m + 1)$-tuple
   - $t_i$ completion time of machine $i$
   - $c$ total cost
3. $V_j$ a set of these tuples (f.a. $j = 1, \cdots, n$)

$T(j, s)$ denote the truth value of: There is a schedule for $J_1, \cdots, J_j$, for which $s$ is the corresponding configuration
Dynamic Programming

1. $J_1, \cdots, J_k$ jobs of the transformed instance
2. A schedule configuration $s = (t_1, \cdots, t_m, c)$ is a $(m + 1)$-tuple
   - $t_i$ completion time of machine $i$
   - $c$ total cost
3. $V_j$ a set of these tuples (f.a. $j = 1, \cdots, n$)
   - for every $j = 1, \cdots, n$ there is a tuple $v \in V_j$ whose entries are 0
4. $T(j, s)$ denote the truth value of: There is a schedule for $J_1, \cdots, J_j$, for which $s$ is the corresponding configuration
Dynamic Programming

1. $J_1, \ldots, J_k$ jobs of the transformed instance

2. A schedule configuration $s = (t_1, \ldots, t_m, c)$ is a $(m+1)$-tuple
   - $t_i$ completion time of machine $i$
   - $c$ total cost

3. $V_j$ a set of these tuples (f.a. $j = 1, \ldots, n$)
   - for every $i = 1, \ldots, m$ there is a tuple $v \in V_j$ whose entries are 0

4. $T(j,s)$ denote the truth value of: There is a schedule for $J_1, \ldots, J_j$, for which $s$ is the corresponding configuration
Dynamic Programming

1. $J_1, \cdots, J_k$ jobs of the transformed instance

2. A schedule configuration $s = (t_1, \cdots, t_m, c)$ is a $(m+1)$-tuple
   - $t_i$ completion time of machine $i$
   - $c$ total cost

3. $V_j$ a set of these tuples (f.a. $j = 1, \cdots, n$)
   - for every $i = 1, \cdots, m$ there is a tuple $v \in V_j$ whose entries are 0
     - Exception: the $i$th component, which is $p_{ij}$,
     - the $(m+1)$th component, which is $c_j$

4. $T(j, s)$ denote the truth value of: There is a schedule for $J_1, \cdots, J_j$, for which $s$ is the corresponding configuration
Dynamic Programming

1. $J_1, \cdots, J_k$ jobs of the transformed instance

2. A schedule configuration $s = (t_1, \cdots, t_m, c)$ is a $(m+1)$-tuple
   - $t_i$ completion time of machine $i$
   - $c$ total cost

3. $V_j$ a set of these tuples (f.a. $j = 1, \cdots, n$)
   - for every $i = 1, \cdots, m$ there is a tuple $v \in V_j$ whose entries are 0
     - Exception: the $i$th component, which is $p_{ij}$,
     - the $(m+1)$th component, which is $c_{ij}$

4. $T(j, s)$ denote the truth value of: There is a schedule for $J_1, \cdots, J_j$, for which $s$ is the corresponding configuration
Dynamic Programming

1. $J_1, \cdots, J_k$ jobs of the transformed instance
2. A schedule configuration $s = (t_1, \cdots, t_m, c)$ is a $(m+1)$-tuple
   - $t_i$ completion time of machine $i$
   - $c$ total cost
3. $V_j$ a set of these tuples (f.a. $j = 1, \cdots, n$)
   - for every $i = 1, \cdots, m$ there is a tuple $v \in V_j$ whose entries are 0
     - Exception: the $i$th component, which is $p_{ij}$,
     - the $(m+1)$th component, which is $c_{ij}$
4. $T(j, s)$ denote the truth value of: There is a schedule for $J_1, \cdots, J_j$, for which $s$ is the corresponding configuration
   - Calculate all $T(j, s)$:
     \[
     T(j, v) = \begin{cases} 
     \text{true} & \text{if } v \in V_j \\
     \text{false} & \text{if } v \notin V_j 
     \end{cases}
     
     T(j, s) = \bigvee_{v \in V_j} T(j-1, s-v) \text{ for } j = 2, \cdots, k
Dynamic Programming

1. $J_1, \cdots, J_k$ jobs of the transformed instance
2. A schedule configuration $s = (t_1, \cdots, t_m, c)$ is a $(m + 1)$-tuple
   - $t_i$ completion time of machine $i$
   - $c$ total cost
3. $V_j$ a set of these tuples (f.a. $j = 1, \cdots, n$)
   - for every $i = 1, \cdots, m$ there is a tuple $v \in V_j$ whose entries are 0
     - Exception: the $i$th component, which is $p_{ij}$,
     - the $(m + 1)$th component, which is $c_{ij}$
4. $T(j, s)$ denote the truth value of: There is a schedule for $J_1, \cdots, J_j$, for which $s$ is the corresponding configuration
   - Calculate all $T(j, s)$:
     - $T(1, v) = \begin{cases} true, & \text{if } v \in V_1 \\ false, & \text{if } v \not\in V_1 \end{cases}$
     - $T(j, s) = \bigvee_{v \in V_j} T(j-1, s-v)$ for $j = 2, \cdots, k$
1. $J_1, \cdots, J_k$ jobs of the transformed instance

2. A schedule configuration $s = (t_1, \cdots, t_m, c)$ is a $(m + 1)$-tuple
   - $t_i$ completion time of machine $i$
   - $c$ total cost

3. $V_j$ a set of these tuples (f.a. $j = 1, \cdots, n$)
   - for every $i = 1, \cdots, m$ there is a tuple $v \in V_j$ whose entries are 0
     - Exception: the $i$th component, which is $p_{ij}$,
     - the $(m + 1)$th component, which is $c_{ij}$

4. $T(j, s)$ denote the truth value of: There is a schedule for $J_1, \cdots, J_j$, for which $s$ is the corresponding configuration
   - Calculate all $T(j, s)$:
     - $T(1, v) = \begin{cases} 
       \text{true}, & \text{if } v \in V_j \\
       \text{false}, & \text{if } v \notin V_j
     \end{cases}$
     - $T(j, s) = \bigvee_{v \in V_j; v \leq s} T(j - 1, s - v)$ for $j = 2, \cdots, k$
Dynamic Programming

1. $J_1, \cdots, J_k$ jobs of the transformed instance

2. A schedule configuration $s = (t_1, \cdots, t_m, c)$ is a $(m + 1)$-tuple
   - $t_i$ completion time of machine $i$
   - $c$ total cost

3. $V_j$ a set of these tuples (f.a. $j = 1, \cdots, n$)
   - for every $i = 1, \cdots, m$ there is a tuple $v \in V_j$ whose entries are 0
     - Exception: the $i$th component, which is $p_{ij}$,
     - the $(m + 1)$th component, which is $c_{ij}$

4. $T(j, s)$ denote the truth value of: There is a schedule for $J_1, \cdots, J_j$, for which $s$ is the corresponding configuration
   - Calculate all $T(j, s)$:
     - $T(1, v) = \begin{cases} \text{true}, & \text{if } v \in V_j \\ \text{false}, & \text{if } v \notin V_j \end{cases}$
     - $T(j, s) = \bigvee_{v \in V_j; v \leq s} T(j - 1, s - v)$ for $j = 2, \cdots, k$
Dynamic Programming

1. $J_1, \cdots, J_k$ jobs of the transformed instance
2. A schedule configuration $s = (t_1, \cdots, t_m, c)$ is a $(m + 1)$-tuple
   - $t_i$ completion time of machine $i$
   - $c$ total cost
3. $V_j$ a set of these tuples (f.a. $j = 1, \cdots, n$)
   - for every $i = 1, \cdots, m$ there is a tuple $v \in V_j$ whose entries are 0
     - Exception: the $i$th component, which is $p_{ij}$,
     - the $(m + 1)$th component, which is $c_{ij}$
4. $T(j, s)$ denote the truth value of: There is a schedule for $J_1, \cdots, J_j$, for which $s$ is the corresponding configuration
   - Calculate all $T(j, s)$:
     - $T(1, v) = \begin{cases} 
     \text{true}, & \text{if } v \in V_j \\
     \text{false}, & \text{if } v \notin V_j
     \end{cases}$
     - $T(j, s) = \vee_{v \in V_j; v \leq s} T(j - 1, s - v)$ for $j = 2, \cdots, k$
Dynamic Programming

1. $J_1, \cdots, J_k$ jobs of the transformed instance
2. A schedule configuration $s = (t_1, \cdots, t_m, c)$ is a $(m + 1)$-tuple
   - $t_i$ completion time of machine $i$
   - $c$ total cost
3. $V_j$ a set of these tuples (f.a. $j = 1, \cdots, n$)
   - for every $i = 1, \cdots, m$ there is a tuple $v \in V_j$ whose entries are 0
     - Exception: the $i$th component, which is $p_{ij}$,
     - the $(m + 1)$th component, which is $c_{ij}$
4. $T(j, s)$ denote the truth value of: There is a schedule for $J_1, \cdots, J_j$, for which $s$ is the corresponding configuration
   - Calculate all $T(j, s)$:
     - $T(1, v) = \begin{cases} 
     \text{true}, & \text{if } v \in V_j \\
     \text{false}, & \text{if } v \notin V_j
     \end{cases}$
     - $T(j, s) = \bigvee_{v \in V_j; v \leq s} T(j - 1, s - v)$ for $j = 2, \cdots, k$
Summary

- **up to now**
  - All jobs rounded
  - Every job has a profile
  - Number of profile constant
  - Grouping $\Rightarrow$ number of jobs constant
  - Schedule per dynamic programming
Summary

- up to now
  - All jobs rounded
  - Every job has a profile
  - Number of profile constant
  - Grouping $\rightarrow$ number of jobs constant
  - Schedule per dynamic programming
Summary

- up to now
  - All jobs rounded
  - Every job has a profile
    - Number of profile constant
    - Grouping $\implies$ number of jobs constant
    - Schedule per dynamic programming
Summary

- up to now
  - All jobs rounded
  - Every job has a profile
  - Number of profile constant
  - Grouping $\Rightarrow$ number of jobs constant
  - Schedule per dynamic programming
**Summary**

- up to now
  - All jobs rounded
  - Every job has a profile
  - Number of profile constant
  - Grouping $\implies$ number of jobs constant
  - Schedule per dynamic programming
Summary

- up to now
  - All jobs rounded
  - Every job has a profile
  - Number of profile constant
  - Grouping $\implies$ number of jobs constant
  - Schedule per dynamic programming
Lemma

For the problem Unrelated Parallel Machines with Costs there is a FPTAS that runs in $O(n) + (\log \frac{m}{\varepsilon})^{O(m^2)}$.

Without proof
Outlook and Discussion

- Implementing the algorithm in Java (quite slow)
- For which other problem would this algorithm match?
- Could the running time be better?
Outlook and Discussion

- Implementing the algorithm in Java (quite slow)
- For which other problem would this algorithm match?
- Could the running time be better?
Outlook and Discussion

- Implementing the algorithm in Java (quite slow)
- For which other problem would this algorithm match?
- Could the running time be better?
Thanks for your attention