Scheduling on Unrelated Parallel Machines

Piotr Wojciechowski[1]

[1] Lane Department of Computer Science and Electrical Engineering
West Virginia University

April 17, 2014

Outline

1. Scheduling on Unrelated Parallel Machines
   - Definition
   - IP formuation
   - Algorithm

Outline

1. Scheduling on Unrelated Parallel Machines
   - Definition
   - IP formuation
   - Algorithm

Definition

We are given a set of n jobs $J = \{j_1, \ldots, j_n\}$,

Definition

We are given a set of n jobs $J = \{j_1, \ldots, j_n\}$, a set of m machines $M\{m_1, \ldots, m_n\}$,

Definition

We are given a set of n jobs $J = \{j_1, \ldots, j_n\}$, a set of m machines $M\{m_1, \ldots, m_n\}$, and a time matrix **P** where p_{ij} is the time it takes the ith machine to run the jth job.

Definition

We are given a set of n jobs $J = \{j_1, \ldots, j_n\}$, a set of m machines $M\{m_1, \ldots, m_n\}$, and a time matrix **P** where p_{ij} is the time it takes the ith machine to run the jth job. Our goal is to schedule the jobs to that the makespan is minimized.

Definition

We are given a set of n jobs $J = \{j_1, \ldots, j_n\}$, a set of m machines $M\{m_1, \ldots, m_n\}$, and a time matrix $\mathbf{P}$ where p_{ij} is the time it takes the ith machine to run the jth job. Our goal is to schedule the jobs to that the makespan is minimized.

Note

This is a generalization of minimum makespan scheduling in which the running time for each job depends on the machine to which it is assigned.

Definition

We are given a set of n jobs $J = \{j_1, \ldots, j_n\}$, a set of m machines $M\{m_1, \ldots, m_n\}$, and a time matrix $\mathbf{P}$ where p_{ij} is the time it takes the ith machine to run the jth job. Our goal is to schedule the jobs to that the makespan is minimized.

Note

This is a generalization of minimum makespan scheduling in which the running time for each job depends on the machine to which it is assigned. A opposed to the PTAS obtained for the original problem we will demonstrate a 2-approximation.

Outline

1. Scheduling on Unrelated Parallel Machines
   - Definition
   - IP formuation
   - Algorithm

Properties

The integer program for scheduling on unrelated parallel machines need to encode the following properties:

Properties

The integer program for scheduling on unrelated parallel machines need to encode the following properties:

1. Each job can be scheduled on at most one machine.

Properties

The integer program for scheduling on unrelated parallel machines need to encode the following properties:

1. Each job can be scheduled on at most one machine.
2. If job j_j is scheduled on the m_i machine the running time is p_{ij}.

Properties

The integer program for scheduling on unrelated parallel machines need to encode the following properties:

1. Each job can be scheduled on at most one machine.
2. If job j_j is scheduled on the m_i machine the running time is p_{ij}.
3. We want to minimize the longest total running time over all machines.

IP formulation

This problem can be represented by the following integer program where the variable x_{ij} is 1 if job j_j is assigned to machine m_i:

IP formulation

This problem can be represented by the following integer program where the variable x_{ij} is 1 if job j_j is assigned to machine m_i:

$$\min\ t$$

IP formulation

This problem can be represented by the following integer program where the variable x_{ij} is 1 if job j_j is assigned to machine m_i:

$$\min \ t$$

$$\sum_{i \in M} x_{ij} = 1 \qquad j \in J$$

IP formulation

This problem can be represented by the following integer program where the variable x_{ij} is 1 if job j_j is assigned to machine m_i:

$$\min \ t$$

$$\sum_{i \in M} x_{ij} = 1 \qquad j \in J$$

$$\sum_{j \in J} x_{ij} \cdot p_{ij} \leq t \qquad i \in M$$

IP formulation

This problem can be represented by the following integer program where the variable x_{ij} is 1 if job j_j is assigned to machine m_i:

$$\min\ t$$

$$\sum_{i \in M} x_{ij} = 1 \qquad j \in J$$

$$\sum_{j \in J} x_{ij} \cdot p_{ij} \leq t \qquad i \in M$$

$$x_{ij} \in \{0,1\} \qquad i \in M, j \in J$$

IP formulation

This problem can be represented by the following integer program where the variable x_{ij} is 1 if job j_j is assigned to machine m_i:

$$\begin{aligned} & \min\ t & \\ & \sum_{i \in M} x_{ij} = 1 & j \in J \\ & \sum_{j \in J} x_{ij} \cdot p_{ij} \leq t & i \in M \\ & x_{ij} \in \{0,1\} & i \in M, j \in J \end{aligned}$$

Integrality Gap

The main issue with this formulation is that it has an unbounded integrality gap.

IP formulation

This problem can be represented by the following integer program where the variable x_{ij} is 1 if job j_j is assigned to machine m_i:

$$\begin{aligned} & \min\ t & \\ & \sum_{i \in M} x_{ij} = 1 & j \in J \\ & \sum_{j \in J} x_{ij} \cdot p_{ij} \leq t & i \in M \\ & x_{ij} \in \{0,1\} & i \in M, j \in J \end{aligned}$$

Integrality Gap

The main issue with this formulation is that it has an unbounded integrality gap. For example consider trying to schedule one job on m machines where $p_{i1} = m$ for all $i \in M$.

Parametric Pruning

We can avoid this situation by making the LP unable to schedule jobs longer than the total makespan.

Parametric Pruning

We can avoid this situation by making the LP unable to schedule jobs longer than the total makespan. This cannot be done in a single LP but can be expressed as a family of Linear Programs.

Parametric Pruning

We can avoid this situation by making the LP unable to schedule jobs longer than the total makespan. This cannot be done in a single LP but can be expressed as a family of Linear Programs.

Pruned LPs

The new linear programs take the following form:

Parametric Pruning

We can avoid this situation by making the LP unable to schedule jobs longer than the total makespan. This cannot be done in a single LP but can be expressed as a family of Linear Programs.

Pruned LPs

The new linear programs take the following form:

1. For $T \in \mathbb{Z}^+$ create the set of machine-job pairs $S_T = \{(i,j) : x_{ij} \leq T\}$.

Parametric Pruning

We can avoid this situation by making the LP unable to schedule jobs longer than the total makespan. This cannot be done in a single LP but can be expressed as a family of Linear Programs.

Pruned LPs

The new linear programs take the following form:

1. For $T \in \mathbb{Z}^+$ create the set of machine-job pairs $S_T = \{(i,j) : x_{ij} \leq T\}$.
2. Create the linear program $LP(T)$ defined as

$$\sum_{i:(i,j)\in S_T} x_{ij} = 1 \qquad j \in J$$

Parametric Pruning

We can avoid this situation by making the LP unable to schedule jobs longer than the total makespan. This cannot be done in a single LP but can be expressed as a family of Linear Programs.

Pruned LPs

The new linear programs take the following form:

1. For $T \in \mathbb{Z}^+$ create the set of machine-job pairs $S_T = \{(i,j) : x_{ij} \leq T\}$.
2. Create the linear program $LP(T)$ defined as

$$\sum_{i:(i,j)\in S_T} x_{ij} = 1 \qquad j \in J$$

$$\sum_{j:(i,j)\in S_T} x_{ij} \cdot p_{ij} \leq t \qquad i \in M$$

Parametric Pruning

We can avoid this situation by making the LP unable to schedule jobs longer than the total makespan. This cannot be done in a single LP but can be expressed as a family of Linear Programs.

Pruned LPs

The new linear programs take the following form:

1. For $T \in \mathbb{Z}^+$ create the set of machine-job pairs $S_T = \{(i,j) : x_{ij} \leq T\}$.
2. Create the linear program $LP(T)$ defined as

$$\sum_{i:(i,j)\in S_T} x_{ij} = 1 \qquad j \in J$$

$$\sum_{j:(i,j)\in S_T} x_{ij} \cdot p_{ij} \leq t \qquad i \in M$$

$$x_{ij} \geq 0 \quad (i,j) \in S_T$$

Parametric Pruning

We can avoid this situation by making the LP unable to schedule jobs longer than the total makespan. This cannot be done in a single LP but can be expressed as a family of Linear Programs.

Pruned LPs

The new linear programs take the following form:

1. For $T \in \mathbb{Z}^+$ create the set of machine-job pairs $S_T = \{(i,j) : x_{ij} \leq T\}$.
2. Create the linear program $LP(T)$ defined as

$$\sum_{i:(i,j)\in S_T} x_{ij} = 1 \qquad j \in J$$

$$\sum_{j:(i,j)\in S_T} x_{ij} \cdot p_{ij} \leq t \qquad i \in M$$

$$x_{ij} \geq 0 \quad (i,j) \in S_T$$

Our goal is to find the smallest T such that $LP(T)$ is feasible, call this value T^* (note that T^* is still a lower bound for OPT).

Parametric Pruning

We can avoid this situation by making the LP unable to schedule jobs longer than the total makespan. This cannot be done in a single LP but can be expressed as a family of Linear Programs.

Pruned LPs

The new linear programs take the following form:

1. For $T \in \mathbb{Z}^+$ create the set of machine-job pairs $S_T = \{(i,j) : x_{ij} \leq T\}$.
2. Create the linear program $LP(T)$ defined as

$$\sum_{i:(i,j)\in S_T} x_{ij} = 1 \qquad j \in J$$

$$\sum_{j:(i,j)\in S_T} x_{ij} \cdot p_{ij} \leq t \qquad i \in M$$

$$x_{ij} \geq 0 \quad (i,j) \in S_T$$

Our goal is to find the smallest T such that $LP(T)$ is feasible, call this value T^* (note that T^* is still a lower bound for OPT). We will then take an extreme point solution to $LP(T^*)$ and round it to obtain an integer solution with makespan no more than $2 \cdot T^*$.

Properties of Extreme Point Solutions

We first need to show that an extreme point solution of $LP(T)$ satisfies certain properties.

Properties of Extreme Point Solutions

We first need to show that an extreme point solution of $LP(T)$ satisfies certain properties.

Theorem

Any extreme point solution to $LP(T)$ has at most $n+m$ nonzero variables.

Properties of Extreme Point Solutions

We first need to show that an extreme point solution of $LP(T)$ satisfies certain properties.

Theorem

Any extreme point solution to $LP(T)$ has at most $n+m$ nonzero variables.

Proof.

- Let $r = |S_t|$ be the number of variables on which $LP(T)$ is defined.

Properties of Extreme Point Solutions

We first need to show that an extreme point solution of $LP(T)$ satisfies certain properties.

Theorem

Any extreme point solution to $LP(T)$ has at most $n+m$ nonzero variables.

Proof.

- Let $r = |S_t|$ be the number of variables on which $LP(T)$ is defined.
- An extreme point solution must satisfy at least r constraints with equality.

Properties of Extreme Point Solutions

We first need to show that an extreme point solution of $LP(T)$ satisfies certain properties.

Theorem

Any extreme point solution to $LP(T)$ has at most $n+m$ nonzero variables.

Proof.

- Let $r = |S_t|$ be the number of variables on which $LP(T)$ is defined.
- An extreme point solution must satisfy at least r constraints with equality.
- At least $r-(n+m)$ of these must be non-negativity constraints.

Properties of Extreme Point Solutions

We first need to show that an extreme point solution of $LP(T)$ satisfies certain properties.

Theorem

Any extreme point solution to $LP(T)$ has at most $n+m$ nonzero variables.

Proof.

- Let $r = |S_t|$ be the number of variables on which $LP(T)$ is defined.
- An extreme point solution must satisfy at least r constraints with equality.
- At least $r-(n+m)$ of these must be non-negativity constraints.
- At least $r-(n+m)$ variables are set to 0.

Properties of Extreme Point Solutions

We first need to show that an extreme point solution of $LP(T)$ satisfies certain properties.

Theorem

Any extreme point solution to $LP(T)$ has at most $n+m$ nonzero variables.

Proof.

- Let $r = |S_t|$ be the number of variables on which $LP(T)$ is defined.
- An extreme point solution must satisfy at least r constraints with equality.
- At least $r-(n+m)$ of these must be non-negativity constraints.
- At least $r-(n+m)$ variables are set to 0.
- At most $n+m$ variables are nonzero.

□

Theorem

Any extreme point solution to LP(T) must set at least $n - m$ jobs integrally.

Theorem

Any extreme point solution to $LP(T)$ must set at least $n - m$ jobs integrally.

Proof.

- Let $\mathbf{x}$ be an extreme point solution to $LP(T)$.

Theorem

Any extreme point solution to $LP(T)$ must set at least $n - m$ jobs integrally.

Proof.

- Let **x** be an extreme point solution to $LP(T)$.
- Let α and β be the number of jobs set integrally and fractionally respectively.

Theorem

Any extreme point solution to $LP(T)$ must set at least $n - m$ jobs integrally.

Proof.

- Let **x** be an extreme point solution to $LP(T)$.
- Let α and β be the number of jobs set integrally and fractionally respectively.
- We have that $\alpha + \beta = n$.

Theorem

Any extreme point solution to LP(T) must set at least $n - m$ jobs integrally.

Proof.

- Let **x** be an extreme point solution to $LP(T)$.
- Let α and β be the number of jobs set integrally and fractionally respectively.
- We have that $\alpha + \beta = n$.
- Assigning a job fractionally results in at least two nonzero variables so $\alpha + 2 \cdot \beta \leq n + m$.

Theorem

Any extreme point solution to $LP(T)$ must set at least $n-m$ jobs integrally.

Proof.

- Let **x** be an extreme point solution to $LP(T)$.
- Let α and β be the number of jobs set integrally and fractionally respectively.
- We have that $\alpha + \beta = n$.
- Assigning a job fractionally results in at least two nonzero variables so $\alpha + 2 \cdot \beta \leq n + m$.
- Thus, $\beta \leq m$ and $\alpha \geq n - m$.

□

Handling Fractionally Assigned Jobs

We now need to specify how fractionally assigned jobs are handled.

Handling Fractionally Assigned Jobs

We now need to specify how fractionally assigned jobs are handled.

- Construct the bipartite graph $G = (M, J, E)$ where $(m_i, j_j) \in E$ if $x_{ij} > 0$.

Handling Fractionally Assigned Jobs

We now need to specify how fractionally assigned jobs are handled.

- Construct the bipartite graph $G = (M, J, E)$ where $(m_i, j_j) \in E$ if $x_{ij} > 0$.
- Let $F \subseteq J$ be the set of fractionally assigned jobs.

Handling Fractionally Assigned Jobs

We now need to specify how fractionally assigned jobs are handled.

- Construct the bipartite graph $G = (M, J, E)$ where $(m_i, j_j) \in E$ if $x_{ij} > 0$.
- Let $F \subseteq J$ be the set of fractionally assigned jobs.
- Define H to be the subgraph of G induced by the vertex set $M \cup F$.

Handling Fractionally Assigned Jobs

We now need to specify how fractionally assigned jobs are handled.

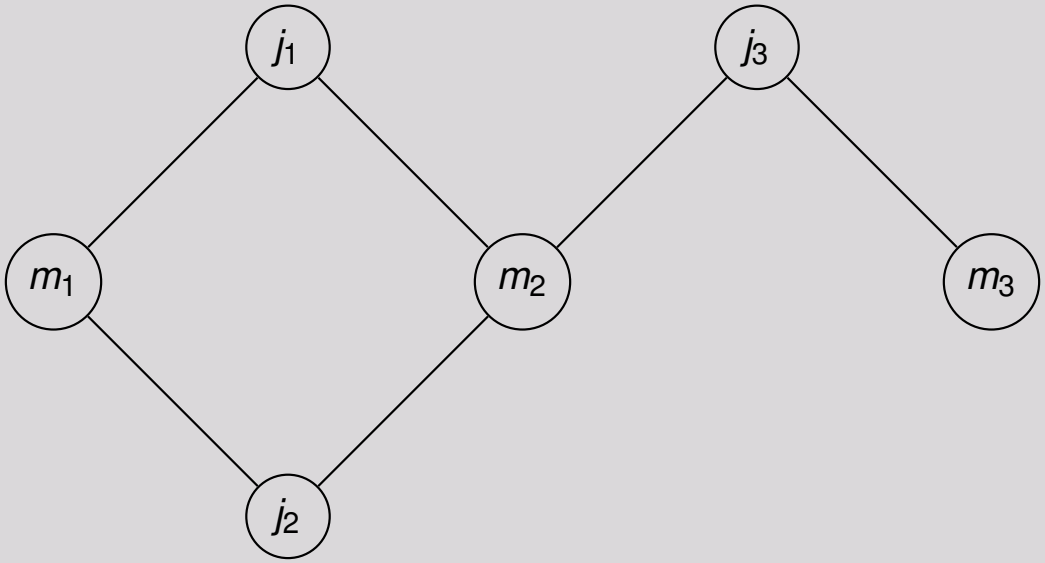
- Construct the bipartite graph $G = (M, J, E)$ where $(m_i, j_j) \in E$ if $x_{ij} > 0$.
- Let $F \subseteq J$ be the set of fractionally assigned jobs.
- Define H to be the subgraph of G induced by the vertex set $M \cup F$.
- Our goal is to find a matching in H so that every job in F is assigned to a unique machine in M.

Handling Fractionally Assigned Jobs

We now need to specify how fractionally assigned jobs are handled.

- Construct the bipartite graph $G = (M, J, E)$ where $(m_i, j_j) \in E$ if $x_{ij} > 0$.
- Let $F \subseteq J$ be the set of fractionally assigned jobs.
- Define H to be the subgraph of G induced by the vertex set $M \cup F$.
- Our goal is to find a matching in H so that every job in F is assigned to a unique machine in M.

Example



Theorem

G is a pseudo-forest.

Theorem

G is a pseudo-forest.

Proof.

- We need to show that each connected component of G is a pseudo-tree.

Theorem

G is a pseudo-forest.

Proof.

- We need to show that each connected component of G is a pseudo-tree.
- Let G_c be a component of G.

Theorem

G is a pseudo-forest.

Proof.

- We need to show that each connected component of G is a pseudo-tree.
- Let G_c be a component of G.
- We can restrict $\mathbf{x}$ and $LP(T)$ to the the machines and jobs in G_c.

Theorem

G is a pseudo-forest.

Proof.

- We need to show that each connected component of G is a pseudo-tree.
- Let G_c be a component of G.
- We can restrict $\mathbf{x}$ and $LP(T)$ to the the machines and jobs in G_c.
- Call the results $\mathbf{x_c}$ and $LP_c(T)$.

Theorem

G is a pseudo-forest.

Proof.

- We need to show that each connected component of G is a pseudo-tree.
- Let G_c be a component of G.
- We can restrict $\mathbf{x}$ and $LP(T)$ to the the machines and jobs in G_c.
- Call the results $\mathbf{x_c}$ and $LP_c(T)$.
- Since $\mathbf{x}$ is an extreme point solution of $LP(T)$ then we must have that $\mathbf{x_c}$ is an extreme point solution to $LP_c(T)$.

Theorem

G is a pseudo-forest.

Proof.

- We need to show that each connected component of G is a pseudo-tree.
- Let G_c be a component of G.
- We can restrict $\mathbf{x}$ and $LP(T)$ to the the machines and jobs in G_c.
- Call the results $\mathbf{x_c}$ and $LP_c(T)$.
- Since $\mathbf{x}$ is an extreme point solution of $LP(T)$ then we must have that $\mathbf{x_c}$ is an extreme point solution to $LP_c(T)$.
- We have that the number of non-zero variables in $\mathbf{x_c}$ cannot exceed the number of machines plus the number of jobs in $LP_c(T)$.

Theorem

G is a pseudo-forest.

Proof.

- We need to show that each connected component of G is a pseudo-tree.
- Let G_c be a component of G.
- We can restrict $\mathbf{x}$ and $LP(T)$ to the the machines and jobs in G_c.
- Call the results $\mathbf{x_c}$ and $LP_c(T)$.
- Since $\mathbf{x}$ is an extreme point solution of $LP(T)$ then we must have that $\mathbf{x_c}$ is an extreme point solution to $LP_c(T)$.
- We have that the number of non-zero variables in $\mathbf{x_c}$ cannot exceed the number of machines plus the number of jobs in $LP_c(T)$.
- Thus the number of edges in G_c cannot exceed the number of verticies.

□

Theorem

H has a matching in which each job in F is assigned to a machine in M.

Theorem

H has a matching in which each job in F is assigned to a machine in M.

Proof.

- In *G*, every integrally set job has exactly one edge incident to it.

Theorem

H has a matching in which each job in F is assigned to a machine in M.

Proof.

- In *G*, every integrally set job has exactly one edge incident to it.
- When we remove these jobs and edges the remaining graph is precisely *H*.

Theorem

H has a matching in which each job in F is assigned to a machine in M.

Proof.

- In *G*, every integrally set job has exactly one edge incident to it.
- When we remove these jobs and edges the remaining graph is precisely *H*.
- Thus, *H* is also a pseudo-forest.

Theorem

H has a matching in which each job in F is assigned to a machine in M.

Proof.

- In *G*, every integrally set job has exactly one edge incident to it.
- When we remove these jobs and edges the remaining graph is precisely *H*.
- Thus, *H* is also a pseudo-forest.
- In *H* each job vertex has degree at least 2 and so the leaves are all machine verticies.

Theorem

H has a matching in which each job in F is assigned to a machine in M.

Proof.

- In *G*, every integrally set job has exactly one edge incident to it.
- When we remove these jobs and edges the remaining graph is precisely *H*.
- Thus, *H* is also a pseudo-forest.
- In *H* each job vertex has degree at least 2 and so the leaves are all machine verticies.
- If a machine node *i* is a leaf we can assign to it its parent job node and remove both from the graph.

Theorem

H has a matching in which each job in F is assigned to a machine in M.

Proof.

- In *G*, every integrally set job has exactly one edge incident to it.
- When we remove these jobs and edges the remaining graph is precisely *H*.
- Thus, *H* is also a pseudo-forest.
- In *H* each job vertex has degree at least 2 and so the leaves are all machine verticies.
- If a machine node *i* is a leaf we can assign to it its parent job node and remove both from the graph.
- This process continues until the only edges in the graph form even length cycles.

Theorem

H has a matching in which each job in F is assigned to a machine in M.

Proof.

- In *G*, every integrally set job has exactly one edge incident to it.
- When we remove these jobs and edges the remaining graph is precisely *H*.
- Thus, *H* is also a pseudo-forest.
- In *H* each job vertex has degree at least 2 and so the leaves are all machine verticies.
- If a machine node *i* is a leaf we can assign to it its parent job node and remove both from the graph.
- This process continues until the only edges in the graph form even length cycles.
- Alternating edges in these cycles form a matching in which each remaining job vertex is assigned to a unique remaining machine vertex.

Theorem

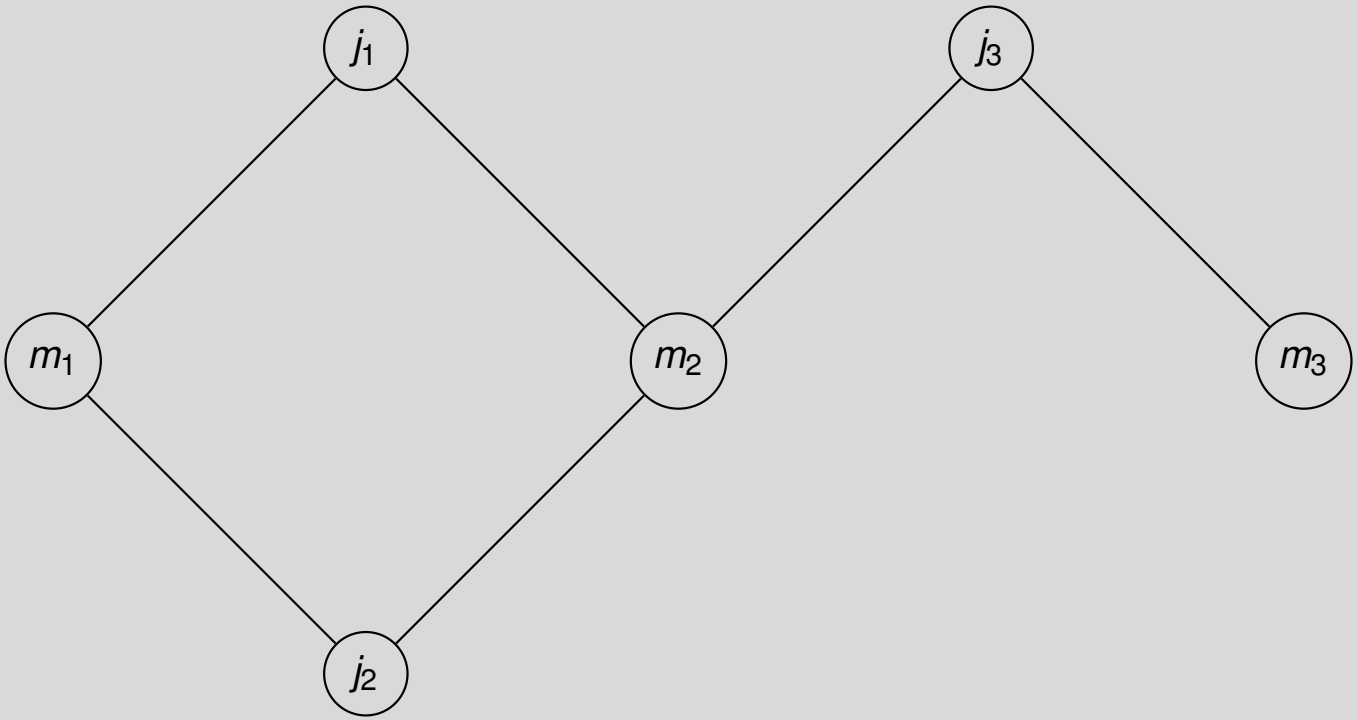
H has a matching in which each job in F is assigned to a machine in M.

Proof.

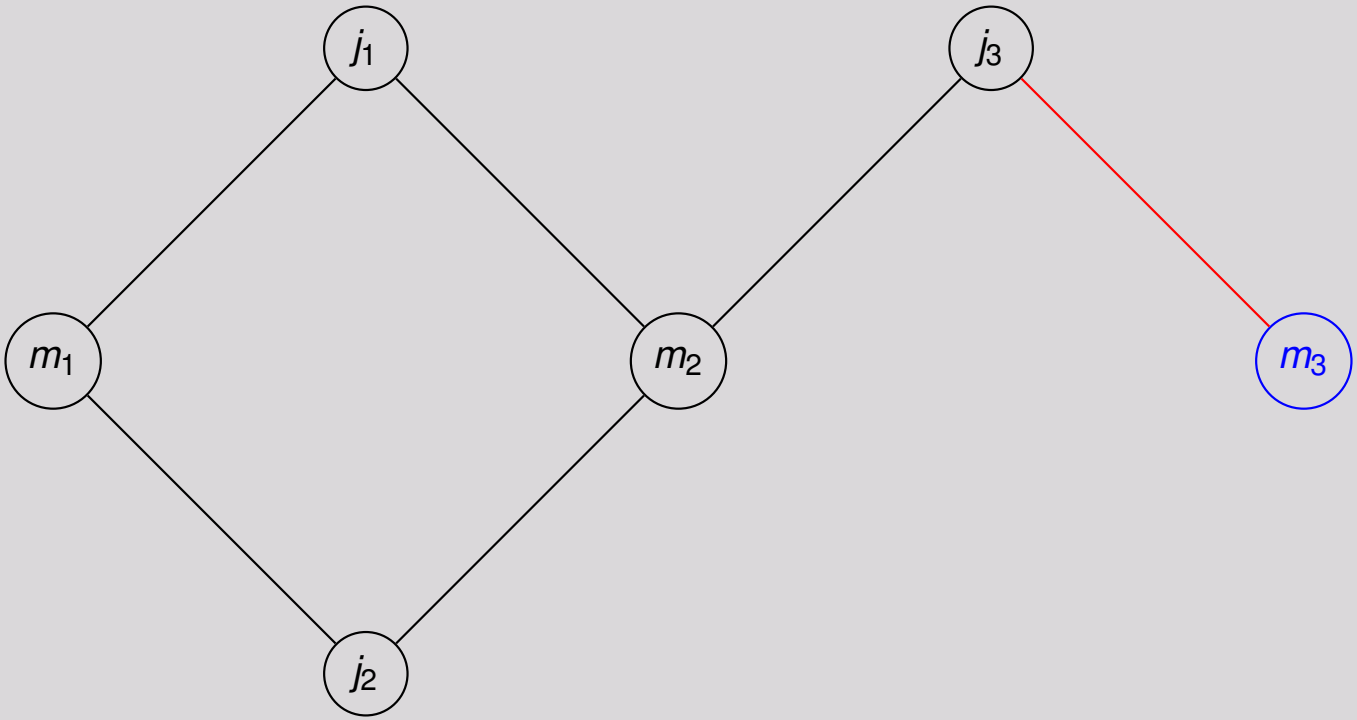
- In *G*, every integrally set job has exactly one edge incident to it.
- When we remove these jobs and edges the remaining graph is precisely *H*.
- Thus, *H* is also a pseudo-forest.
- In *H* each job vertex has degree at least 2 and so the leaves are all machine verticies.
- If a machine node *i* is a leaf we can assign to it its parent job node and remove both from the graph.
- This process continues until the only edges in the graph form even length cycles.
- Alternating edges in these cycles form a matching in which each remaining job vertex is assigned to a unique remaining machine vertex.
- Thus at the end of this procedure each job in *F* has been uniquely assigned to one of the machines in *M*.

□

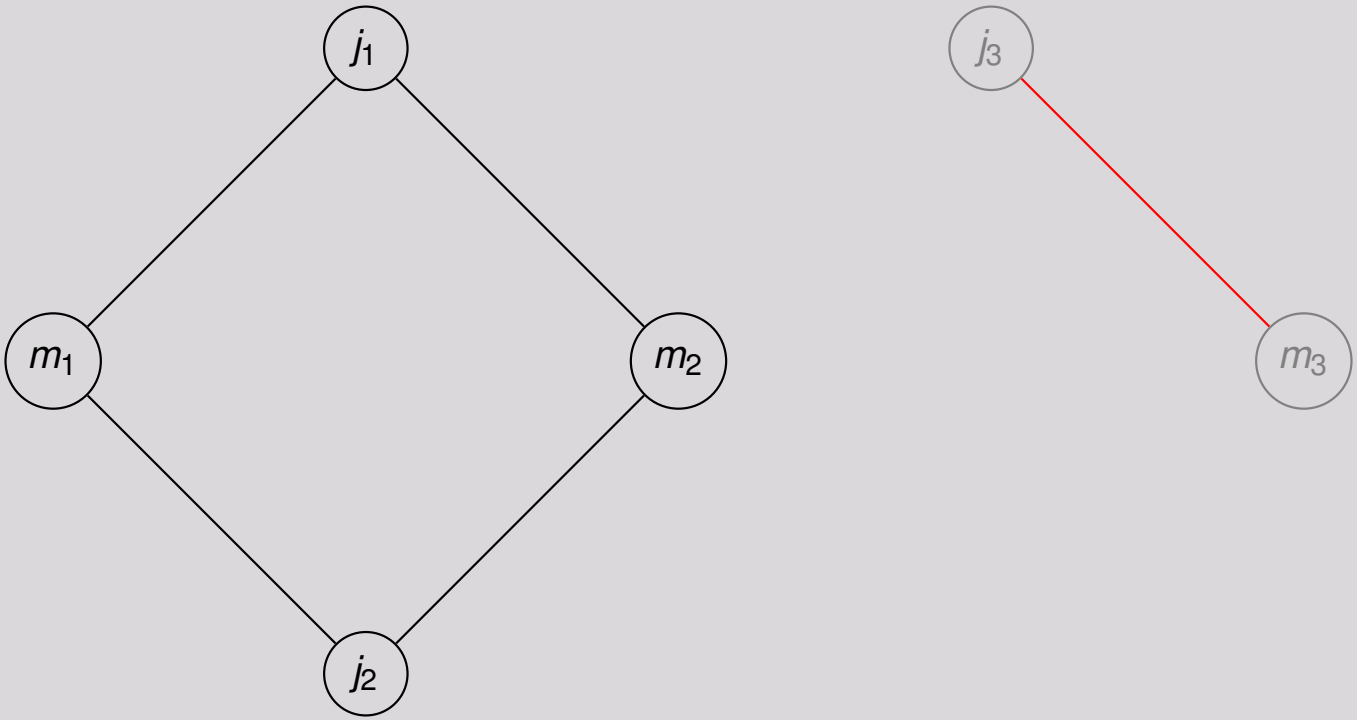
Example



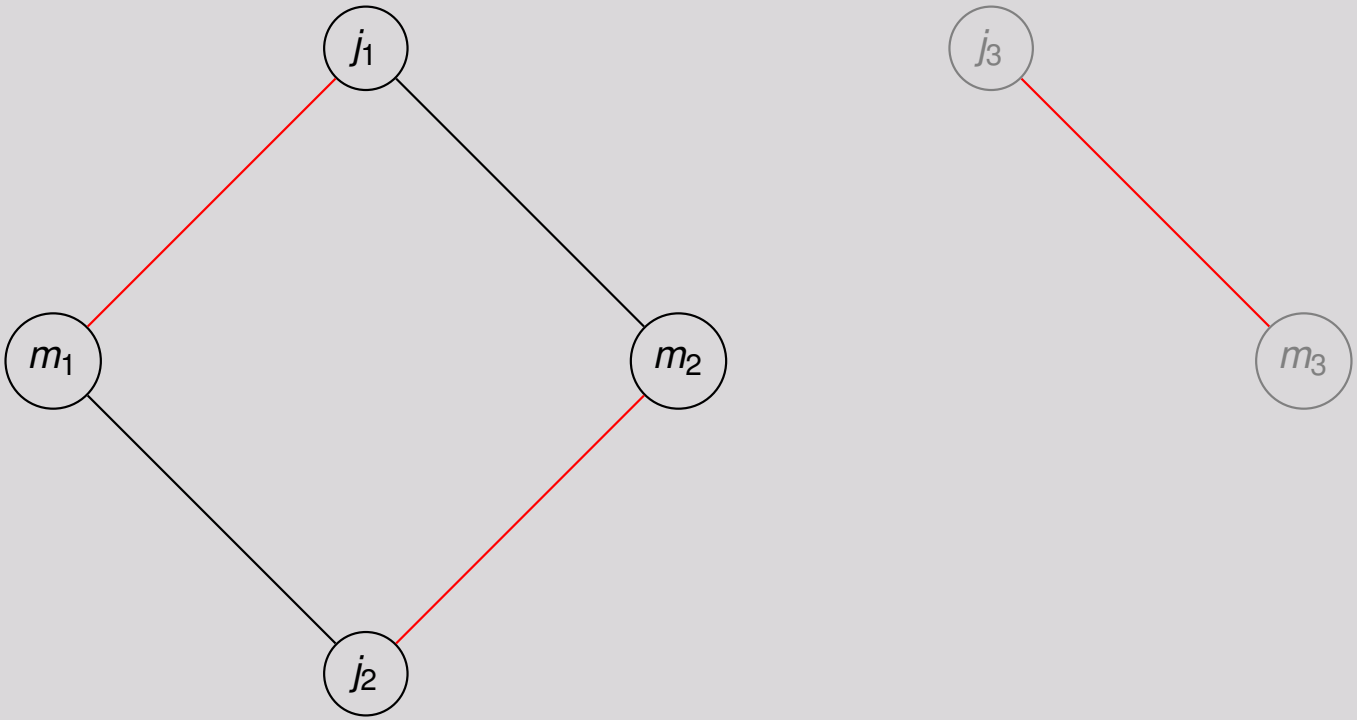
Example



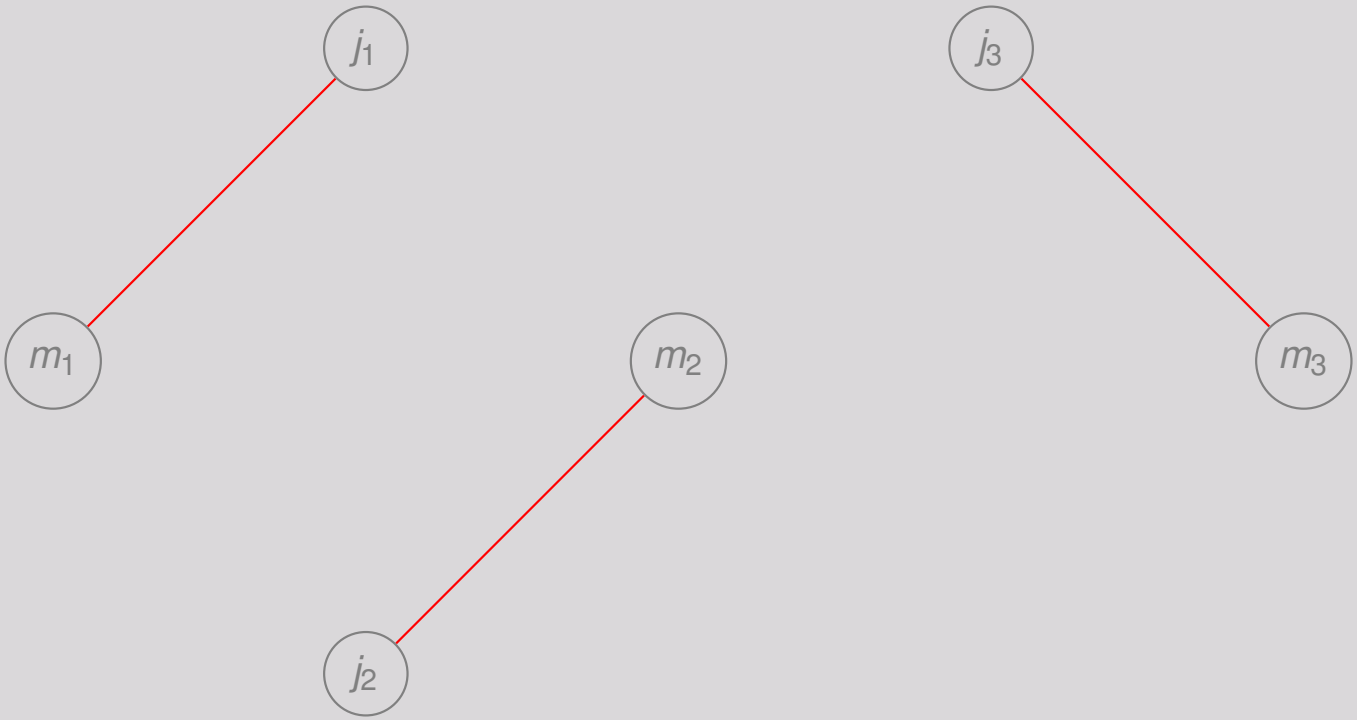
Example



Example



Example



Outline

1. Scheduling on Unrelated Parallel Machines
   - Definition
   - IP formuation
   - Algorithm

Algorithm

The approximation algorithm proceeds as follows:

1. Let α be the makespan of the schedule formed by assigning each job to the machine on which it has the shortest running time.

Algorithm

The approximation algorithm proceeds as follows:

1. Let α be the makespan of the schedule formed by assigning each job to the machine on which it has the shortest running time.
2. Using a binary search over the interval $[\frac{\alpha}{m}, \alpha]$ find the smallest $T \in \mathbb{Z}^+$ such that $LP(T)$ is linearly feasible. Call this value T^*.

Algorithm

The approximation algorithm proceeds as follows:

1. Let α be the makespan of the schedule formed by assigning each job to the machine on which it has the shortest running time.
2. Using a binary search over the interval $[\frac{\alpha}{m}, \alpha]$ find the smallest $T \in \mathbb{Z}^+$ such that $LP(T)$ is linearly feasible. Call this value T^*.
3. Find $\mathbf{x}$, an extreme point solution to $LP(T^*)$.

Algorithm

The approximation algorithm proceeds as follows:

1. Let α be the makespan of the schedule formed by assigning each job to the machine on which it has the shortest running time.
2. Using a binary search over the interval $[\frac{\alpha}{m}, \alpha]$ find the smallest $T \in \mathbb{Z}^+$ such that $LP(T)$ is linearly feasible. Call this value T^*.
3. Find $\mathbf{x}$, an extreme point solution to $LP(T^*)$.
4. Assign each integrally set job to the machine it is assigned in $\mathbf{x}$.

Algorithm

The approximation algorithm proceeds as follows:

1. Let α be the makespan of the schedule formed by assigning each job to the machine on which it has the shortest running time.
2. Using a binary search over the interval $[\frac{\alpha}{m}, \alpha]$ find the smallest $T \in \mathbb{Z}^+$ such that $LP(T)$ is linearly feasible. Call this value T^*.
3. Find $\mathbf{x}$, an extreme point solution to $LP(T^*)$.
4. Assign each integrally set job to the machine it is assigned in $\mathbf{x}$.
5. Assign each fractionally set job to a unique machine according to the matching found in H.

Theorem

The algorithm has an approximation factor of 2.

Theorem

The algorithm has an approximation factor of 2.

Proof.

- We have that $T^* \leq OPT$.

Theorem

The algorithm has an approximation factor of 2.

Proof.

- We have that $T^* \leq OPT$.
- The makespan of the schedule corresponding to $\mathbf{x}$ is no more than T^*.

Theorem

The algorithm has an approximation factor of 2.

Proof.

- We have that $T^* \leq OPT$.
- The makespan of the schedule corresponding to $\mathbf{x}$ is no more than T^*.
- Thus when restricted to integrally assigned jobs the makespan of the schedule is still at most T^*.

Theorem

The algorithm has an approximation factor of 2.

Proof.

- We have that $T^* \leq OPT$.
- The makespan of the schedule corresponding to $\mathbf{x}$ is no more than T^*.
- Thus when restricted to integrally assigned jobs the makespan of the schedule is still at most T^*.
- According to the matching in H each machine is assigned at most one originally fractionally assigned job.

Theorem

The algorithm has an approximation factor of 2.

Proof.

- We have that $T^* \leq OPT$.
- The makespan of the schedule corresponding to $\mathbf{x}$ is no more than T^*.
- Thus when restricted to integrally assigned jobs the makespan of the schedule is still at most T^*.
- According to the matching in H each machine is assigned at most one originally fractionally assigned job.
- This increases the makespan of the schedule by at most T^*.

Theorem

The algorithm has an approximation factor of 2.

Proof.

- We have that $T^* \leq OPT$.
- The makespan of the schedule corresponding to $\mathbf{x}$ is no more than T^*.
- Thus when restricted to integrally assigned jobs the makespan of the schedule is still at most T^*.
- According to the matching in H each machine is assigned at most one originally fractionally assigned job.
- This increases the makespan of the schedule by at most T^*.
- Thus the resultant schedule has total makespan at most $2 \cdot T^* \leq 2 \cdot OPT$.

□