Linear Programming

Scheduling problems

Linear programming (LP)

$$z(x) = c_1 x_1 + \ldots + c_n x_n \to \min$$

$$\begin{gathered} a_{11} x_1 + \ldots + a_{1n} x_n \geq b_1 \\ \vdots \\ a_{m1} x_1 + \ldots + a_{mn} x_n \geq b_m \\ x_i \geq 0 \;\text{ for }\; i = 1, \ldots, n. \end{gathered}$$

Extreme points

$$\sum_{j=1}^{n} c_j x_j \to \min$$

$$s.t. \quad \sum_{j=1}^{n} a_{ij} x_j \geq b_i, \qquad i = 1, \ldots, m$$

$$x_j \geq 0, \qquad j = 1, \ldots, n$$

- $x = \{x_1, \ldots, x_n\} \in R^n$
- A set $\boldsymbol{S}$ of all vectors x which satisfy all the constraints of LP is said to be a ***set of feasible solutions***, and any $x \in \boldsymbol{S}$ is ***a feasible solution*** or ***a feasible point***.
- If x^1, $x^2 \in \boldsymbol{S}$, then $\alpha\, x^1 + (1-\alpha)\, x^2 \in \boldsymbol{S}$, $0 \leq \alpha \leq 1$.
- A point $x \in \boldsymbol{S}$ is called ***extreme point solution***, it is a vertex of polyhedron, i.e. it cannot be expressed as a convex combination of two feasible solutions.

Properties of extreme points

- From linear programming theory we know that for any objective function there is an extreme point solution that is optimal.
- Both ellipsoid algorithm and simplex method find is an extreme point solution.

Why LP is so useful in approximation algorithms?

- Many combinatorial optimization problems can be stated as integer programs.
- Once this is done, the linear relaxation of this program provides a natural way of lower bounding the cost of the optimal solution.

Fundamental design techniques

- **LP-Rounding (Rounding)**
  - Solve the linear program.
  - Convert the fractional solution obtained into an integral solution.
  - The approximation guarantee is established by comparing the cost of the integral and fractional solutions.
- **Primal-dual schema**
  - An integral solution to the primal program and a feasible solution to the dual program are constructed iteratively.
  - Any feasible solution to the dual also provides a lower bound on OPT. The approximation guarantee is established by comparing the two solutions.

Integrality gap

- Given an LP-relaxation for a minimization problem Π, let $OPT_f(I)$ denote the cost of an optimal fractional solution to instance I, i.e., the objective function value of an optimal solution to the LP-relaxation.
- Define the **integrality gap** of the relaxation to be $\sup_I \frac{OPT(I)}{OPT_f(I)}$.

Approximation factor and integrality gap

- If the cost of the solution found by the algorithm is compared directly with the cost of an optimal fractional solution (or a feasible dual solution), as is done in most algorithms, the best approximation factor we can hope to prove is the integrality gap of relaxation.
- Interestingly enough, for many problems, both techniques have been successful in yielding algorithms having guarantees essentially equal to the integrality gap of the relaxation.

$R||C_{\max}$

- Given a set $\boldsymbol{J}=\{1,\ldots, n\}$ of jobs and
- a set $\boldsymbol{M}=\{M_1,\ldots, M_m\}$ of machines.
- For each $j \in \boldsymbol{J}$ and $M_i \in \boldsymbol{M}$, $p_{ij} \geq 0$, the time taken to process job j on machine M_i.
- The problem is to schedule the jobs on the machines so as to minimize the makespan.

ILP ($R||C_{\max}$)

$$\begin{aligned}
\text{minimize} \quad & C_{\max} \\
\text{s.t.} \quad & \sum_{i \in M} x_{ij} = 1, \qquad j = 1, \ldots, n, \\
& \sum_{j \in J} x_{ij} p_{ij} \le C_{\max}, \quad i = 1, \ldots, m, \\
& x_{ij} \in \{0, 1\} \qquad i = 1, \ldots, m, \quad j = 1, \ldots, n
\end{aligned}$$

x_{ij} is an indicator variable denoting whether job j is scheduled on M_i. The first set of constraints ensure that each job is scheduled on one of the machines. The second set ensures that each machine has processing time of at most $C_{\max}$.

Remark

ILP ($R||C_{\max}$) has unbounded integrality gap.

- Instance: $J_1 : p_{i1} = m \ \ (i=1,\ldots, m)$.
- $C_{\max}$ (ILP) $= m$.
- $C_{\max}$ (LP) $= 1$.

LP ($R||C_{\max}$)

- Why the LP obtains the solution much better than an optimal solution of the ILP?
- The ILP automatically sets x_{ij} to 0, if $p_{ij} > t$.
- The LP is allowed to set these variables to nonzero values, and thereby obtain a cheaper solution.
- We can improve LP to add the following constraint $\forall i = 1,\ldots,m, j=1,\ldots,n$: if $p_{ij} > t$, then $x_{ij}=0$.
- However, this is not a linear constraint!

Parametric pruning

- The parameter will be $T \in \mathbf{Z}^+$, which is our guess for a lower bound on the optimal makespan. The parameter will enable us to prune away all job-machine pairs such that $p_{ij} > T$. Define $S_T = \{(i, j) | \, p_{ij} \leq T\}$.
- We will define a family of linear programs, LP(T), one for each value of parameter $T \in \mathbf{Z}^+$. LP(T) uses the variables x_{ij} for $(i, j) \in S_T$ only, and asks if there is a feasible, fractional schedule of makespan $\leq T$ using the restricted possibilities.

LP(T)

- $T \in \mathbf{Z}^{+}$: $S_T = \{(i, j) | p_{ij} \leq T\}$.

$$\mathrm{LP}(T):$$

$$\sum_{i:(i,j)\in S_T} x_{ij} = 1, \quad j = 1, \ldots, n$$

$$\sum_{j:(i,j)\in S_T} x_{ij} p_{ij} \leq T, \quad i = 1, \ldots, m$$

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

Properties of extreme point solutions

- **Lemma 9.1**

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

Proof of Lemma 9.1

- Let $r = |S_T|$ represent the number of variables on which LP(T) is defined.
- A feasible solution to LP(T) is an extreme point solution $\Leftrightarrow$ it corresponds to setting r linearly independent constraints of LP(T) to equality.
- Of these r linearly independent constraints, at least $r - (n + m)$ must be chosen from the third set of constraints.
- The corresponding variables are set to 0.

Proof of Lemma 9.1

- Let $r = |S_T|$ represent the number of variables on which LP(T) is defined.
- A feasible solution to LP(T) is an extreme point solution $\Leftrightarrow$ it corresponds to setting r linearly independent constraints of LP(T) to equality.
- Of these r linearly independent constraints, at least $r - (n + m)$ must be chosen from the third set of constraints.
- The corresponding variables are set to 0.
- So, any extreme point solution has at most $n + m$ nonzero variables.

Definition

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

  We will say that job j is **integrally set** in $\boldsymbol{x}$ if it is entirely assigned to one machine. Otherwise, we will say that job j is **fractionally set**.

Properties of extreme point solutions

- **Corollary 9.2**

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

Proof

- Let
  - $\boldsymbol{x}$ be an extreme point solution to LP(T)
  - α be the number of jobs integrally set by $\boldsymbol{x}$
  - β be the number of jobs fractionally set by $\boldsymbol{x}$
- $\alpha + \beta = n$

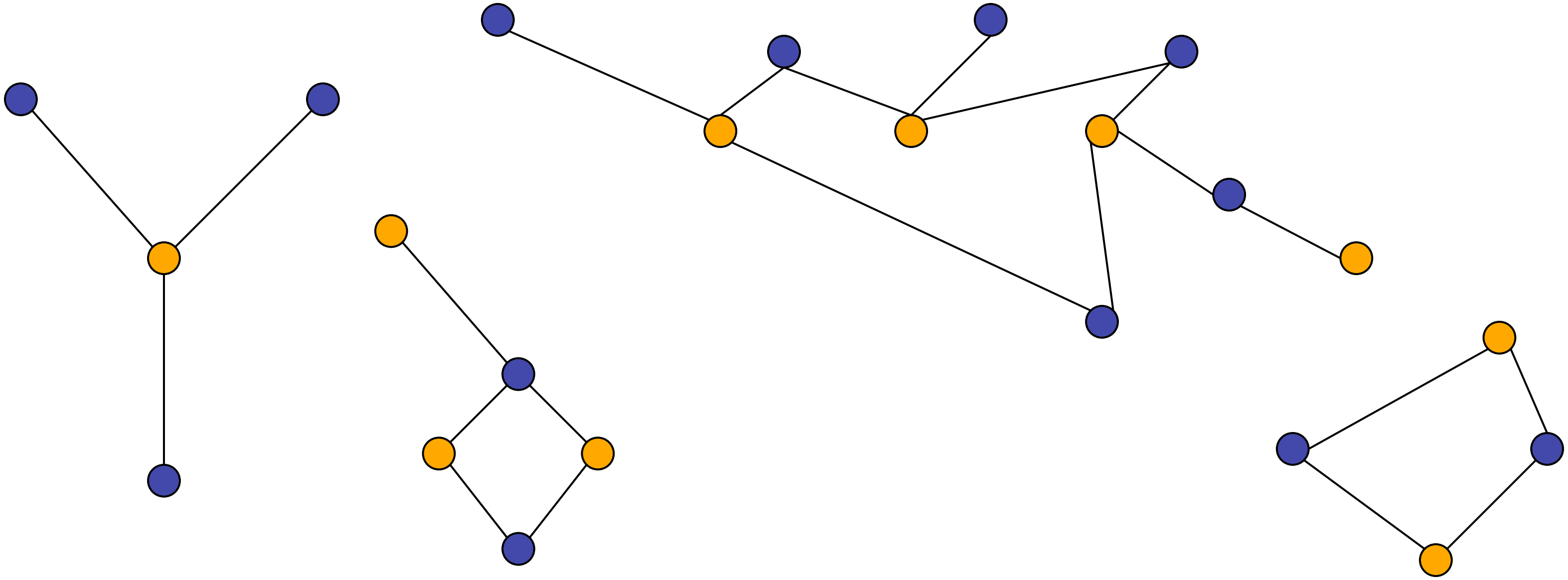
Proof

- Let
  - $\boldsymbol{x}$ be an extreme point solution to LP(T)
  - α be the number of jobs integrally set by $\boldsymbol{x}$
  - β be the number of jobs fractionally set by $\boldsymbol{x}$
- $\alpha + \beta = n$
- Each fractional job is assigned to at least 2 machines and therefore results in at least 2 nonzero entries in x.
- $\alpha + 2\beta \leq n + m$.
- $\beta \leq m$ and $\alpha \geq n - m$.

Bipartite graph

$$G = \left(J \bigcup M, E\right), \quad E = \left\{(i, j) \mid x_{ij} \neq 0\right\}.$$

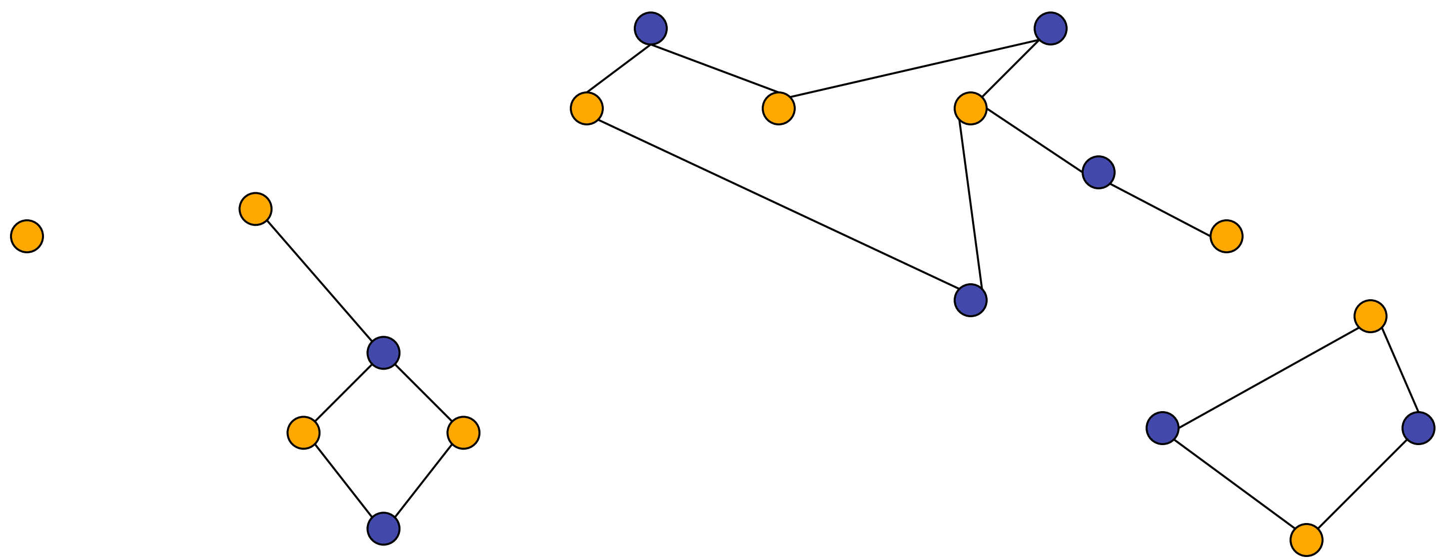
Machines and **Jobs**



$F \subset J$, F is the set of jobs that are fractionally set in $\boldsymbol{x}$

$$H = \left(F \bigcup M, E\right), \quad E = \left\{(i, j) \mid 0 < x_{ij} < 1\right\}.$$

Machines and **Jobs**



Matching

A matching in H will be called a ***perfect matching*** if it matches every job $J_i \in F$.

The rounding procedure uses the fact that graph H has a perfect matching.

Binary search

The algorithm starts by computing the range in which it finds the right value of T. For this, it constructs the greedy schedule, in which each job is assigned to the machine on which it has the smallest processing time. Let α be the makespan of this schedule. Then the range is

$$\alpha/m \leq C_{\max}(\sigma^*) \leq \alpha.$$

Algorithm LST (Lenstra, Shmoys, Tardosh 1990)

Input ($J=\{1,..., n\}$, $\boldsymbol{M}=\{M_1,..., M_m\}$, p: $J\times M \rightarrow \mathbf{Q}^+$)

1. By a binary search in the interval $[\alpha/m,\alpha]$, find the smallest value of $T\in\mathbf{Z}^+$ for which LP(T) has a feasible solution. Let this value be T^*.
2. Find an extreme point solution, say $\boldsymbol{x}$, to LP(T^*) .
3. Assign all integrally set jobs to machines as in $\boldsymbol{x}$.
4. Construct graph H and find a perfect matching μ in it.
5. Assign fractionally set jobs to machines according to matching. Let σ be the obtained schedule.

Output (σ)

Pseudo-Forest

- We will say that a connected graph on vertex set V is a ***pseudo-tree*** if it contains at most $|V|$ edges.
- A graph is a ***pseudo-forest*** if each its connected components is a pseudo-tree.

Lemma 9.3 Graph G is a pseudo-forest.

Proof of Lemma 9.3

- Consider a connected component G_C.
- Restrict LP(T) and $\boldsymbol{x}$ to the jobs and machines of G_C only, to obtain LP$_C$(T) and $\boldsymbol{x}_C$.
- The important observation is that $\boldsymbol{x}_C$ must be an extreme point solution to LP$_C$(T).
- Lemma 9.1 $\Rightarrow \boldsymbol{x}_C$ has at most $n_C + m_C$ nonzero variables $\Rightarrow G_C$ has at most $n_C + m_C$ edges $\Rightarrow$ G_C is a pseudo-tree.

Perfect Matching

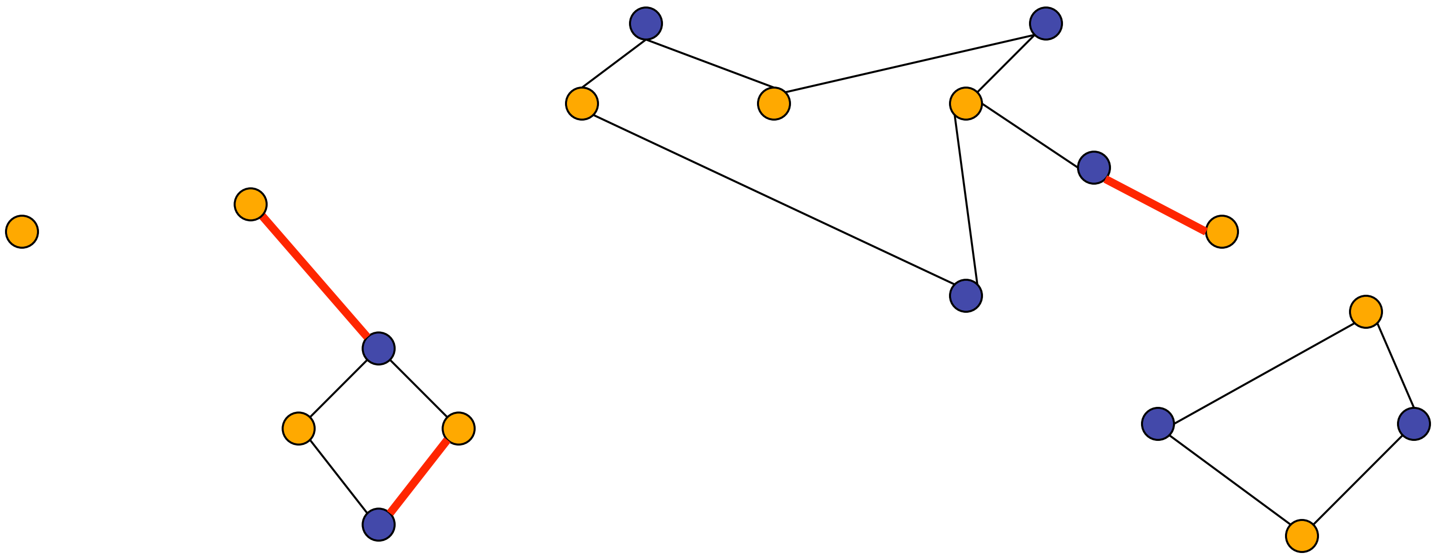
- **Lemma 9.4**

  Graph H has a perfect matching.

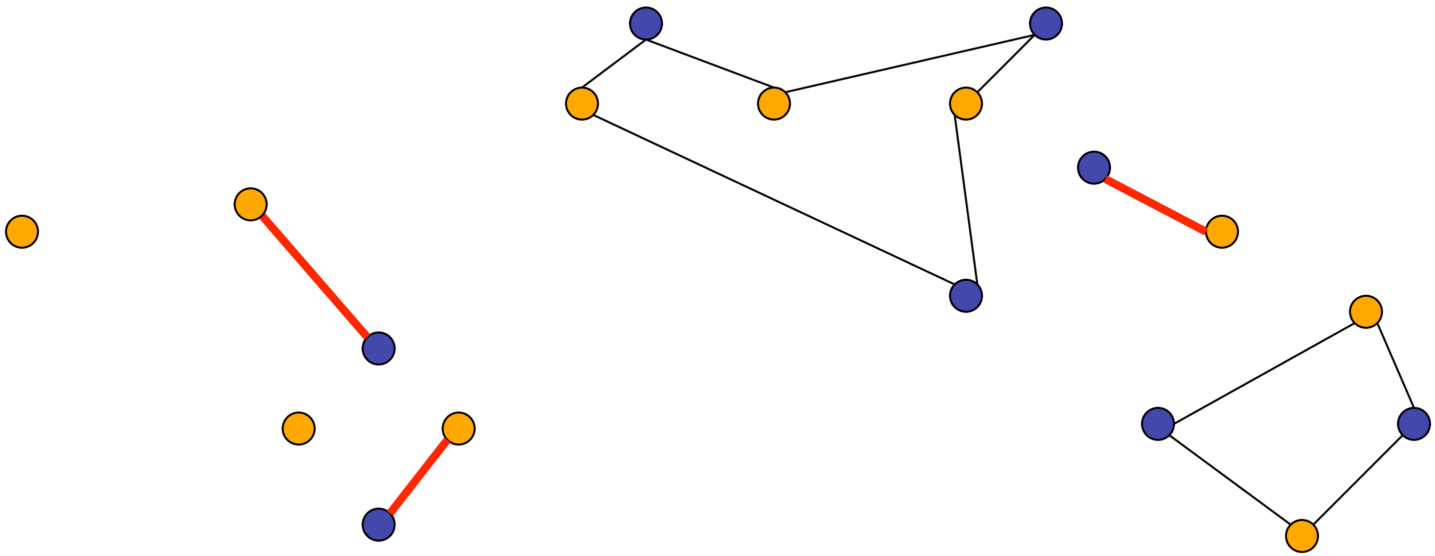
Proof of Lemma 9.4

- Each job that is integrally set in $\boldsymbol{x}$ has exactly one edge incident at it in G. Remove these jobs, together with their incident edges, from G. Clearly the remaining graph is H.
- Since an equal number of edges and vertices were removed, H is also pseudo-forest.
- In H, each job has a degree of at least 2.
- So, all leaves in H must be machines.
- Keep matching a leaf with the job it is incident to, and remove them both from the graph.

Machines and Jobs and Matching



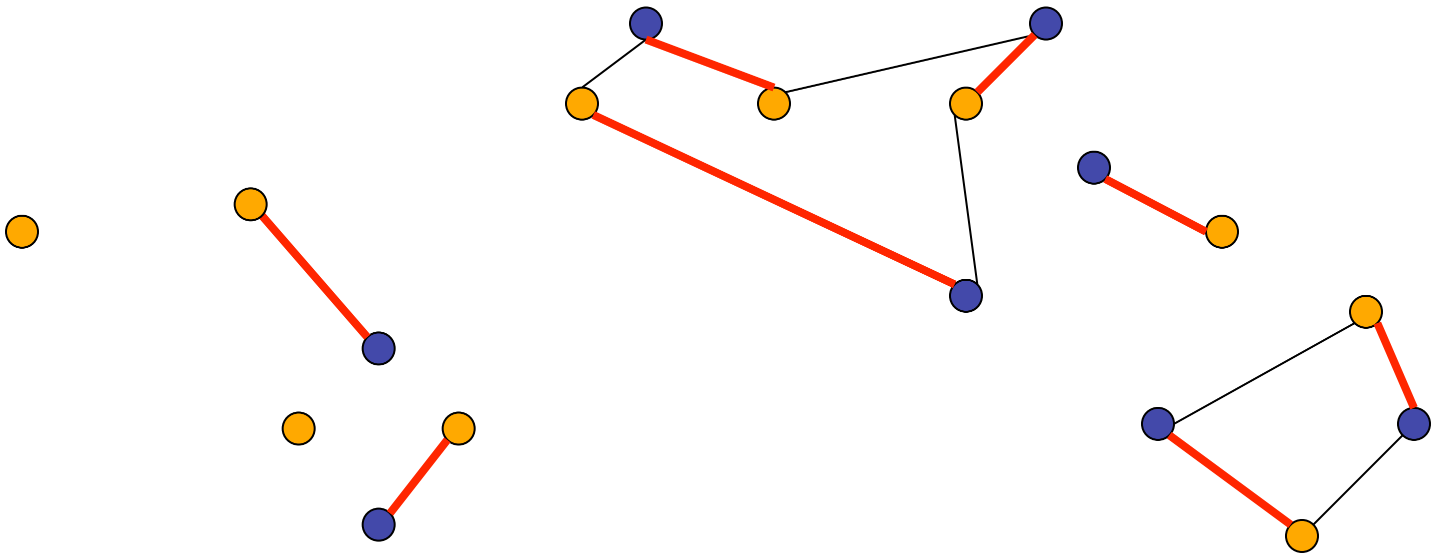
Machines and Jobs and Matching



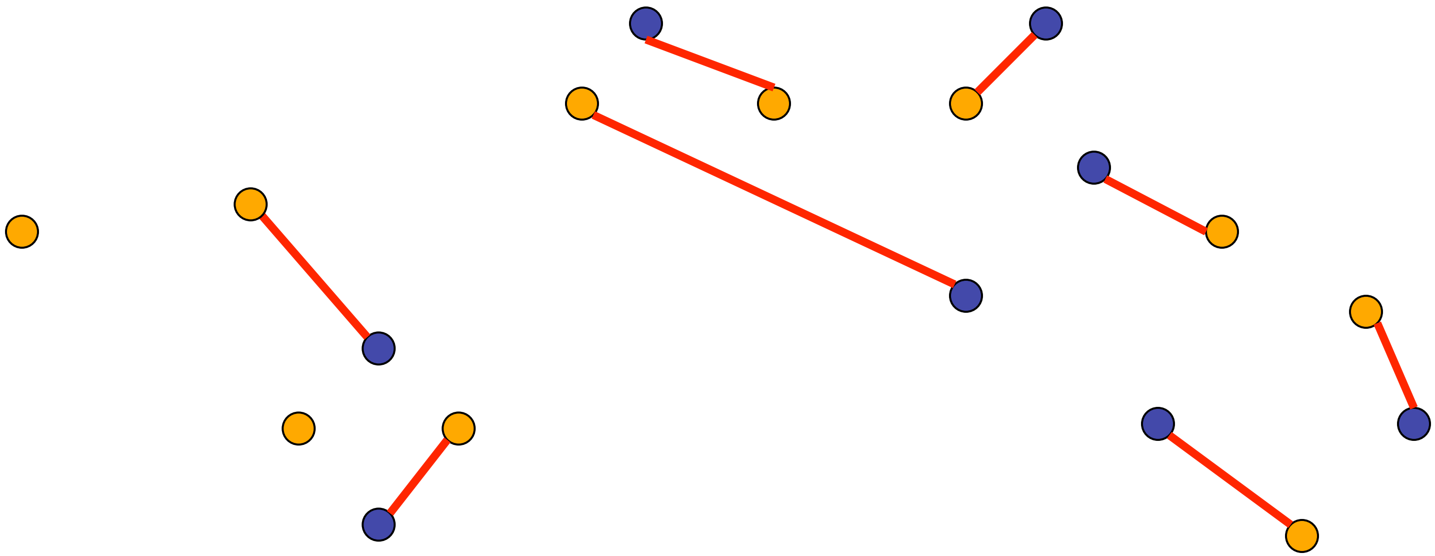
Proof of Lemma 9.4

- Each job that is integrally set in $\boldsymbol{x}$ has exactly one edge incident at it in G. Remove these jobs, together with their incident edges, from G. Clearly the remaining graph is H.
- Since an equal number of edges and vertices were removed, H is also pseudo-forest.
- In H, each job has a degree of at least 2.
- So, all leaves in H must be machines.
- Keep matching a leaf with the job it is incident to, and remove them both from the graph.
- In the end we will be left with even cycles.
- Match off alternate edges of each cycle.
- This gives a perfect matching in H.

Machines and Jobs and Matching



Machines and Jobs and Matching



Algorithm LST

- **Theorem 9.5**

  Algorithm LST achieves an approximation guarantee of factor 2 for the $R||C_{\max}$ problem.

Algorithm LST (Lenstra, Shmoys, Tardosh 1990)

Input ($J=\{1,\ldots, n\}$, $\boldsymbol{M}=\{M_1,\ldots, M_m\}$, $p: J\times M \to \mathbf{Q}^+$)

1. By a binary search in the interval $[\alpha/m,\alpha]$, find the smallest value of $T\in\mathbf{Z}^+$ for which LP(T) has a feasible solution. Let this value be T^*.
2. Find an extreme point solution, say $\boldsymbol{x}$, to LP(T^*) .
3. Assign all integrally set jobs to machines as in $\boldsymbol{x}$.
4. Construct graph H and find a perfect matching μ in it.
5. Assign fractionally set jobs to machines according to matching. Let σ be the obtained schedule.

Output (σ)

Proof of Theorem 9.5

- Clearly, $T^* \leq OPT$, since LP(OPT) has a feasible solution. The extreme point solution, $\boldsymbol{x}$, to LP(T^*) has a fractional makespan of $\leq T^*$.
- Therefore, the restriction of $\boldsymbol{x}$ to integrally set of jobs has an integral makespan of $\leq T^*$.
- The perfect matching found in H schedules at most one extra job on each machine.
- Hence, the total makespan is $\leq 2T^* \leq 2OPT$.
- The algorithm clearly runs in polynomial time.

Tight examples

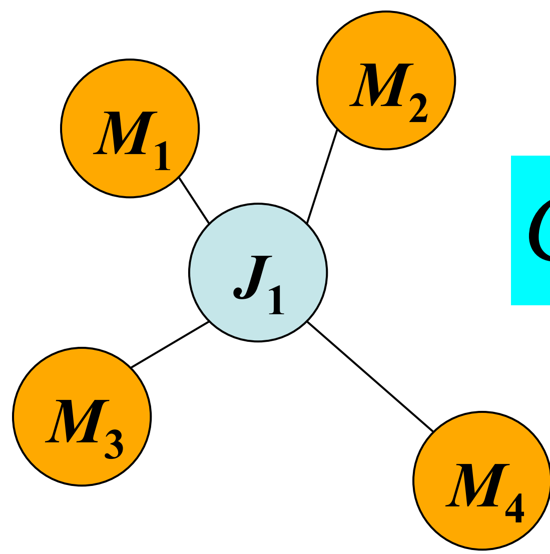
- Let us provide a family of tight examples. The m-th instance consists of $m^2 - m + 1$ jobs that need to be scheduled on m machines. The first job has a processing time of m on all machines, and all the remaining jobs have unit processing time on each machine.
- The optimal schedule assigns the first job to one machine, and m of the remaining jobs to each of the remaining $m - 1$ machines. Its makespan is m.
- Suppose the following extreme point solution to LP(m) is picked. It assign $1/m$ of the first job and $m - 1$ other jobs to each of the m machines. Rounding will produce a schedule having a makespan of $2m - 1$.

The Instance with 4 machines

M_1	J_1			
M_2	J_2	J_5	J_8	J_{11}
M_3	J_3	J_6	J_9	J_{12}
M_4	J_4	J_7	J_{10}	J_{13}

$$C^*_{\max} = m$$

M_1	J_1	J_2	J_3	J_4
M_2	J_1	J_5	J_8	J_{11}
M_3	J_1	J_6	J_9	J_{12}
M_4	J_1	J_7	J_{10}	J_{13}



$$C^{ЛП}_{\max} = 2m - 1$$

Exercises

- Does Algorithm LST achieve a better factor than 2 for the special case that the machines are identical?
- Prove that $\boldsymbol{x}_C$ must be an extreme point solution to $\mathrm{LP}_C(T)$.
- Prove that the solution given to LP(m) in the Example is an extreme point solution.