Tutorial #4

Optimization Methods

Integer Linear Programming

An integer linear program (abbreviated ILP) is a linear program (abbreviated LP) with the additional constraints that the variables must take integer values. For example, the following is an ILP:

$$
\begin{aligned}
\text{maximize} \quad & x_1 - x_2 + 2x_3 \\
\text{subject to} \quad & x_1 - x_2 \leq 1 \\
& x_2 + x_3 \leq 2 \\
& x_1 \in \mathbb{N} \\
& x_2 \in \mathbb{N} \\
& x_3 \in \mathbb{N}
\end{aligned}
$$

Integer Linear Programming

Very expressive language to formulate combinatorial optimization problems.

- **They can capture in a natural and direct way a large number of combinatorial optimization problems.**
- **Finding optimal solutions for ILPs is NP-hard.**

makespan: Scheduling on Unrelated Parallel Machines

We have machines corresponding to the set M. There are jobs which are supposed to run on these machines from the set J.

The time taken for each job j to complete on machine m is given to us as p_{mj}.

Individual jobs can't be broken down.

Our objective is to formulate a program that'll let us do this in the minimum time possible, which is called **makespan**.

How to formulate it as an IP optimization problem?

Let's have a binary variable x_{mj}, denoting whether machine m runs job j or not. Since we only need to run the job once, we'll program the constraint as below.

$$\sum_{m=1}^{|M|} x_{mj} = 1 \qquad \text{for } j = 1, \ldots, |J|$$

$$x_{mj} \in \{0, 1\}$$

Translating objective: English to LP

$$\text{minimize} \quad \max_m \left\{ \sum_j p_{jm} \cdot x_{mj} \right\}$$

How do you handle the $\max$?

Translating objective: English to LP

$$\text{minimize} \quad \max_{m} \left\{ \sum_{j} p_{jm} \cdot x_{mj} \right\}$$

How do you handle the $\max$?

Introduce a new variable, of course. Let t be the optimal makespan in the above formulation.

Translating objective: English to LP

The overall integer program is straightforward.

$$\begin{aligned}
&\text{minimize} && t \\
&\text{subject to} && \sum_j p_{mj} \cdot x_{mj} \leq t && \text{for } m = 1, \ldots, |M| \\
& && \sum_{m=1}^{|M|} x_{mj} = 1 && \text{for } j = 1, \ldots, |J| \\
& && x_{mj} \in \{0, 1\}
\end{aligned}$$

Analysis

How many decision variables?

Search space size?

Analysis

How many decision variables?

- $|J| \times |M|$

Search space size?

- $2^{|J| \times |M|}$, with each taking {0, 1}.
- Prune with the constraint - any improvements?

Approximate Algorithms

- We're happy if we get a solution which comes with certain guarantees w.r.t to the optimal solution, but can be solved in polynomial time.
- LP Relaxations and combinatorial optimization problems are a natural marriage in such settings.

Linear Programming Relaxation

1. Relax integer constraints to convex linear constraints.
2. Get a solution X using Linear Programming in polynomial time. This is as optimal as it gets.
3. Use some techniques (rounding and similar) to find a integer solution close to the best.
    - Let X^* be the optimal integral solution.
    - Let X' be what's obtained from rounding X.
    - Analyze how X' compares with X^*.

makespan: LP Relaxation

$$\text{minimize} \qquad t$$

$$\text{subject to} \qquad \sum_j p_{mj} \cdot x_{mj} \leq t \qquad \text{for } m = 1, \ldots, |M|$$

$$\sum_{m=1}^{|M|} x_{mj} = 1 \qquad \text{for } j = 1, \ldots, |J|$$

$$0 \leq x_{mj} \leq 1$$

A bunch of constraints are redundant. Which all?

makespan: LP Relaxation

$$\begin{aligned}
\text{minimize} \quad & t \\
\text{subject to} \quad & \sum_j p_{mj} \cdot x_{mj} \leq t && \text{for } m = 1, \ldots, |M| \\
& \sum_{m=1}^{|M|} x_{mj} = 1 && \text{for } j = 1, \ldots, |J| \\
& x_{mj} \geq 0 && \forall m, j
\end{aligned}$$

If we relax the constraints $x_{ij} \in \{0, 1\}$, it turns out that this formulation has unbounded integrality gap.

Example?

makespan: Can we do better?

We can make constraints stronger, if we know a threshold value t

$$\begin{aligned}
\text{minimize} \quad & t \\
\text{subject to} \quad & \sum_j p_{mj} \cdot x_{mj} \leq t && \text{for } m = 1, \ldots, |M| \\
& \sum_{m=1}^{|M|} x_{mj} = 1 && \text{for } j = 1, \ldots, |J| \\
& x_{mj} = 0 && \text{for } p_{mj} > t \\
& x_{mj} \geq 0
\end{aligned}$$

$$x_{mj} = 0 \qquad \text{for } p_{mj} > t$$

But can we formulate this statement as a Linear constraint?
If not how can we handle this?

$$
\begin{aligned}
\text{minimize} \quad & 0 \\
\text{subject to} \quad & \sum_{j} p_{mj} \cdot x_{mj} \leq T && \text{for } m = 1, \ldots, |M| \\
& \sum_{m=1}^{|M|} x_{mj} = 1 && \text{for } j = 1, \ldots, |J| \\
& x_{mj} = 0 && \text{for } p_{mj} > T \\
& x_{mj} \geq 0
\end{aligned}
$$

For a given T ,we can solve LP to check if it has a feasible solution. Now to find the minimum T we can perform a Binary Search. This minimum T would be the LPR optimal solution T^* .

Rounding

- We have $|J|$ jobs and $|M|$ machines.
- Solving the LP relaxed formulation gives us $x_{mj} \in [0, 1]$.
    - If $x_{mj} = 1$ or $x_{mj} = 0$, then we can assign the job j to machine m. We denote these as **integrally set**.
    - For $x_{mj} \in (0, 1)$, we denote them as **fractionally set**.

- A graph is induced by the subset of edges and its endpoints corresponding to fractionally set variables.
    - Let this graph be $H = (M' \cup J', E')$.
    - It's obvious that this graph is bipartite.
    - Finding an assignment reduces to finding a matching.

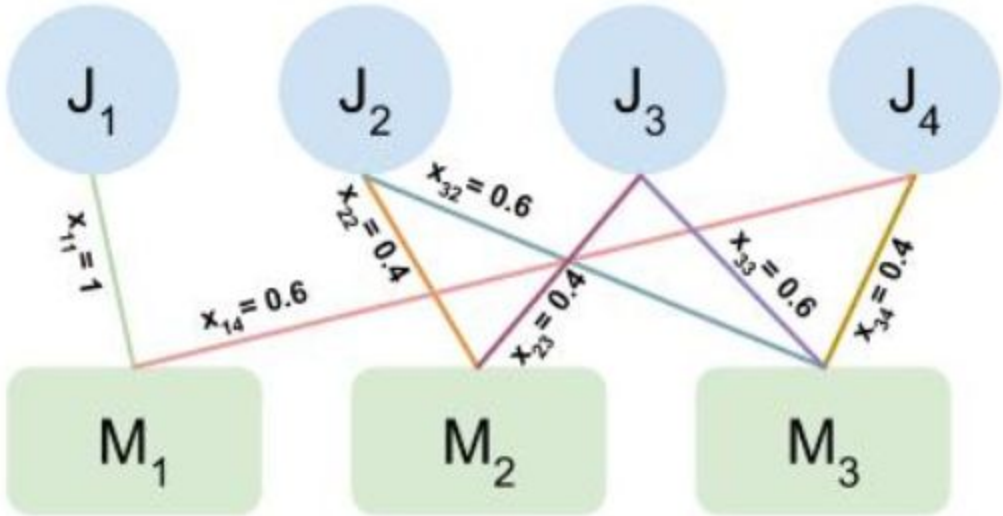
Finding an Assignment

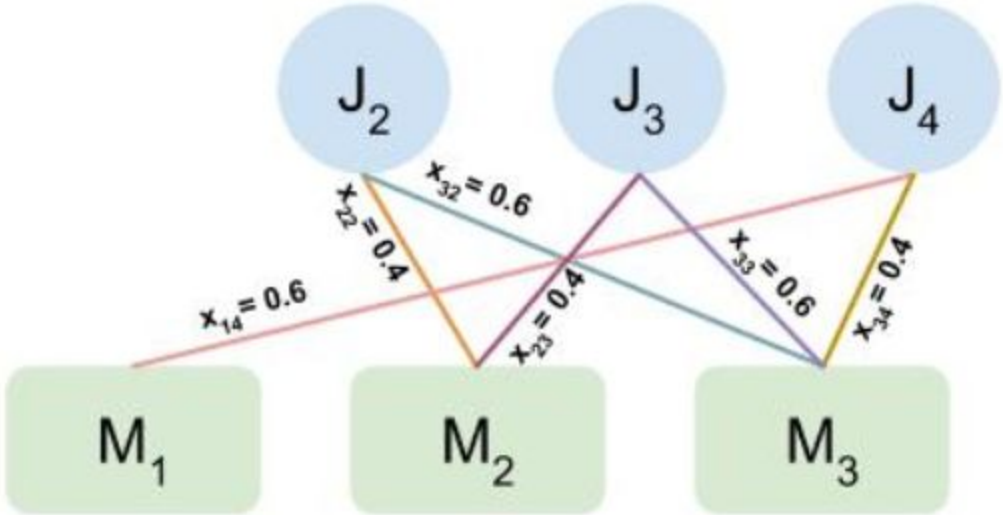
- In the graph H all the leaves would be machines only because each job would be having at least a degree of two (since it is fractional set job).
- Match a leaf with the job it is incident to and remove both of them from the graph.
- In the end we will be left with even cycles. Since the graph is bipartite all the cycles contains equal number of machines and jobs. Hence we can find an assignment by matching alternating edges of each cycle.

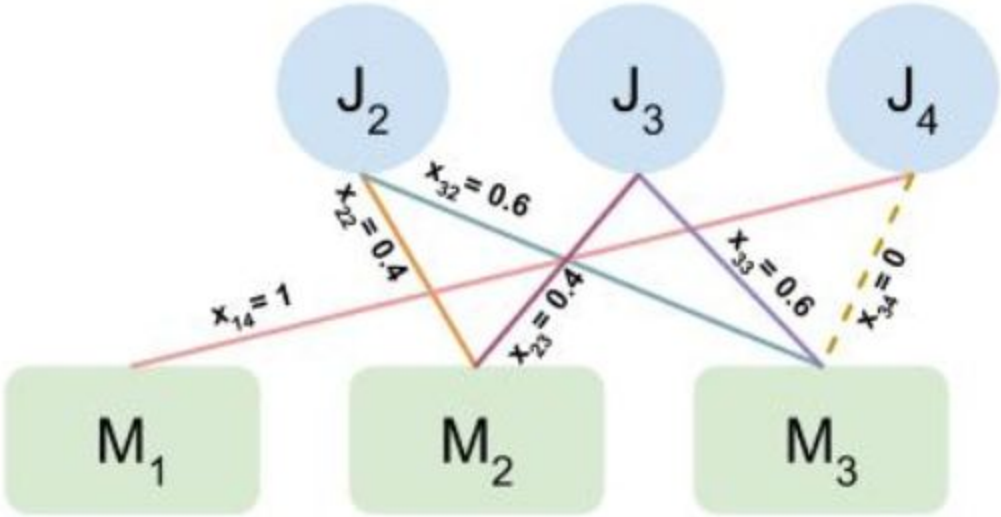
Prove:

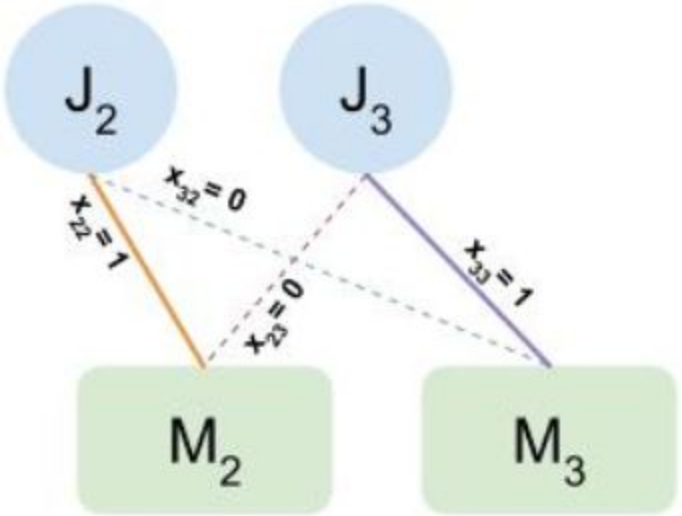
LP relaxation used here is 2-approximation for minimum makespan scheduling on unrelated machines.

Solve:
Round the following LPR solution to a IP solution




J2
J3
J4
x32 = 0.6
x22 = 0.4
x14 = 0.6
x23 = 0.4
x33 = 0.6
x34 = 0.4
M1
M2
M3


J2
J3
J4
x32 = 0.6
x22 = 0.4
x33 = 0.6
x34 = 0
x14 = 1
x23 = 0.4
M1
M2
M3


J2
J3
x32 = 0
x22 = 1
x23 = 0
x33 = 1
M2
M3