CSE 632: Analysis of Algorithms II **Fall 2017**

Lecture 9 (09/27/2017): Linear Programming

Lecturer: Shi Li *Scribe: Yuze Liu*

9.1 Make Span minimization on unrelated machines

Problem Description as follows :
Given : $[n]$ of jobs
$[m]$ of machines
$P_{i,j} \in \mathbb{Z} > 0$, $\forall i \in [m], \forall j \in [n]$
$P_{i,j}$ indicatng processing time of job j on machine i
Goal : find a partition of $[n]$ into $J_1, J_2, \cdots, J_m$, so as to minimize $(\max_i \sum_{j \in J_i} P_{i,j})$

First we have an Integer Programming (IP) for this problem:

$\min T$, s.t
$\sum_{i\in[m]} x_{i,j} \le T$, $\forall j \in [n]$. (Every job should be assigend)
$\sum_{j\in[n]} x_{i,j} P_{i,j} \le T$, $\forall i \in [m], j \in [n]$
$x_{i,j} \in \{0,1\}$, $\forall i \in [m], j \in [n]$. ($x_{i,j} \in \{0,1\}$: whether job j is assigned to i or not.)

To change the above Integer Programming (IP) to Linear Programming(LP), we can make follow changes : change $x_{i,j} \in \{0,1\}$ to $x_{i,j} \ge 0$

To solve the LP programming, we first guess an opt solution :
opt : guessed optimum makespan
require :

$$\begin{cases} x_{i,j} = 0, \text{ if } P_{i,j} > opt \text{ (if } P_{i,j} > opt\text{, not assign job j to machine i)} \\ T \le opt \end{cases}$$

Note : if for LP we don't have a solution , then for IP, we won't have a solution.
And then the LP becomes to :
Assume :
we have $\{x_{i,j}\}$
s.t $\sum_{i\in[m]} x_{i,j} = 1$, $\forall j \in [n]$
$\sum_{j\in[n]} x_{i,j} \le opt$, $\forall i$
$x_{i,j} = 0$, if $P_{i,j} > opt$
$x_{i,j} \ge 0$, $\forall i, j$
Goal :
find a solution with makespan $2 \cdot opt$

Before we introduce the algorithm, we first introduce rotation operation, here is a simple example of rotation operation:

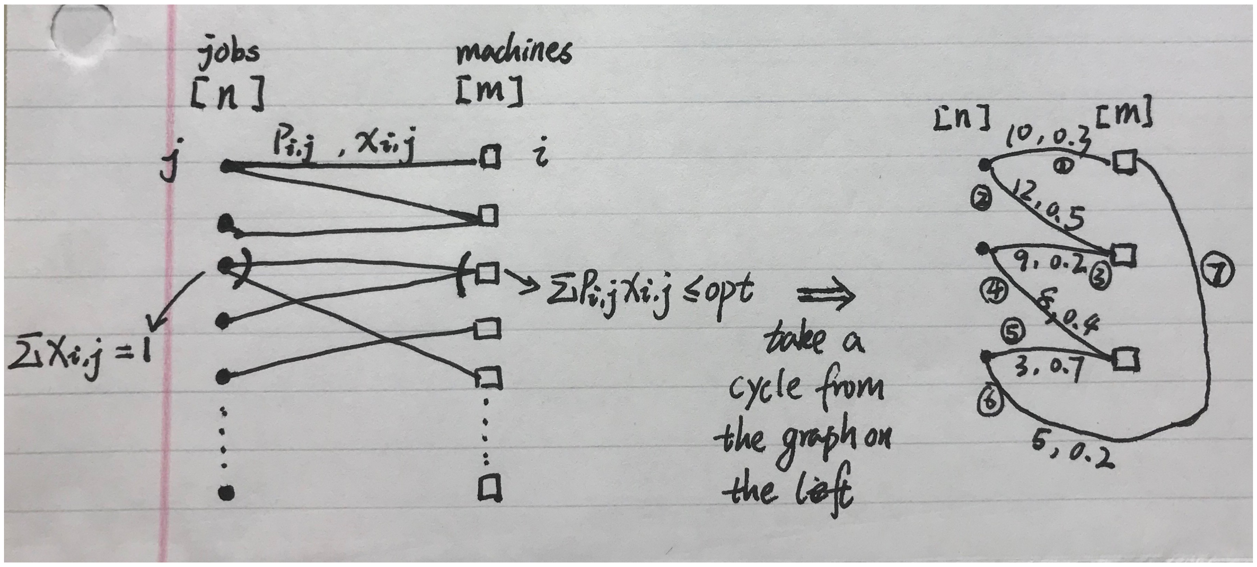

Figure 9.1: Figure of example of rotation operation, The figure on the left describe the LP problem in bipartite graph, left column are jobs, right column are machines. The figure on the right is a cycle we take from the original graph.

Figure 9.1 describe the problem in bipartite graph, we have $[n]$ jobs and $[m]$ machines, each edge between a job j and a machine m indicate to assign this job j to machine i. $P_{i,j}$ is the processing time for job i on machine j and in LP, $x_{i,j} \in [0, 1]$ indicates the probability that we assign job j to machine i. We can see, for each node job j, we have $\sum x_{i,j} = 1$ means each job j should be assigned to a machine finally, and for each node machine i, we have $\sum P_{i,j} x_{i,j} \le opt$ to satisfy our assumption that we have a guessed optimum makespan.

Then we take a cycle from the bipartite graph like what we have on the right part. There are 3 jobs and 3 machines which form a cycle, note that because this is only a part of the bipartite graph, so $\sum x_{i,j} \le 1$.

What we are going to do is start an edge, ie : edge 1, each time we change the $x_{i,j}$ a little bit, in the mean time we maintain the two properties of the graph : the sum of $x_{i,j} of$ a job j stays the same, and the $\sum P_{i,j} x_{i,j} \le opt$ for a machine i.

For example we reduce ϵ from edge 1, then to maintain property $1(0.3 \to 0.3 - \epsilon)$, we need to add ϵ to edge 2 $(0.5 \to 0.5 + \epsilon)$, in this way, the total sum of node 1 is still 0.8 after we make this change. Then comes to edge 3, for each machine, we maintain property 2 , so we have $12 \cdot \epsilon + 9 \cdot change = 0$, then for edge 3, the change should be $-\frac{4}{3} \cdot \epsilon$. we can follow this prcedure for all the edges, make changes and keep the above property for each job and machine. Then it coms to the last edge, edge 7, we. find that to keep the property of job 3, we need to add $+\frac{32}{9}\epsilon$, since it is a circle, for mahcine 1, the work load on it becomes $-\epsilon \cdot 10 + \frac{32}{9}\epsilon \cdot 5 = \frac{70}{9}\epsilon$, which means the workload on this machine increased. To change that, we can simply change the sign of ϵ (can also be seen as instead of decrease ϵ from edge 1, we can actually add ϵ on edge 1.

Then we need to figure out, how to choose this ϵ, there are two things we need to keep in mind: 1. We should choose ϵ as large as possible. 2. After decreasing, every edge should still have ≥ 0 property. Then from our example, we need to choose ϵ that $\epsilon = \min\{0.5, \frac{0.4}{4/3}, \frac{0.2}{32/9}\} = \frac{0.2}{32/9} = \frac{9}{160}$.

Similar to the above example, we can define :

C: a simple cycle of edges, s.t. $\forall (i,j) \in C$, $x_{i,j} \in (0.1)$
rotate(c) :
  color C in white and black.
  B : set of black edges in C.
  W: set of white edges in C.
  w.l.o.g. $\prod_{(i,j)\in W} P_{i,j} \leq \prod_{(i,j)\in B} P_{i,j}$

Explain :
Suppose we have machines : $i_1, i_2, i_3, \cdots, i_t, i_{t+1} = i_1$, and we have jobs : $j_1, j_2, j_3, \cdots, j_t$ which form a cycle like the above example, then each $i_m j_m, m \in [1,t]$ has an edge in W, each $i_m j_{m-1}$ has an edge in B.

Then we rotate C like we did in the example:

increase	x_{i_1,j_1}	by	ϵ
decrease	x_{i_2,j_1}	by	ϵ
increase	x_{i_2,j_2}	by	$\frac{P_{i_2,j_1}}{P_{i_2,j_2}}\epsilon$
decrease	x_{i_3,j_2}	by	$\frac{P_{i_2,j_1}}{P_{i_2,j_2}}\epsilon$
increase	x_{i_3,j_3}	by	$\frac{P_{i_2,j_1}P_{i_3,j_2}}{P_{i_2,j_2}P_{i_3,j_3}}\epsilon$
$\vdots$			
decrease	x_{i_{t+1},j_t}	by	$\frac{P_{i_2,j_1}P_{i_3,j_2}P_{i_4,j_3}\cdots P_{i_t,j_{t-1}}}{P_{i_2,j_2}P_{i_3,j_3}P_{i_4,j_4}\cdots Pi_t, j_t}\epsilon$

The total increase of load on i_1 will be $P_{i_1,j_1} \cdot \epsilon - \frac{P_{i_2,j_1}P_{i_3,j_2}P_{i_4,j_3}\cdots P_{i_t,j_{t-1}}}{P_{i_2,j_2}P_{i_3,j_3}P_{i_4,j_4}\cdots Pi_t,j_t} \cdot \epsilon = P_{i_1,j_1} \cdot \epsilon(1 - \frac{\prod_{(i,j)\in B} P_{i,j}}{\prod_{(i,j)\in W} P_{i,j}}) \leq 0$.

After this rotate operation, the graph will become a forest.

Algorithm and Analysis :

Algorithm 1 2-approximation of Make Span Minimization Probelm

```
1:  while there exists a cycle $C$, s.t. $\forall (i,j) \in C$, $x_{i,j} \in (0,1)$ do
2:      apply rotation(C)
3:  $\{(i,j) : x_{i,j} \in (0,1)\}$ forms a forest, each job $j$ has degree either 0, or at least 2 in the forest.
4:  if $x_{i,j} = 1$ then
5:      assign $j$ to $i$
6:  for each tree in the forest do
7:      let an arbitrary job be the root
8:      for every job $j$ do
9:          assign it to one of its child machines.
10:         Change $x_{i,j} \to 1$, for that child machine $i$, and $x_{i,j} \to 0$ for all the other machines.
11:         This is called force assign $j$ on $i$.
12: after force-asign, each machine $i$ if force-assigned at most 1 job $j^*$
```

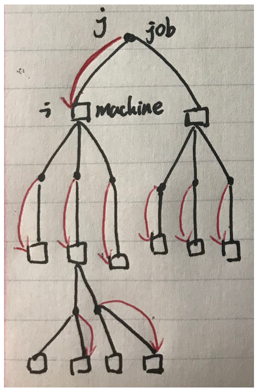

Figure 9.2: Figure of example of a tree from the Forest. Each node is a job and each square is a machine, red line indicates that we assign the job to that machine. We can see that each job j has degree either 0 or at least 2

$\sum_j P_{i,j}x_{i,j} \le opt + (1 - x_{i,j^*})P_{i,j*} \le opt + P_{i,j^*} \le 2 \cdot opt$, thus we get a 2-approximation.

Explain : Before the force-assign, after the rotation operation, the total load of all machines are less or equal to the *opt*, after the force-assign, we only consider the load that increased because of the force assign to the machine, and the worst case is x_{i,j^*} is very small and we have a 2-approximation algorithm.