Scheduling Theory **CO 454, Spring 2009**

Lecture 15,16,17: Factor 2 approximation for $(R||C_{max})$

June 30th, July 2nd, 7th, 2009

In the three lectures, we looked at a factor 2 approximation for minimizing makespan in unrelated machines. We used linear programming techniques to get the algorithm, and I suggest looking back into your CO350 notes for a quick brush-up on LP theory. The theorem that we will need is the following.

Theorem 0.1. *Given a polytope $\{Ax \geq b, x \geq 0\}$ where A has m rows (constraints) and n columns (variables), and let x be a basic feasible solution in the polytope, then x has at most m non-zero entries. This theorem makes sense only if the number of constraints is smaller than the number of variables.*

1 IP formulation

We start with an integer programming (IP) formulation for the problem. For every machine i and every job j, we have variable $x_{ij} \in \{0,1\}$ with the following semantic: $x_{ij} = 1$ iff job j is processed on machine i, in the optimum schedule. Note that given such x_{ij}'s, the total load on machine i is $\sum_{j=1}^{n} p_{ij}x_{ij}$. We also have a variable C to denote the makespan. The IP is as follows:

$$\begin{aligned} & \text{Minimize} \quad C \qquad (1) \\ \forall i: \quad & \sum_{j=1}^{n} p_{ij}x_{ij} \leq C \\ \forall j: \quad & \sum_{i=1}^{m} x_{ij} = 1 \\ \forall i,j: \quad & x_{ij} \in \{0,1\} \end{aligned}$$

The objective says we want to minimize makespan. The first constraint says that the makespan is at least as large as the total processing time on any machine. The second constraint says that every job must be processed in some machine. The last constraint says that x_{ij} is either 0 or 1.

Theorem 1.1. *The value of the IP equals the optimum makespan.*

Proof. Given a schedule S, we can construct a solution to the IP of value C^S_{max}. This shows $val(IP) \leq OPT$. To see this, set $C = C^S_{max}$ and $x_{ij} = 1$ if i processes j and 0 otherwise.

Conversely, given a solution to the IP of value C, one can construct a schedule with makespan C implying $OPT \leq val(IP)$. To see this, process job j on machine i iff $x_{ij} = 1$. The second constraint implies that exactly one of the x_{ij}'s corresponding to a job is 1 and thus each job is processed on exactly one machine. □

2 LP relaxations and Integrality gaps

As is expected, it is NP-hard to solve IPs. However, if we replace the constraint $x_{ij} \in \{0,1\}$ by $x_{ij} \geq 0$ in (1), we get a linear program, LP, which we can solve in polynomial time. In fact, we can find an optimal basic feasible solution in polynomial time. This LP is called the LP-relaxation of the IP.

The scheme to get an approximation algorithm is as follows. We construct the IP given the scheduling instance. From the IP we get the LP relaxation, LP, which we solve in polynomial time. Note that

$$val(LP) \leq val(IP) = OPT$$

The hope is if one can construct a schedule S with makespan C_{max} such that $C_{max} \leq \alpha \cdot val(LP)$, then one gets an α-factor approximation algorithm. This is because of the following string of inequalities

$$C_{max} \leq \alpha \cdot val(LP) \leq \alpha \cdot val(IP) = \alpha \cdot OPT$$

There is an inherent bottleneck to this approach. Given instance I of a scheduling problem, let $OPT(I)$ denote the optimum makespan for that instance, and $IP(I)$ and $LP(I)$ be the corresponding IP and LP relaxation for that instance. We claim that if $\frac{OPT(I)}{LP(I)} > \alpha$, then there is no α-approximation possible by the above method. This is because no matter what schedule we return, C_{max} of our schedule will have $C_{max} \geq OPT(I)$. If we further have $C_{max} \leq \alpha \cdot val(LP(I))$, then we will get $\frac{OPT(I)}{LP(I)} \leq \alpha$ countering our premise. Thus, the above discussion shows that the best possible factor one can hope using the LP relaxation for instance I is at most $\frac{OPT(I)}{LP(I)}$. Since our approximation algorithm must satisfy the approximation ratio for *all* instances, the best possible factor using this LP relaxation is

$$\max_{\text{instances } I} \frac{OPT(I)}{LP(I)}$$

This supremum is called the *integrality gap* of the LP-relaxation; and one cannot get an approximation factor better than the integrality gap using just the LP relaxation as a lower bound on the optimum.

How bad can the integrality gap be for the LP relaxation of (1)? We now show an example of an instance I such that the fraction $\frac{OPT(I)}{LP(I)} \geq m$, the number of machines. This will show that the integrality gap is at least m, and therefore one cannot get a constant factor approximation algorithm using this LP relaxation.

Example 2.1. *Consider a single job j and m machines $1, 2, \cdots, m$. The job j takes time $p_{ij} = m$ on each machine. Thats the instance I. Note that $OPT(I) = m$ since the job must be processed on some machine. However $LP(I) \leq 1$, since we show a feasible fractional solution of value 1. This is obtained by setting $x_{ij} = \frac{1}{m}$ for every $i = 1$ to m.*

3 A sequence of LPs and a factor 2 approximation

The above example shows what is wrong with the LP relaxation. The value of the LP, which is supposed to be the makespan, was 1, although the job took much longer time on any of the machines. In particular, we haven't used the following lower bound on the makespan

If job j is processed on machine i by the optimum solution, then we have $OPT \geq p_{ij}$.

However, we do not a priori know which machine processes which job (in fact that is what we want to find out), so the above seems to be a difficult constraint to encode for the IP. The way that this is taken care of is the following. Suppose we actually knew the value of the optimum makespan. Say it was C. Then, we know that jobs j which on machine i have processing time $p_{ij} > C$, will not be processed on machine i. For these (i, j) pairs, we can set $x_{ij} = 0$, explicitly. Maybe, then, the LP relaxation will not have a bad integrality gap.

Nevertheless, this seems to be weird since we do not know the optimum makespan value. The main insight is the following: we can still "guess" it. We know that the optimal makespan is at least 1 (assuming integral processing times) and at most np_{max}, where p_{max} is the maximum processing time of any job on any machine. Thus, if we had a procedure which given a guess on the optimum, say C, either returns a feasible schedule of makespan at most $\alpha \cdot C$, or it *proves* that the optimum makespan is larger than C, then we could perform a binary search to obtain an α-factor approximation. We now show that a modified LP as the LP relaxation of (1) suffices as a procedure for $\alpha = 2$.We now give the details.

Given a parameter C, let $S_C := \{(i, j) : p_{ij} \leq C\}$. Define the following linear polytope

$$\begin{aligned} LP_C := \{\forall i : \quad & \sum_{j=1}^{n} p_{ij} x_{ij} \leq C \\ \forall j : \quad & \sum_{i:(i,j)\in S_C} x_{ij} = 1 \\ \forall i, j : \quad & x_{ij} \geq 0\} \end{aligned}$$

Firstly note that for $C = OPT$, LP_C is feasible. This is similar to the reason why the value of the above IP was equal to the optimum. Therefore, if for some C, we get that LP_C is infeasible, then $C < OPT$, and we know that our guess, C, is too low. We will prove the following theorem later.

Theorem 3.1. *If LP_C is feasible for any C, then one can find a schedule with makespan at most $2C$ in polynomial time.*

Now we can state the algorithm.

1. Initialize $L = 1$ and $U = np_{max}$.
2. While $U - L > 1$, do:
   - Let $C := \frac{(L+U)}{2}$
   - If LP_C is feasible, then $U = C$.
   - If LP_C is not feasible, then $L = C$.
3. Let C^* be the final C. Note it is the minimum C such that LP_C is feasible. Now use the Theorem 3.1 to get a schedule of makespan at most $2C^*$.

Theorem 3.2. *The above algorithm runs in polynomial time and returns a schedule with makespan at most* $2OPT$

Proof. The algorithm runs in time $\log(np_{max}) \cdot T_1 + T_2$, where T_1 is the time taken to solve LPs and T_2 is the time taken to run the algorithm in Theorem 3.1, which is a polynomial. The proof completes by noting that $C^* \leq OPT$ since LP_{OPT} is feasible. □

4 Proof of Theorem 3.1

Let x^* be a basic feasible solution for LP_C. We now use x^* to construct the schedule with makespan at most $2C$. Let $E^* := \{(i,j) : x^*_{ij} > 0\}$. By LP theory, we have that $|E^*|$ is at most the number of constraints, which is $(n+m)$. Construct the following bipartite graph $G = (V, E^*)$ where there is a vertex for every machine i and job j, and there is an edge (i,j) if $(i,j) \in E^*$.

We first claim that we may assume that the graph $G(V, E^*)$ is connected. Suppose not. Suppose there are two components (V_1, E_1) and (V_2, E_2). But then, thinking of the vertices as jobs and machines again, we will have two separate machine scheduling problems for which LP_C is feasible. By "induction", we can solve them separately since they are smaller problems, and get a schedule for them separately, each of makespan at most $2C$. Thus, we may assume there is only one component in G. Since G is connected and has $|E^*| \leq |V|$, G is either a tree or a tree plus an edge. In particular, G can have at most one cycle.

Call a job j a *leaf job* if it has degree 1 in the graph G. Let machine i be a parent of leaf job j if the edge (i,j) is the only edge incident on j. Note that $x^*_{ij} = 1$ for a leaf job j and its parent machine i. This is because for every job j we have $\sum_{i:(i,j)\in E^*} x^*_{ij} = 1$. Let J_i be the leaf jobs connected to a machine i. We process all jobs in J_i on machine i. Note that for any i, we have

$$\sum_{j \in J_i} p_{ij} = \sum_{j \in J_i} p_{ij} x^*_{ij} \leq C$$

where the first inequality follows since j is a leaf job and the second from the LP_C constraint.

Delete all leaf jobs from G to get the graph G'. Now every job j in G' has degree at least 2. Now do the following procedure

- Let M be an empty set.
- While G' is a single cycle or empty do
    1. Find a machine i with degree 1. Let j be the machine it is connected to. Add (i,j) to M. Delete the vertex i and j from G'. If any machine i has degree 0, delete it as well.
- If G' is a cycle, find a matching M' in the cycle matching each job to a unique machine. Let $M = M \cup M'$.
- For each job j, assign it to the machine i with $(i,j) \in M$.

We claim that the above procedure either ends with an empty graph or a single cycle. Firstly note that if G' is neither a single cycle nor empty, then it must have vertices of degree 1. Since all jobs have degree at least 2, this much be a machine. Also, deleting this machine i, and deleting the unique job connected to it doesn't decrease the degree of any other jobs. So we can repeat this procedure, till we get a cycle or an empty graph. If a cycle, we find another matching M' to augment M. In the end, every job is paired to a unique machine.

Thus, every machine i has total processing $\sum_{j \in J_i} p_{ij} + p_{ij(i)}$ where $j(i)$ is the job assigned to machine i in the above procedure. if no job is assigned, let $p_{ij(i)} = 0$. The first term in the sum is at most C. The second term is also at most C since $(i, j(i)) \in S_C$). This proves the theorem.