Approximation Algorithms and Hardness of Approximation March 13, 2013

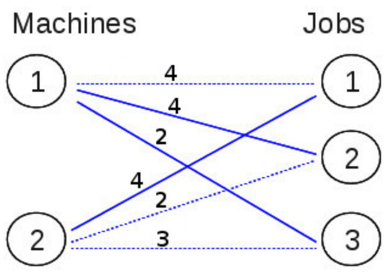
Lecture 7

Lecturer: Ola Svensson *Scribes: Alfonso Cevallos*

1 Scheduling on Unrelated Parallel Machines

This week we continue exploring some techniques to obtain an approximation guarantee to a problem, by formulating it exactly as an integer program (IP), then solving its relaxation to a linear program (LP), and finally rounding the LP fractional solution to obtain a feasible integer solution, hopefully without losing too much. Recall also that if the region defined by an LP is bounded, then we can find an optimal solution which is an extreme point. Thus we want to exploit any structure on the extreme points of our LP. For instance, last week we proved (and exploited the fact) that all the extreme points of the Vertex Cover LP are half-integral.

This week we will study the problem of Scheduling on Unrelated Parallel Machines (SUPM), and the approximation algorithm due to Lenstra, Shmoys, and Tardos (1990). Given n jobs, m machines, and a nonnegative integer function $p : [m] \times [n] \to \mathbb{Z}^+$, where $p_{i,j} = p(i,j)$ represents the time it takes for machine i to execute job j, the problem is to find a schedule that minimizes the makespan, i.e. assign a machine to each job, in a way that minimizes the maximum total processing time of any machine. In the next example, we represent the m machines and n jobs as the vertices of the complete bipartite graph $K_{m,n}$, and write the processing times as edge weights. In this instance, the optimal schedule is shown with dashed edges, and it has a makespan of value 5.



1.1 IP formulation, first attempt

We keep in mind the representation of the problem as the complete bipartite graph $k_{m,n}$ with weighted edges, and we formulate an IP where to each edge (i,j) corresponds an indicator variable $x_{i,j}$, taking the value 1 if job j is assigned to machine i, or 0 otherwise. We introduce a variable t which represents the makespan, to be minimized. Our IP looks like this:

$$\begin{aligned}
&\text{minimize: } t \\
&\text{subject to: } \sum_{j=1}^{n} x_{i,j} p_{i,j} \leq t && 1 \leq i \leq m && \text{(Load of each machine} \leq \text{makespan } t) \\
&\sum_{i=1}^{m} x_{i,j} = 1 && 1 \leq j \leq n && \text{(Each job is assigned to one machine)} \\
&x_{i,j} \in \{0,1\} && 1 \leq i \leq m,\ 1 \leq j \leq n && \text{(Integrality)}
\end{aligned}$$

And in the LP relaxation the last constraint becomes $0 \leq x_{i,j} \leq 1$, but since we are trying to minimize t, we can safely remove the upper-bounding constraint:

$$\begin{aligned} \text{minimize: } & t & \\ \text{subject to: } & \sum_{j=1}^{n} x_{i,j} p_{i,j} \leq t & \forall i \in [m] \\ & \sum_{i=1}^{m} x_{i,j} = 1 & \forall j \in [n] \\ & x_{i,j} \geq 0 & \forall i \in [m],\ j \in [n] \end{aligned}$$

However, after some inspection we discover that this approach is hopeless, because the integrality gap is really bad. Consider the instance where there are m machines and only one job, and each machine needs a time m to process the job. The optimal integral solution will assign the job arbitrarily to some machine, with a makespan of m, while the optimal solution of our LP relaxation is fractional, and splits the job in equal parts to all the machines, with a makespan of 1; so the integrality gap is at least m.

1.2 Parametric pruning

In order to get around this problem, we will need to use LPs differently. If we inspect again the instance given in the previous paragraph, we get the impression that it would be a good idea to "check for feasibility" before we accept an alleged optimal makespan t, by comparing it to the heaviest edges in the corresponding solution. More concretely, we should define a new LP that, for a fixed value of makespan t, "prunes off" the edges with value greater than t, and then tries to find a feasible schedule with the remaining edges (note: this LP doesn't optimize anything, it just tries to find a feasible point). This is exactly what we will do. Our "parametric pruning" LP looks like this:

$$\begin{aligned} \text{Input: } & T & \text{(Candidate makespan)} \\ \text{Define: } & S_T = \{(i,j) | p_{i,j} \leq T\} & \text{(Prune off the heaviest edges)} \\ \text{Find feasible schedule s.t.: } & \sum_{j:(i,j)\in S_T} x_{i,j} p_{i,j} \leq T & \forall i \in [m] \\ & \sum_{i:(i,j)\in S_T} x_{i,j} = 1 & \forall j \in [n] \\ & x_{i,j} \geq 0 & \forall i \in [m],\ j \in [n] \end{aligned}$$

With this LP, we can quickly find the minimum makespan value T for which there is a feasible, fractional solution, by running a binary search: $T^* = \min\{T : \text{LP}(T) \text{ is feasible}\}$. This value is clearly a lower bound on the optimal integer value ($T^* \leq \text{OPT}$), but, as it turns out from the analysis of the extreme points of LP(T^*), it is not too far from OPT. We will be able to design an algorithm that rounds an extreme point of LP(T^*), and gives a 2-approximation, or even a far better approximation in instances without particularly heavy edges. We skip the proof of the following theorem to the end of the next section.

Theorem 1 *There is a polytime algorithm for SUPM that finds a schedule with a makespan of value* $\leq T^* + \max_{(i,j)\in S_{T^*}} p_{i,j} \leq 2T^* \leq 2OPT$.

1.3 Extreme points

Remark: One of the several equivalent definitions of an extreme point is the following. In an LP with r variables, a feasible point X is an extreme point iff there are r linearly independent constraints of the LP that are set to equality in X.

Lemma 2 *Any extreme point of LP(T^*) has at most $n+m$ non-zero variables.*

Proof LP(T^*) has $r = |S_{T^*}| = O(mn)$ variables, and three kinds of constraints: n of the form $\sum_i x_{i,j} = 1$, m of the form $\sum_j x_{i,j} p_{i,j} \leq T$, and r of the form $x_{i,j} \geq 0$. Let $X = (x_{i,j})_{(i,j)\in S_{T^*}}$ be an extreme point of LP(T^*). By the previous remark, there are r linearly independent constraints that are set to equality by X, of which at most $n+m$ are of the first two kinds, and at least $r-n-m$ are of the third kind. Thus, at least $r-n-m$ coordinates of X are zero, which is what we wanted to prove. ■

Corollary 3 *Any extreme point of LP(T^*) has at most m fractionally assigned jobs.*

Proof Let X be an extreme point of LP(T^*), and let k be the size of its support, i.e. the number of non-zero variables, so by the previous lemma we know that $k \leq n+m$. Notice that k represents the number of job-machine assignations in X. Each job needs at least one such assignation, and it needs more than one assignation iff it is fractionally assigned. Therefore the number of fractionally assigned jobs in X is $\leq k-n \leq m$. ■

A graph G is called a pseudotree if it is connected and it contains at most one cycle. In other words, a pseudotree is either a tree, or a tree plus an extra edge. Also, a graph G is called a pseudoforest if each connected component of G is a pseudotree. Let X be an extreme point of LP(T^*), and let $G(X)$ be the subgraph of $K_{m,n}$ corresponding to the support of X, i.e. $G(X) = G([m]\times\ [n], \{(i,j) \in S_{T^*} | x_{i,j} > 0\})$.

Lemma 4 *For any extreme point X, $G(X)$ is a pseudoforrest.*

Proof Fix an extreme point X. The graph $G(X)$ is bipartite, it has $m+n$ vertices, and at most $m+n$ edges. We consider two cases:

If G is connected, then $G(X)$ necessarily corresponds to a spanning tree of $K_{m,n}$ with at most one extra edge, so it is a pseudotree.

If G is not connected, then we must prove that every connected component is a pseudotree. Assume, towards contradiction, that there is a connected component C which is not a pseudotree. Let $\bar{C}$ denote the rest of $G(X)$. Then C and $\bar{C}$ define two separate scheduling instances, and the corresponding restrictions of X in them, X_C and $X_{\bar{C}}$, are clearly feasible points of the corresponding restricted instances. But X_C cannot be an extreme point, because $C = G(X_C)$ is connected and is not a pseudotree, so it would violate case 1. Thus, X_C can be written as the convex combination of other feasible points: $X_C = \lambda Y_1 + (1-\lambda)Y_2$, with $Y_1 \neq Y_2$. But then X could also be written as the convex combination of other feasible points of the original instance: $X = \lambda(X_{\bar{C}} + Y_1) + (1-\lambda)(X_{\bar{C}} + Y_2)$, which contradicts the fact that X was an extreme point. Therefore G is a pseudoforrest. ■

Finally, we proof Theorem 1 by showing how to round an optimal fractional solution.

Proof Let X be an optimal extreme point of LP(T^*). We know that it gives a makespan of value $\leq T^* \leq$ OPT, and we will show that we can round the fractional job assignations by adding at most one job integrally to each machine, so the value of the makespan increases by at most $\max_{(i,j)\in S_{T^*}} p_{i,j}$, and the theorem follows. We do this by finding an injective function from the fractionally assigned jobs to the machines.

We consider the pseudoforrest $G(X)$. First, we keep all the integrally assigned jobs in our rounding, and remove them from $G(X)$ (they correspond to leaves). Now, the remaining graph is composed of machines and fractionally assigned jobs, where all the edges join a job with a machine, and all the leaves are machines. What remains to do is simply find a matching that covers all jobs. As long as there are leaves, pick a leaf at random, add the corresponding edge to the matching (i.e. to the rounding), and remove from the graph the corresponding job with all of its children leaves. Notice that the algorithm keeps the property that all the leaves are machines.

When there are no more leaves, what remains is either an empty graph, or an even cycle. In the latter case, we pick a perfect matching. So we can find a matching that covers all fractionally assigned jobs. This completes the proof. ■