CMPUT 675: Approximation Algorithms **winter 2018**

Lecture 12 (Feb 15 2018): Scheduling on unrelated parallel machines

Lecturer: Mohammad R. Salavatipour *Scribe: Maryam Mahboub*

12.1 Problem: Scheduling on unrelated parallel machines

Given a set J of n jobs, and set M of m machines, for each $j \in J$ and $i \in M$, $p_{ij} \in Z^+$ is the processing time of job j on machine i. Our goal is to schedule the jobs on the machines to minimize the "makespan" (i.e. the latest finish time of machines).

Special cases: Identical machines: p_j is the processing time for each job j on all machines: there is a $PTAS$ for identical machines; the idea is similar to the Bin Packing problem. Note that in this setting we have a fixed number of "bins" and we have to pack the items to minimize the bin size we use to fit all jobs whereas in bin packing we have a fixed bin size and have to use minimum number of bins.

Related machines: Each machine has a speed s_i, processing time of job j on machine i is $\frac{p_j}{s_i}$ time. There is a $PTAS$ for related machines too.

In this lecture we present a 2-approximation for the general setting using LP rounding.

12.1.1 LP for unrelated parallel machines:

Consider the following natural LP relaxation for this problem:

$$\begin{array}{rcll} \text{minimize} & t & & \\ \sum_{i=1}^{m} x_{ij} & = & 1 & \forall j \in J \\ \sum_{j \in J} p_{ij} x_{ij} & \leq & t & \forall i \in M \\ x_{ij} & \geq & 0 & \forall j \in J, \forall i \in M \end{array}$$

We show that this integer program has unbounded integrality gap.

Example 1 *Suppose we have only one job, which has a processing time of m on each of the m machines. Clearly, the minimum makespan is m. However, the optimal solution to the linear relaxation is to schedule the job to the extend of $\frac{1}{m}$ on each machine, thereby leading to an objective function value of 1, and giving an integrality gap of m.*

We will use the technique of parametric pruning to get around this difficulty. The parameter will be $T \in Z^+$, which is our guess for a lower bound of the optimal makespan. The parameter will enable us to prune 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 parameters $T \in Z^+$. 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 $\leq T$.

$$\begin{array}{rcll} \displaystyle\sum_{i:(i,j)\in S_T} x_{ij} & = & 1 & \forall j \in J \\ \displaystyle\sum_{j:(i,j)\in S_T} p_{ij}x_{ij} & \leq & T & \forall i \in M \\ x_{ij} & \geq & 0 & (i,j) \in S_T \end{array}$$

We find the smallest value for T which $LP(T)$ is feasible using binary search. Let's call it T^*.

NOTE: $T^* \leq OPT$ (because $LP(OPT)$ is also feasible). We will give an algorithm that find a solution of makespan $\leq 2T^*$.

Lemma 1 *Any basic feasible solution to $LP(T)$ has at most $n+m$ nonzero variables.*

Proof. We have total of $r = |S_T|$ many variables. In each basic feasible solution we have at least $r-(m+n)$ tight constraint of the third type, so at most $(n+m)$ of $x'_{ij}s$ are nonzero. ∎

Let x be a basic feasible solution to $LP(T)$. We will say that job j is integrally set in x if it is entirely assigned to one machine. Otherwise, we will say that job j is fractionally set.

Lets build bipartite graph $G = (J \cup M, E)$, such that $(i,j) \in E$ iff $x_{ij} \neq 0$. Let $F \subseteq J$ be the set of fractionally assigned jobs and $H \subseteq G$ be the sub-graph induced by $F \cup M$. We prove the following Lemma shortly:

Lemma 2 *Graph H has a perfect matching that assigns each job in F to one machine.*

Assuming this lemma for now, we describe the algorithm and do the analysis of its performance ratio.

12.1.2 The algorithm

The algorithm starts by computing the range in which it finds the right value for $T \in [0, \sum p_{ij}]$, because we know that the makespan would not be greater than the sum of all processing times.

Algorithm 1 Scheduling on unrelated parallel machines

1. Using binary search in $[0, \sum p_{ij}]$, find smallest T such that LP(T) is feasible, call it T^*.
2. Let x be a basic feasible solution to LP(T^*).
3. Assign all integrally assigned jobs in x.
4. Build graph H and find a matching M and assign fractional jobs based on M.

Definition 1 *A pseudo-tree is a connected graph with $|V|$ edges. A pseudo-forest is a graph where each component is a pseudo-tree.*

Lemma 3 *G is a pseudo-forest.*

Proof. We will show that the number of edges in each connected component of G is bounded by the number of vertices in it. Hence, each connected component is a pseudo-tree. Consider any connected component called C and focus on x restricted to jobs and machines in C. Note that x_c is a basic feasible solution to $LP(T)$ restricted to jobs and machines in C. By Lemma 1, the number of nonzero variables is exactly $|J_c| + |M_C| = |V_c|$. So component C has $|V_c|$ edges. So C is pseudo-tree and G is a pseudo-forest. ■

Now we prove Lemma 2.

Proof. [of Lemma 2:] We showed that H is a pseudo-forest (because G is a pseudo-forest). Each job has a degree of at least 2, because all jobs in F are fractionally assigned. So, all the leaves in H must be machines. If a machine i has degree 1, then pick that edge and add it to M. Keep matching a leaf with the job it is incident to, and remove both from the graph. After each step all the leaves must be machines, because we are not removing any edge incident to remaining jobs. At the end what is left is some even cycles each of them has an obvious perfect matching. This means that H has a perfect matching. ■

Theorem 1 *The algorithm for Scheduling on unrelated parallel machines is a 2-approximation for makespan.*

Proof. Clearly $T^* \leq OPT$, since $LP(OPT)$ has a feasible solution. By assigning integral jobs, the load on each machine is at most T^*. Since for each job $p_{ij} \leq T^*$, assigning fractional jobs based on matching M increases the load on each machine by at most T^*. So the solution has makespan at most $2T^*$. The algorithm clearly runs in polynomial time. ■

12.2 Generalized Assignment Problem

In the generalized assignment problem, we are given a collection of n jobs to be assigned to m machines. Each job $j \in \{1, ..., n\}$ has to be assigned to exactly one machine; if it is assigned to machine i, then it requires p_{ij} units of processing time, and incurs a cost of c_{ij}. Furthermore, we are given a time bound T that limits the total processing of each machine $i \in \{1, ..., m\}$. The aim is to find a feasible assignment of minimum total cost with makespan at most T.

If we write this problem as an integer program, and introduce a $0-1$ variable x_{ij}, for each $i \in \{1, ..., m\}$ and $j \in \{1, ..., n\}$ to indicate whether job j is assigned to machine i, then we obtain:

$$\begin{array}{rrcll}
Min & \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} & & & \\
& \sum_{i=1}^{m} x_{ij} & = & 1 & j = 1, ..., n \\
& \sum_{j=1}^{n} p_{ij}x_{ij} & \leq & T & i = 1, ..., m \\
& x_{ij} & \in & \{0,1\} & j = 1, ..., n, i = 1, ..., m
\end{array}$$

We will provide a bi-criteria algorithm which detects if the input is in-feasible or find a solution of cost $\leq OPT$ but with makespan at most, $T + P_{max} \leq 2T$.