CMPUT 675: Approximation Algorithms	Fall 2014

Lecture 28 (Nov 14): Minimizing Makespan on Unrelated Machines

Lecturer: Zachary Friggstad *Scribe: Antonio Carlos Salzvedel Furtado Junior*

28.1 Minimizing Makespan on Unrelated Machines

In the problem Minimizing Makespan on Unrelated Machines, we are given a set of jobs $J = \{1, ..., n\}$ and a set of machines $M = \{1, ..., m\}$. Each job $j \in J$ requires processing time $p_{ij} \geq 0$ to be run on machine $i \in M$. Our objective is to find an assignment function $\phi : J \rightarrow M$ that minimizes the makespan:

$$\max_{i \in M} \sum_{j:\phi(j)=i} p_{ij}$$

Unlike the case of identical machines from an earlier lecture, there is probably no PTAS for this problem.

Theorem 1 *For any $c < \frac{3}{2}$, there is no c-approximation for the* Minimizing the Makespan on Unrelated Machines*, unless* $\mathbf{P} = \mathbf{NP}$.

The proof is simple, but we skip it for the sake of time.

In this lecture, we saw an iterative rounding algorithm that provides a 2-approximation for this problem. Our presentation follows the approximation for the Generalized Assignment Problem that is recorded in [LRS11], but we focus on the makespan minimization version for simplicity.

Consider the following LP relaxation for the problem. Here γ represents the makespan and x_{ij} is a binary variable that indicates if job j is assigned to machine i.

$$\begin{array}{rrl} \text{minimize :} & \gamma & \\ \text{subject to :} & \sum_{i \in M} x_{ij} = 1 & \forall j \in J \\ & \sum_{j \in J} p_{ij} \cdot x_{ij} \leq \gamma & \forall i \in M \\ & x, \gamma \geq 0 & \end{array} \qquad \textbf{(LP-1)}$$

Unfortunately, (**LP-1**) has a bad integrality gap. For example, consider the instance with jobs $J = \{1\}$ and machines $M = \{1, ..., m\}$ such that $p_{i1} = m$ for each $i \in M$. A feasible solution to (**LP-1**) with value 1 is $\overline{x}_{i1} = 1/m$ for each $i \in M$ and $\gamma = 1$. However, the optimal solution $OPT = m$, since m is the processing time any machine will take to run the only job. Despite this bad gap, (**LP-1**) will still be useful in our approximation after we strengthen it a bit.

The 2-approximation follows the same basic approach as the PTAS for identical machines: "guess" the target makespan with a binary search.

Theorem 2 *There is a polynomial-time algorithm that, given a value T, either returns a solution with makespan at most $2T$ or else reports there is no solution. If $T \geq OPT$, it is guaranteed to find a solution with makespan at most $2T$.*

The main tool used to prove Theorem 2 is the strengthening (**Feasibility-LP**) of (**LP-1**). Note that it is not an LP in the strictest sense of the definition as there is no objective function; we are only interested in whether there is a feasible solution.

$$
\begin{aligned}
\sum_{i \in M} x_{ij} &= 1 & \forall j \in J \\
\sum_{j \in J} p_{ij} \cdot x_{ij} &\leq T & \forall i \in M \\
x_{ij} &= 0 & \text{if } p_{ij} > T \\
x &\geq 0
\end{aligned}
\qquad (\textbf{Feasibility-LP})
$$

In our bad example above, (**Feasibility-LP**) has no feasible solution if $T < m$.

The rest of this lecture is devoted to proving the following statement.

Theorem 3 *If (***Feasibility-LP***) has no feasible solution then $OPT > T$. If (***Feasibility-LP***) has a feasible solution, then we can find an integer solution with makespan $\leq 2T$ in polynomial time.*

Clearly when $T \geq OPT$ then the natural $\{0, 1\}$ solution corresponding to the optimum is a feasible LP solution. So, we focus on proving that if there is a feasible solution then we can find an assignment $\phi : J \to M$ with makespan at most $2T$.

In the course of the algorithm, we will be dropping some potential $j \to i$ assignments and we will also be removing some machines from consideration (after assigning them some jobs). So, it will be convenient to view the problem in a more general setting.

Let $G = (J \cup M, E)$ be a bipartite graph and, for each $i \in M$, let T_i be a bound on the target running time of machine i. Now consider the following more general feasibility LP.

$$
\begin{aligned}
\sum_{i:(i,j) \in E} x_{ij} &= 1 & \forall j \in J \\
\sum_{j:(i,j) \in E} p_{ij} \cdot x_{ij} &\leq T_i & \forall i \in M \\
x &\geq 0
\end{aligned}
\qquad (\textbf{Feasibility-LP2})
$$

Algorithm 1 is the rounding algorithm, which we call an *iterative rounding* algorithm because it alternates between rounding some variables and solving the residual problem on the unrounded variables. In every step, the reference to (**Feasibility-LP2**) is with respect to the current bipartite graph G in the algorithm.

Algorithm 1 A Relaxed Decision Procedure via Iterative Rounding

1: $G \leftarrow (J \cup M; \{(i,j) : p_{ij} \leq T\})$
2: $T_i \leftarrow T$ for each $i \in M$
3: **if** (**Feasibility-LP2**) is infeasible **then**
4:    **return** *no solution*
5: **while** $J \neq \emptyset$ **do**
6:    Find an extreme point solution $\bar{x}$ for (**Feasibility-LP2**)
7:    Delete every edge (i,j) from E such that $\bar{x}_{ij} = 0$
8:    **if** $\bar{x}_{ij} = 1$ for some $(i,j) \in E$ **then**
9:       assign j to i, reduce T_i by p_{ij}
10:       remove j from J
11:    **if** some i has ≤ 1 neighbour in G **then**
12:       if i has a neighbour j in G, assign j to i and remove j from G
13:       remove i from G
14:    **if** some i has exactly 2 neighbours j, j' **and** $x_{ij} + x_{ij'} \geq 1$ **then**
15:       assign j, j' to i
16:       remove i, j, j' from G

We will soon prove that in every iteration of the **while** loop that (**Feasibility-LP2**) has a feasible solution and that some job or edge is removed from G in each iteration. If so, then the number of iterations is at most $|E| + |J|$ so it is a polynomial-time algorithm. The following also holds.

Lemma 1 *If the algorithm terminates, then assignment has makespan $\leq 2T$.*

Proof. Consider the load of a machine i upon termination of the algorithm. Let J_i^* be the set of jobs assigned to i over all executions of step 10, and let J_i be *all* jobs that are assigned to i over all iterations. Also let $T_i' = \sum_{j \in J_i^*} p_{ij}$. We first make a few simple observations.

Observation 1: $|J_i - J_i^*| \leq 2$ because i is removed from G as soon as it is assigned a job j in either step 12 or step 15.

Observation 2: $T_i' \leq T$ because we have that $p_{ij} \cdot \overline{x}_{ij} \leq T_i$ and $\overline{x}_{ij} = 1$ just when $j \in J_i^*$ is assigned to i. When j is assigned to i, we reduced T_i by $p_{ij} = p_{ij} \cdot \overline{x}_{ij}$, so $T_i' = \sum_{ij} p_{ij} \leq T$.

Observation 3: If $|J_i - J_i^*| \leq 1$, then $\displaystyle\sum_{j \in J_i} p_{ij} \leq 2 \cdot T$. This is because $J_i = J_i^*$ means the load on machine i is exactly $T_i' \leq T$. Otherwise, the load is exactly $T_i' + p_{ij'}$ where $j' \in J_i - J_i^*$. Since $(i,j) \in E$, then by step 1 we have $p_{ij'} \leq T$.

The only thing left to show is that if $P_i^* - P_i = \{j, j'\}$ then the load of machine i is at most $2 \cdot T$. We bound this as follows, where T_i and $\bar{x}$ refer to the values in Algorithm 1 just before j and j' are assigned to i in step 15. Note that this T_i value is precisely $T - T_i'$ where $T_i' = \sum_{j* \in J_i^*} p_{ij*}$.

$$
\begin{aligned}
\sum_{j'' \in J_i} p_{ij''} &= T_i' + p_{ij} + p_{ij'} \\
&= T_i' + (1 - \bar{x}_{ij}) \cdot p_{ij} + (1 - \bar{x}_{ij'}) \cdot p_{ij'} + \bar{x}_{ij} \cdot p_{ij} + \bar{x}_{ij'} \cdot p_{ij'} \\
&\leq T_i' + (2 - \bar{x}_{ij} - \bar{x}_{ij'}) \cdot T + \bar{x}_{ij} \cdot p_{ij} + \bar{x}_{ij'} \cdot p_{ij'} \\
&\leq T_i' + T + T_i \\
&= T_i' + T + (T - T_i) = 2 \cdot T
\end{aligned}
$$

Here, the first inequality is because $(i, j), (i, j') \in E$ so $p_{ij}, p_{ij'} \leq T$ by step 1. The second is because $\bar{x}$ is a feasible solution to (**Feasibility-LP2**).

■

Lemma 2 *In every iteration, there is a solution to (***Feasibility-LP2***) and $|E| + |J|$ strictly decreases.*

Proof. We first show (**Feasibility-LP2**) has a feasible solution in each iteration. Initially this is true, otherwise Algorithm 1 would have returned *no solution*. Now consider an iteration that starts with a feasible solution $\bar{x}$ for the LP over the graph G. In the body of this loop, some edges and nodes of G are removed to get a subgraph G'. It is easy to verify that in each case, the restriction of $\bar{x}$ to the remaining edges and nodes remains feasible for the subgraph G', so the next iteration will have a feasible LP solution as well.

Finally, we prove that an edge or job node is always removed from G. So, suppose that at the start of an iteration with the extreme point $\bar{x}$ that $0 < \bar{x}_{ij} < 1$ for each $(i, j) \in E$ and that no $i \in M$ has degree one in G (otherwise some edge or job node will be removed from G). We will show that some $i \in M$ has exactly two neighbours j, j' with $\bar{x}_{ij} + \bar{x}_{ij'} \geq 1$, in which case both j and j' will be removed.

By assumption, every $i \in M$ has $\deg(i) \geq 2$. For each $j \in J$, because $\sum_{i:(i,j)\in E} \bar{x}_{ij} = 1$ and no $\bar{x}_{ij}$ is exactly 1, then $\deg(j) \geq 2$ as well. By the characterization of extreme points, we have that there are at least $|E|$ tight constraints under $\bar{x}$. Since none of these tight are nonnegativity constraints and since there are $|J| + |M|$ other constraints, then:

$$\begin{aligned} |J| + |M| &\geq \#\text{ tight constraints} \\ &\geq |E| \\ &= \frac{\sum_{j\in J} \deg(j) + \sum_{i\in M} \deg(i)}{2} \\ &\geq \frac{2|J| + 2|M|}{2} \\ &= |J| + |M| \end{aligned}$$

So every inequality must hold with equality. In particular, every node of G has degree exactly 2. Counting degrees on both sides of G, we have $2 \cdot |J| = |E| = 2 \cdot |M|$ so in fact $|J| = |M|$.

We conclude by observing

$$|M| = |J| = \sum_{j\in J} \sum_{i:(i,j)\in E} \bar{x}_{ij} = \sum_{i\in M} \sum_{j:(i,j)\in E} \bar{x}_{ij}.$$

In particular, there must be some $i \in M$ such that $\sum_{j:(i,j)\in E} \bar{x}_{ij} \geq 1$. This is what we wanted to show: some machine i has degree 2 and $\sum_{j:(i,j)\in E} \bar{x}_{ij} \geq 1$.

■

28.2 Discussion

A 2-approximation for this problem was first presented by Lenstra, Shmoys, and Tardos [LST90]. Currently, the best lower and upper bounds are exactly what are presented in this lecture: the 2-approximation and the NP-hardness of approximating better than 3/2.

Algorithm 1 can be easily modified to address a more general problem called the General Assignment Problem. The input is much like that for the unrelated machine scheduling problem, except that the input also includes running time bounds T_i for each machine i and a costs c_{ij} for every potential $j \to i$ assignment. The goal is to find a minimum c-cost assignment $\phi : J \to M$ such that no machine $i \in M$ is assigned a running time load of more than T_i. A modification of Algorithm 1 (see [LRS11] for details) will either find a solution with cost at most OPT (if there is a feasible solution) where each machine i has running time at most $2 \cdot T_i$. Such an algorithm was initially presented in [ST93].

It seems we can do more in an interesting special case. Consider an instance of MINIMIZING MAKESPAN ON UNRELATED MACHINES where each job j can only be processed by some machines, but it has the same processing time on each of these machines. That is, $p_{ij} \in \{p_j, \infty\}$ for each $j \in J, i \in M$. Svensson shows that for any constant $\epsilon > 0$, we can compute a value v^* such that $OPT \leq v^* \leq (33/17 + \epsilon) \cdot OPT \approx 1.9412 \cdot OPT$ in polynomial time [S11]. This is accomplished through rounding a (feasibility) LP relaxation that can be solved with a $(1+\epsilon)$-factor in polynomial time, thus bounding the "integrality gap", but the rounding algorithm itself takes exponential time. Such an algorithm that approximates the optimum solution value without producing a corresponding feasible solution is sometimes called an *estimation algorithm*. It would be interesting to see a polynomial-time approximation algorithm for this case that actually produces a feasible solution with makespan within some constant factor $c < 2$ of the optimum.

References

LRS11 L. C. Lau, , R. Ravi, and M. Singh. *Iterative methods in combinatorial optimization.* Cambridge University Press, 2011.

LST90 J. K. Lenstra, D. B. Shmoys, and É. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46:259–271, 1990.

ST93 D. B. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming*, 62:461–474, 1993.

S11 O. Svensson. Santa Claus schedules jobs on unrelated machines. In Proceedings of STOC, 2011.