CSC2420 Algorithm Design, Analysis and Theory	September 22, 2010
Lecture 2	
Lecturer: Allan Borodin	*Scribe: Joel Oren*

1 The makespan problem for identical machines — continued

Last time we gave a Brute force+Greedy algorithm that runs in $O(m^{m\cdot s}\cdot n)$-time to get a $(1+\frac{1}{s})$-approximation for the minimum makespan problem on m identical machines.

If we fix m, so that the number of machines is not a parameter of the problem, and take $s \geq \frac{1}{\epsilon}$, then we get an algorithm that runs in $O(m^{m/\epsilon}\cdot n)$-time algorithm and gives a solution that is a $(1+\epsilon)$ approximation. If m is fixed, this is can be considered a linear time approximation algorithm — PTAS algorithm.

We are interested in an algorithm that runs in poly time for a fixed ϵ, and arbitrary parameters m, n. Our algorithm's key ingredients will be:

1. Suppose we have a procedure that for each T will either declare that $\nexists$ a schedule with *makespan* $\leq T$, or will produce a schedule with *makespan* $\leq c\cdot T$.

   Remark Both answers are valid for the case where $OPT \in (T, c\cdot T]$.

   We can then get a c-approximation by running a binary search.

2. By scaling down "large jobs" we can produce a problem instance such that there exists only a small number of job types, say, $d = s^2$ types. We let $\{z_1, \ldots, z_d\}$ be the distinct job sizes. So:
$$p_i \in \{z_1, \ldots, z_d\}, \text{ for each job } j \tag{1}$$

3. Use DP (dynamic programming) to optimally solve the problem with d job types in time $n^{O(d)} \sim n^{2d}$.

4. Fill in small jobs greedily.

Theorem 1. *Given makespan problem with target T, in which there exists at most d job types, there exists a $n^{O(d)}$-steps DP algorithm that either reports that there is no solution with makespan $\leq T$, or finds an optimal solution.*

The DP algorithm Let $z_1, \ldots, z_d$ be the sizes of the job types. Define a configuration on a given machine:
$$\overrightarrow{N} = (N_1, \ldots, N_d), \text{ where } N_i = \# \text{ of jobs of size } z_i \tag{2}$$
And let $V = \{(N_1, \ldots, N_d) | \sum_i N_i \cdot z_i \leq T\}$ be the set of configurations for a machine that do not exceed T. Clearly $|V| \leq n^d$, since $N_i \leq n, \forall i$.

Let $M(x_1, \ldots, x_d)$ denote the minimal number of machines required for scheduling x_i jobs of size z_i within a makespan T, for every $i \in [1, d]$. We want to know whether $M(r_1, \ldots, r_d) \leq m$, where the input has r_i jobs of size z_i, for every $i \in [1, d]$.

Consider the following DP recursion relation:
$$M(x_1, \ldots, x_d) = 1 + min_{\overrightarrow{N}\in V} M(x_1 - N_1, \ldots, x_d - N_d),$$

which considers all the possible assignments for the first machine. The DP matrix has $O(n^d)$ entries, and filling each entry requires $O(n^d)$ recursive calls, which gives a running time of $O(n^{2d})$.

We now define the approximation algorithm that produces a solution within $[T, (1+1/s)\cdot T]$. Define a large job to be one where $p_j > T/s$. Round p_j down to nearest the multiple of T/s^2, and let the rounded size be p'_j. We consider the makespan problem for the modified set of jobs.

Hand-waiving comment: Assume integrality wherever necessary.

We make the following observations:

1. For any valid solution with makespan T, there exist *at most* s large jobs on any machine. This holds since for every job j $p_j \leq T/s$.
2. $p_j - p_j \leq T/s^2$.
3. There are at most s^2 job types.

We now run the DP algorithm on the set of rounded large jobs, T, and $d = s^2$. Notice that if the DP algorithm reports that no solution within makespan $\leq T$ exists for the set of rounded down jobs, it must also be the case for the original set of jobs. On the other hand, if such a solution is produced by the DP algorithm, we can restore the job loads in it to get a makespan within:

$$s \cdot T/s^2 = (1 + \frac{1}{s}) \cdot T, \tag{3}$$

which follows from observations 1 and 2.

Then greedily assign the small jobs (of load $\leq T/s$) to the schedule (the task of proving that the approximation ratio remains the same is left as a homework assignment).

The algorithm:

Input: Jobs $J = \{p_j\}_{j=1,\dots,n}$, m identical unrelated machines, non-negative number T.
Output: "No T-makespan solution" if no solution with makespan $\leq T$ exists.
Otherwise: a schedule of the n jobs with a makespan $\leq (1+1/s)\cdot T$.
Let $J' = \{p'_j | p_j \in J, p_j > T/s, p'_j = \lfloor \frac{p_j}{T/s^2} \rfloor\}$;
Run DP algorithm using $J', d = s^2, T$;
If returned "No solution" return "No solution".;
Otherwise, use the returned schedule for J and fill in the remaining small jobs using the greedy algorithm. ;
If the greedy algorithm fails, then return "No T-makespan solution".

Procedure `ScheduleT`($\{p_j\}_{j=1,\dots,n}, m, T$)

Input: Set of jobs $J = \{p_j\}_{j=1,\dots,n}$, set of m identical unrelated machines
Output: A schedule of the n jobs with a makespan that is within a factor of $(1+1/s)$ of the optimum
Use binary search on T to look for the minimal makespan T, using $ScheduleT(J, m, T)$;

Algorithm 2: The $(1+1/s)$-approximation algorithm

2 Integer Programming (IP) and Linear Programming (LP)

In order to introduce the method of IP/LP, we consider the following problem:

2.1 The Vertex Cover Problem

Let $G = (V, E)$ be a graph with vertex weight function: $w : V \to \mathbb{R}$. Let $|V| = n$. We say that $V' \subseteq V$ is a *vertex cover* if for all $(u, v) \in E$ either $u \in V'$ or $v \in V'$ (or both).

Goal: Minimize:

1. $|V'|$ (unweighted case).
2. $\sum_{u \in V'} w_i$ (weighted case).

This problem is known to be NP-hard. Furthermore, it is known to be NP-hard to approximate within a ~ 1.38 approximation.Subject to other complexity assumptions (Khot's unique games conjecture — UGC), it is NP-hard to get a $2 - \epsilon$ approximation for any $\epsilon > 0$.

The corresponding IP for this problem:

$$\begin{array}{rl} \min & \sum_{i \in [n]} w_i \cdot x_i \\ \text{subject to:} & x_u + x_v \leq 1, \forall (u, v) \in V \\ & x_i \in \{0, 1\}, \forall i \in [n] \end{array}$$

Where the intended meaning of x_i:

$$x_i = \begin{cases} 0 & v_i \notin V' \\ 1 & v_i \in V' \end{cases} \tag{4}$$

The LP relaxation of the IP would replace the last line with:

$$0 \leq x_i \leq 1 \tag{5}$$

Any instance of an LP can be solved optimally in poly-time. However, the worst case poly-time algorithms have time complexity$\sim O(n^{3.5} \cdot L)$, where $L =$ length of description of input instance. Practically, the Simplex method often beats the worst case methods.

Open problem: Does there exist a strongly poly-time (in m, n) algorithm for solving an LP with an $m \times n$ constraint matrix?

The canonical minimization problem LP

$$\begin{array}{rl} \min & \overrightarrow{c} \cdot \overrightarrow{x} \\ \text{subject to:} & A_{m \times n} \cdot \overrightarrow{x} \geq \overrightarrow{b} \end{array}$$

This is called the canonical/standard LP form for a minimization problem.

The canonical LP form for a maximization problem:

$$\begin{array}{rl} \max & \overrightarrow{C} \cdot \overrightarrow{x} \\ \text{subject to:} & A_{m \times n} \cdot \overrightarrow{x} \leq \overrightarrow{b} \end{array}$$

When all of the coefficients are non-negative, the minimization form is often referred to as a covering problem; the maximization form is referred to as a packing problem.

Let $\overrightarrow{x}$ and $\hat{x}$ be the optimal LP and integral solutions, respectively. Clearly:

$$cost(\overrightarrow{x}) \leq cost(\hat{x}), \tag{6}$$

since the LP relaxation can have a larger solution space. Round the LP solution:

$$\overline{x}_i = \begin{cases} 0 & x_i < 0.5 \\ 1 & x_i \geq 0.5 \end{cases} \tag{7}$$

Claim 2. *Let $\overline{x} = (x_1, \dots, x_n)$. Then:*

1. *$\overline{x}$ is an integral solution.*
2. *$cost(\overline{x}) \leq 2 \cdot cost(\overrightarrow{x}) \leq 2 \cdot OPT$.*

2.2 Returning to The Makespan Problem

Unrelated machines model (non-uniform): Job j is described by a vector $(p_1, \dots, p_{jm})$, where p_{ji} is the processing time/load on the j'th for job machine i.

Special case: Restricted machines model For each job j, and machine i: $p_{ji} \in \{p_j, \infty\}$. That is, each job is allowed to run on some machines, and on the allowed machines the load is the same.
Remark the natural greedy algorithm for this problem is a $log_2 m$ approximation and this is a very tight bound within an additive 1.

Open Problem: Is there a greedy or DP algorithm that has a $O(1)$ approximation.

A special case of the restricted machines model is:

$$|\{i \in [m] : p_{ji} < \infty\}| \leq 2, \forall j, \tag{8}$$

Consider an even more restricted case:

$$|\{i \in [m] : p_{ji} < \infty\}| = 2, \forall j \tag{9}$$

in which job is allowed to run on exactly two machines. This is often referred to as *The Graph Orientation Problem*.

We can view this as problem on a multigraph $G = (V, E)$.

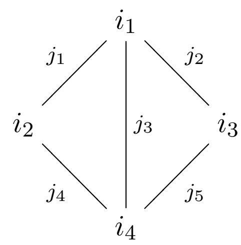

Figure 1: Multigraph representation of the job scheduling problem

where: V = set of machines, E = set of jobs.

We want to direct the edges, such that the maximal collective weight of incoming edges for any node is reduced. Lenstra, Shmoys and Tardos ([**?**]) showed how to use IP/LP rounding to achieve a 2-approximation (for the unrelated machines model). They also show that it is NP-hard to achieve better than a 1.5 approximation for the graph orientation problem. The gap between the 2-approximation, which has been improved to a PTAS $(1 - \frac{1}{m})$-approximation algorithm for a fixed m, and the 1.5 inapproximability result has been open for 20 years now.

In SODA2008 Ebenlendr, Krcal and Sgall ([**?**]) were able to obtain a 1.75 approximation for the graph orientation problem. The online greedy algorithm for the restricted machines model has been shown to give a $log_2 m$-approximation ratio, which is tight even for the graph orientation problem.