IEOR 8100: Scheduling Algorithms 10/13/16

Lecture 10

Instructor: Cliff Stein Scribed by: Mauro Escobar

1 PTAS for $1|r_j|\sum C_j$

Consider an instance I of $1|r_j|\sum C_j$ with optimal value OPT and $\epsilon > 0$. Each of the following transformations of the original instance will increase its optimal value by a factor that is less or equal than $(1+\epsilon)$. Therefore, we will get a new instance I' which optimal value will be $\leq (1+\epsilon)^k \approx 1 + k\epsilon = 1 + O(\epsilon)$, where k is the number of transformations done to I.

Transformations:

1. Round r_j and p_j up to powers of $(1+\epsilon)$. The new release date r_j will be $R_x := (1+\epsilon)^x$ for some $x \in \mathbb{N}$. Define the interval $I_x := [R_x, R_{x+1}]$, then $|I_x| = (1+\epsilon)^{x+1} - (1+\epsilon)^x = \epsilon(1+\epsilon)^x = \epsilon R_x$.
2. Charging the end of the interval as completion time.
3. Set $r_j = \max\{r_j, \epsilon p_j\}$.

With these transformations we have the following lemma.

Lemma 1. *Each job runs in at most $s := \log_{1+\epsilon}\left(1 + \frac{1}{\epsilon}\right)$ intervals.*

Proof. Note that $p_j \leq \frac{1}{\epsilon} r_j = \frac{1}{\epsilon} R_x = \frac{1}{\epsilon^2}|I_x|$, for some $x \in \mathbb{N}$. Therefore, the size of the s consecutive intervals (starting with I_x) is

$$\sum_{j=0}^{s-1} |I_{x+j}| = \sum_{j=0}^{s-1} |I_x|(1+\epsilon)^j = \frac{(1+\epsilon)^s - 1}{\epsilon}|I_x| = \frac{1}{\epsilon^2}|I_x| \geq p_j.$$

□

We say that a job j is *small* if $p_j \leq \epsilon |I_x|$, where $r_j = R_x$. Then,

- if all the jobs are small, then SPT is $(1+\epsilon)$-optimal;
- there are at most $\frac{1}{\epsilon}$ non-small (large) jobs.

Let $t = \epsilon^7 OPT$, then

- no more than $\frac{1}{\epsilon^7}$ jobs have $C_j > t$.

Lemma 2. *There is a $(1+O(\epsilon))$-optimal schedule in which each job is either: small when it runs, or runs after time t.*

Proof. Let $k = \log_{1+\epsilon}\frac{1}{\epsilon^4}$. In the optimal schedule OPT, take each job j that is not small and run before time t, and move it forward k intervals. Note that

$$p_j \leq \frac{1}{\epsilon} R_x = \frac{1}{\epsilon^2}|I_x| \qquad \text{and} \qquad |I_{x+k}| = |I_x|(1+\epsilon)^k = |I_x|\frac{1}{\epsilon^4}.$$

Then, $p_j \le \frac{1}{\epsilon^2}|I_x| = \epsilon^2|I_{x+k}|$. Since the intervals are expanded by a factor of $(1+\epsilon)$, job j can be held inside one interval when it is moved forward k intervals, moreover, job j is going to be small in interval I_{x+k}. And even if $\frac{1}{\epsilon}$ jobs are sent forward to an interval I_{x+k}, in total they use $\le \frac{1}{\epsilon} \cdot \epsilon^2|I_{x+k}| = \epsilon|I_{x+k}|$. Therefore,

$$\begin{aligned}\sum_{\substack{j\text{: job pushed}\\ \text{forward}}} C_j &\le \frac{1}{\epsilon}\cdot\epsilon^3 OPT + \frac{1}{\epsilon}\cdot\frac{\epsilon^3 OPT}{1+\epsilon} + \frac{1}{\epsilon}\cdot\frac{\epsilon^3 OPT}{(1+\epsilon)^2} + \cdots \\ &= \epsilon^2 OPT\left(1 + \frac{1}{1+\epsilon} + \frac{1}{(1+\epsilon)^2} + \cdots\right) \\ &= \epsilon^2 OPT \frac{1+\epsilon}{\epsilon} = \epsilon(1+\epsilon)OPT.\end{aligned}$$

□

With these analysis, we consider the following algorithm.

Algorithm 1 PTAS for $1|r_j|\sum C_j$

```
1: Guess which $\frac{1}{\epsilon^7}$ jobs run after time $t$.
2: Guess the order these jobs run in.
3: Schedule the remaining jobs by SPT.
```

The correctness of Algorithm 1 is given by the fact that, by Lemma 2, the completion time of large jobs (the ones that are moved forward) is $O(\epsilon)OPT$ and SPT is $(1+\epsilon)$-optimal for the small jobs. The complexity of Algorithm 1 is $O\left(n^{1/\epsilon^7} n\log n\right)$. However, this can be improved to $O\left(n\log n + (\frac{1}{\epsilon})!\right)$.

2 $1|\textbf{prec}|\sum w_jC_j$

We will consider different mathematical programming formulations for $1|\text{prec}|\sum w_jC_j$:

1. If the variables are $\{C_j\}$, the completion times,

$$\begin{aligned} \text{minimize} \quad & \sum w_j C_j \\ \text{subject to} \quad & C_j \ge r_j + p_j, && \forall j \\ & C_j \le C_k - p_k, && \forall j \prec k \\ & C_j \le C_k - p_k \ \textbf{or}\ C_k \le C_j - p_j, && \forall j,k. \end{aligned} \tag{1}$$

2. If the variables are $\{\delta_{ij}\}$ where $\delta_{ij} = \begin{cases}1, & \text{if job } j \text{ runs before job } j,\\ 0, & \text{otherwise.}\end{cases}$

3. If the variables are $\{x_{jt}\}$ where $x_{jt} = \begin{cases}1, & \text{if job } j \text{ finishes at time } t,\\ 0, & \text{otherwise.}\end{cases}$

Let T be an upper bound on the schedule length, then $C_j = \sum_{t \le T} t x_{jt}$.

$$\begin{array}{llll}
\text{minimize} & \displaystyle\sum_j w_j \sum_{t \le T} t x_{jt} & & \\
\text{subject to} & \displaystyle\sum_t x_{jt} = 1, & \forall j & \text{(every job runs)} \\
& \displaystyle\sum_j \sum_{s=t+1}^{t+p_j} x_{js} \le 1, & \forall t \le T & \text{(at each } t \text{ at most one job runs)} \\
& \left.\begin{array}{l} \sum_t t x_{jt} \le \sum_t t x_{kt} - p_k \\ \sum_{s=1}^{t} x_{js} \ge \sum_{s=1}^{t+p_k} x_{ks} \end{array}\right\} & \forall j \prec k,\ \forall t = r_j + p_j, \dots, T - p_k & \text{(precedent constraints)} \\
& x_{jt} = 0 \text{ if } t < r_j + p_j & \forall j, \forall t & \text{(release dates)} \\
& x_{jt} \in \{0,1\} & \forall j, \forall t & \text{(integrality)}
\end{array}$$

Consider the relaxed problem, that is, replacing the constraint $x_{jt} \in \{0,1\}$ by the weaker constraint $0 \le x_{jt} \le 1$. What does this relaxed problem mean?

Note that there is an exponential number of variables and constraints. However, we can deal with problem, instead of considering $t \in \{1, \dots, T\}$ we allow $t \in \{1, (1+\epsilon), (1+\epsilon)^2, \dots\}$, which is a set of size $\log_{1+\epsilon} T$. We obtain then a poly-sized integer program whose optimum is a $(1+\epsilon)$-approximation of the real optimum.

About the **or** constraints in (1):

$$C_j \le C_k - p_k \textbf{ or } C_k \le C_j - p_j, \quad \forall j, k \tag{2}$$

From [1]: the difficulty with this 'disjunctive' constraints is that they are not linear inequalities and cannot be modeled using linear inequalities. Instead we use a class of valid inequalities introduced by Queyranne (1993) and Wolsey (1985), that are motivated by considering Smith's rule for scheduling the jobs when there are no release dates or precedence constraints. Smith (1956) proved that a schedule is optimal if and only if the jobs are scheduled in order of non-decreasing ratio p_j/w_j. As a result, if we set $w_j = p_j$ for all j, then the sum $\sum_j w_j C_j = \sum_j p_j C_j$ is invariant for any ordering of the jobs. In particular, for the ordering $1, \dots, n$, if there is no idle time in schedule then $C_j = \sum_{k=1}^{j} p_k$; therefore, for any schedule we can write down the valid constraint

$$\sum_{j=1}^{n} p_j C_j \ge \sum_{j=1}^{n} p_j \sum_{k=1}^{j} p_k = \sum_{j=1}^{n} \sum_{k=1}^{j} p_j p_k = \frac{1}{2}\left(p^2(J) + p(J)^2\right), \tag{3}$$

where the inequality results from the possibility of idle time in the schedule, and

$$p(S) := \sum_{j \in S} p_j, \qquad p(S)^2 = \Big(\sum_{j \in S} p_j\Big)^2, \qquad p^2(S) = \sum_{j \in S} p_j^2.$$

Consider the completion times for a feasible schedule, C_j, $j \in J$. For each subset $S \subset J$, we can consider the instance induced by S; the induced completion times C_j, $j \in S$, correspond to a feasible schedule for this smaller instance. Hence, we can apply the precious inequality (3) to each subset and derive the following valid inequalities:

$$\sum_{j \in S} p_j C_j \ge \frac{1}{2}\left(p^2(S) + p(S)^2\right), \quad \forall S \subseteq J. \tag{4}$$

As explained in [1], we note that these inequalities remain valid even if we allow the schedule to be preemptive; that is, the processing of a job may be interrupted and continued at a later point in time. Furthermore, Queyranne (1993) and Wolsey (1985) have shown that constraints (4) are sufficient to describe the convex hull of completion time vectors of feasible schedules for instances of $1||\sum w_jC_j$. These constraints are no longer sufficient, however, if we add constraints that enforce release dates. Although we do not have exact characterizations for $1|\text{prec}|\sum w_jC_j$, $1|r_j|\sum w_jC_j$, or $1|r_j,\text{prec}|\sum w_jC_j$, we will show a linear relaxation that can be used to find near optimal solution of $1|\text{prec}|\sum w_jC_j$.

Note also that there is an exponential number of constraints in (4). Therefore, we can use the Ellipsoid method with a separation oracle to solve the relaxation of

$$\begin{array}{rll} \text{minimize} & \sum w_jC_j & \\ \text{subject to} & C_j \le C_k - p_k, & \forall j \prec k \\ & \displaystyle\sum_{j\in S} p_jC_j \ge \frac{1}{2}\left(p^2(S) + p(S)^2\right), & \forall S \subseteq J. \end{array} \tag{5}$$

Algorithm 2 2-approximation of $1|\text{prec}|\sum w_jC_j$

```
1: Solve the LP relaxation of (5). Let $C'_1, \dots, C'_n$ be the solution.
2: Renumber the jobs so that $\hat{C}_1 \le \dots \le \hat{C}_n$.
3: Schedule greedily in the order of $\hat{C}_j$, to obtain $\tilde{C}_j$.
```

Claim 3. $\hat{C}_j \ge \frac{1}{2}\sum_{k=1}^{j} p_k$.

Proof. Let $S = \{1, \dots, j\}$. Then,

$$\sum_{k=1}^{j} p_k\hat{C}_k \ge \frac{1}{2}\left(p^2(S) + p(S)^2\right) \qquad \text{and} \qquad \hat{C}_j \ge \hat{C}_k,\ \forall k \in \{1, \dots, j\}.$$

Therefore,

$$p(S)\hat{C}_j = \sum_{k=1}^{j} p_k\hat{C}_j \ge \sum_{k=1}^{j} p_k\hat{C}_k \ge \frac{1}{2}\left(p^2(S) + p(S)^2\right) \ge \frac{1}{2}p(S)^2.$$

Finally, $\hat{C}_j \ge \frac{1}{2}p(S) = \frac{1}{2}\sum_{k=1}^{j} p_k$. □

Lemma 4. *Algorithm 2 is 2-approximation of* $1|\text{prec}|\sum w_jC_j$.

Proof. By feasibility of $\{C'_j\}$, we have that $C'_j \le C'_k - p_k \le C'_k$ for all $j \prec k$, that is $\{C'_j\}$ satisfy the precedent constraints. Therefore, after renumbering the jobs, the schedule given by the order of $\{\hat{C}_j\}$ also satisfy the precedent constraints.

Finally, by Claim 3 we obtain that $\tilde{C}_j = \sum_{k=1}^{j} p_k \le 2\hat{C}_j$. Then, $\sum_j w_j\tilde{C}_j \le 2\sum_j w_j\hat{C}_j = 2LP$. □

References

[1] L. A. Hall, A. S. Schulz, D. B. Shmoys, and J. Wein. Scheduling to minimize average completion time: Off-line and on-line approximation algorithms. *Mathematics of Operations Research*, 22(3):513–544, Aug. 1997.