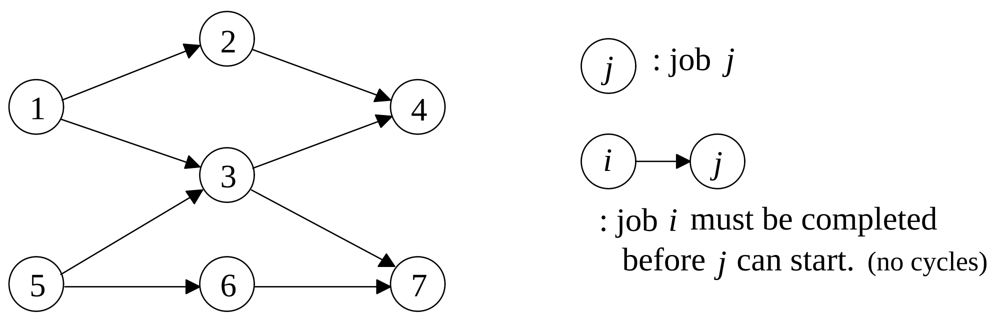
Computational Intractability 2010/6/17

Lecture 9

Professor: David Avis *Scribe:Daichi Paku, rev. DA 2011/6/30*

1 Single machine scheduling with precedence constraints



1.1 Problem

Input: a precedence graph with n jobs, and each job j has processing time p_j.

Several possible objective:

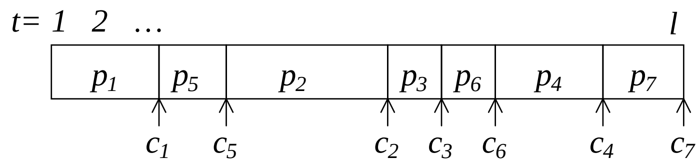
(i) makespan: minimum length schedule. $\rightarrow$ This is easy, by topological sort.

(ii) minimum sum of completion times, possibly weighted.

$$\min \sum_{j=1}^{n} w_j c_j \quad \left(\begin{array}{lcl} c_j & : & \text{completion time of job } j. \\ w_j & : & \text{weight of job } j. \end{array} \right)$$

This is NP-hard, and we discuss this problem in this lecture.

Feasible schedule: just a permutation of $1, 2, \cdots n$ consistent with the given graph.

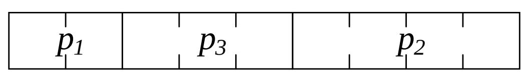


In this example, $c_1 = p_1, c_2 = p_1 + p_5, c_3 = p_1 + p_5 + p_2, \cdots, c_7 = l$, where $l = \sum_{j=1}^{n} p_j$.

1.2 Smith's rule

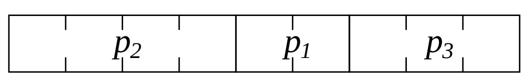
Suppose there are no precedence constraints, we can use Smith's rule.

For example, $p_1 = 2, p_2 = 4, p_3 = 3$. We can gain the optimal solution by scheduling the jobs in nondecreasing order of p_j.



Then, this is optimal.

If we have weights, $w_1 = 1, w_2 = 10, w_3 = 1$, then we schedule the jobs in nondecreasing order of the ratios p_j/w_j.



This is optimal.

1.3 Formulation

Decision variable

$$x_{jt} = \begin{cases} 1 & \text{if job } j \text{ starts at time } t \\ 0 & \text{otherwise} \end{cases} \qquad \begin{pmatrix} j = 1, 2, \cdots, n \\ t = 1, 2, \cdots, l \end{pmatrix}$$

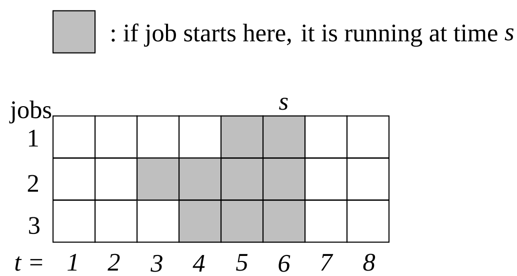
Constraints

1. Each job must start sometime.

$$\sum_{t=1}^{l} x_{jt} = 1 \quad (j = 1, 2, \cdots n) \tag{1}$$

2. At each time exactly only one job is runnning.

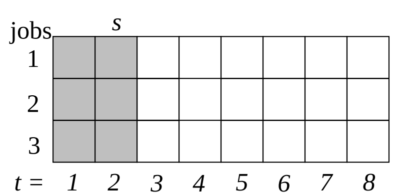
   For example.$(n = 3, p_1 = 2, p_2 = 4, p_3 = 3)$



   Exactly one job must start in the shaded area, so,

$$x_{15} + x_{16} + x_{23} + x_{24} + x_{25} + x_{26} + x_{34} + x_{35} + x_{36} = 1.$$

   Another example.



(b)

   In this case,

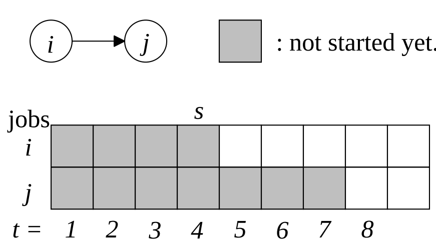
$$x_{11} + x_{12} + x_{21} + x_{22} + x_{31} + x_{32} = 1.$$

   Generally.

$$\sum_{j=1}^{n} \sum_{t=\max(1,s+1-p_j)}^{s} x_{jt} = 1 \quad (s = 1, 2, \cdots l) \tag{2}$$

3. Precedence constraints.

   Example. $(p_i = 3, p_j = 4, i \rightarrow j)$



If job i has not started in time $1, 2, \cdots, s$, job j cannot start in time $1, 2, \cdots, s + p_i$.

$$\sum_{t=1}^{s+p_i} x_{jt} \leq \sum_{v=1}^{s} x_{iv} \quad \begin{pmatrix} s = 1, 2, \cdots l - p_i - p_j \\ \text{for each } (i \to j) \end{pmatrix} \tag{3}$$

4. (Release time: job j cannot start before time r_j)

$$x_{js} = 0 \quad (s = 1, 2, \cdots, r_j - 1) \tag{4}$$

Objective function

If job j starts at time t, that is if $x_{jt} = 1$, then j will finish at $c_j = t + p_j$.
So,

$$\min \sum_{j=1}^{n} w_j c_j = \sum_{j=1}^{n} w_j \Big[\sum_{t=1}^{l} (t + p_j) x_{jt} \Big].$$

1.4 Second formulation

Decision variable

$$x_{ij} = \begin{cases} 1 & \text{if job } i \text{ precedes job } j \text{ in the schedule} \\ 0 & \text{otherwise} \end{cases} \qquad \Big(\text{for all jobs } i, j \text{ distinguished}\Big)$$

For example.

p_2	p_3	p_1	p_4

$$\longrightarrow \quad \begin{pmatrix} x_{23} = 1, x_{21} = 1, x_{24} = 1, & \\ x_{31} = 1, x_{34} = 1, x_{14} = 1, & \text{else } x_{ij} = 0 \end{pmatrix}$$

Constraints

1. Antireflexive. It must be that either job i is before job j, or j is before i in the scheduling, then

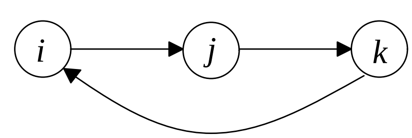
$$x_{ij} + x_{ji} = 1 \quad (\text{for all } i, j) \tag{5}$$

2. Transitivity. We allow no cycles. That means:

(a) If $x_{ij} = 1$ and $x_{jk} = 1$ then $x_{ki} = 0$.

(b) If $x_{jk} = 1$ and $x_{ki} = 1$ then $x_{ij} = 0$.

(c) If $x_{ki} = 1$ and $x_{ij} = 1$ then $x_{jk} = 0$.



and we can write these as the single constraint:

$$x_{ij} + x_{jk} + x_{ki} \leq 2 \quad (\text{for all } i, j, k \text{ distinguished}) \tag{6}$$

Now, we can eliminate half of the variables by using (5).

$$x_{ji} = 1 - x_{ij} \quad (j > i)$$

Then (6) is,

$$\begin{array}{rcl} x_{ij} + x_{jk} - x_{ik} & \leq & 1 \\ -x_{ij} - x_{jk} + x_{ik} & \leq & 0 \end{array} \quad \left(\begin{array}{c} \text{for all } i, j, k \\ i < j < k \end{array} \right) \tag{7}$$

3. Precedence constraints.

   Actually easy.

$$x_{ij} = 1 \quad (\text{for each } (i \to j)) \tag{8}$$

Objective function

For example.

p_4	p_2	p_3	p_1	p_5

$$\longrightarrow \left(\begin{array}{rcl} c_3 & = & p_4 + p_2 + p_3 \\ & = & x_{43}p_4 + x_{23}p_3 + x_{13}p_1 + x_{53}p_5 \end{array} \right)$$

Generally.

$$\min \sum_{j=1}^{n} w_j c_j = \sum_{j=1}^{n} w_j \Big[\sum_{i=1, i \neq j}^{n} p_i x_{ij} + p_j \Big] \tag{9}$$

Again we can eliminate half of the variables using (5).

Question: Can we include release times r_j for each job j in this model?
This looks tricky. Since release time may cause idle time, the current objective function is not correct. Nevertheless, Nemhauser and Savelsbergh [2] showed it could be done as follows. Assume the jobs are labelled so that $0 \leq r_1 \leq r_2 ... \leq r_n$.

- For simplicity, introduce new constant variables $x_{jj} = 1$ for each job j.
- Introduce lower bounds on completion time c_j for each job j as follows:

$$c_j \geq r_i x_{ij} + \sum_{k < i, k \neq j} p_k (x_{ik} + x_{kj} - 1) + \sum_{k \geq i, k \neq j} p_k x_{kj} + p_j \qquad 1 \leq i, j \leq n \tag{10}$$

- Use the objective function $\min \sum_{j=1}^{n} w_j c_j$

The correctness of the lower bound on c_j can be seen as follows. Let i be any job that is processed before j, ie. $x_{ij} = 1$. Clearly job i cannot start before r_i. To this we can add the following to get a lower bound on c_j:

- the processing times of all jobs $k < i$ (which by assumption have release time at most r_i) which go after job i and before job j, ie. $x_{ik} + x_{kj} - 1 = 1$.
- the processing times of all jobs $k \geq i$ (which by assumption have release time at or after r_i) which go before job j, ie. $x_{kj} = 1$.

- the processing time of job j.

To see the correctness of the objective function, consider an optimum solution to the problem and let x_{ij} be set according to this solution. We need to see that c_j as specified by the bounds (10) is the correct value for the completion time of job j, $j = 1, 2, ..., n$. This means that it should satisfy at least one inequality as an equation, and this equation should give the correct value of c_j. In the optimum solution, the jobs are scheduled in consective blocks that contain no idle time. The blocks are separated by idle time. Let B be the block containing job j. If j is the first job in B then necessarily j starts at r_j and (10) is an equation giving the correct completion time $r_j + p_j$ since the two summations are empty. Otherwise let $i \neq j$ be the first job in the block B. As there is no idle time in B, j will start immediately after the sum of the processing times of all jobs that precede it and are either i or follow i in the schedule. For jobs with $k \geq i$ we require only $x_{kj} = 1$ since they could not be scheduled before r_i. For jobs with $k < i$ we also require $x_{ik} = 1$, for otherwise they would be scheduled in another block. Therefore (10) is satisfied as an equation for this value or i and j and gives the correct completion time for job j.

As a final note, in (10) we could eliminate the second summation entirely and drop the condition $k < i$ in the first summation. However, the formulation (10) gives a stronger linear programming relaxation.

References

[1] A.B. Keha, K. Khowala. J.W. Fowler, "Mixed integer programming formulations for single machine scheduling problems",Computers & Ind. Eng. 56(2009)357-367.

[2] G. L. Nemhauser and M.W.P. Savelsbergh, "A cutting plane algorithm for the single machine scheduling problem with release times," NATO ASI serries F: Computer and Systems Sciences 82(1992)63-84.