Lecture 7: Single Machine Environments (Minimizing Weight of Late Jobs)

May 26th, 2009

---

1 Minimizing the Weight of Late Jobs ($1||\sum w_jU_j$)

Recall that in a schedule a job j is late ($U_j = 1$) if $C_j > d_j$. We wish to find a schedule which minimizes the total weight of the late jobs. To understand the optimal substructure we need to understand how the optimal schedule looks like.

Note that any schedule S divides the jobs into two sets: L^S, late jobs with $U_j = 1$ and T^S, timely jobs with $U_j = 0$. Also note that minimizing the total weight of late jobs is equivalent to maximizing the total weight of timely jobs. Now, we make the following observations.

Observation 1.1. *We may assume that S processes all the jobs in T^S before processing any job in L^S.*

Proof. Since a job in L^S is late, it doesn't matter when it is scheduled. Thus, if a job in T^S is scheduled after a job in L^S, swapping them doesn't change the objective. □

Observation 1.2. *If T is the set of timely jobs, then we may assume that the schedule S processes them in increasing order of due date (EDD).*

Proof. We know that processing EDD minimizes L_{max} and thus is T can be scheduled in a manner such that $L_{max} \leq 0$, then EDD will achieve it. □

We call a subset of jobs feasible if when processed in the EDD order none of the jobs is late. Thus, the problem boils down to recognizing the maximum weight feasible subset of jobs. Now we are ready to describe the optimal substructure of the problem. Order the jobs in increasing order of their due dates as $\{1, \ldots, n\}$.

Maintain a table $T[i, q]$ which returns a feasible subset X of jobs from $\{1, \ldots, i\}$ of maximum weight such that $\sum_{j \in X} p_j = q$. If no such subset exists, let $T[i, q]$ be `null`. Also maintain a parallel table $t[i, q]$ which contains the weight of $T[i, q]$. If $T[i, q]$ is `null`, then we let $t[i, q] = -\infty$. Note that i goes from 0 to n and q goes from 0 to $\sum_j p_j$.

Suppose we have constructed the complete table. Then, we look at all the entries of the row $T[n, q]$ and the non-null entry entry with the largest weight must be the maximum weight feasible subset of jobs. We now describe a dynamic program to update the entries of the tables.

Note that the entries $T[i, 0]$ for all i is the empty set $\varnothing$ and thus $t[i, 0] = 0$ for all i. Also note that $T[0, q]$ for any $q \geq 1$ is `null`. Now we state a rule heorem to update the $T[i+1, q]$ entry of the table given entries in the table up to the ith row.

Lemma 1.3.

$$T[i+1,q] = \begin{cases} T[i, q-p_{i+1}] \cup \{(i+1)\}, & \textit{if case A and B happens} \\ T[i,q] & \textit{otherwise} \end{cases}$$

Case A: $q \leq d_{i+1}$
Case B: $t[i, q-p_{i+1}] + w_{i+1} \geq t[i,q]$.

Proof. Consider the set $T[i+1,q]$. Note that if the element $(i+1) \notin T[i+1,q]$, then $T[i+1,q] = T[i,q]$. Thus it suffices to prove that the element $(i+1) \in T[i+1,q]$ implies case A and B, and case A and B implies $T[i+1,q] = T[i, q-w_{i+1}] \cup \{(i+1)\}$.

If $(i+1) \in T[i+1,q]$, then since d_{i+1} is the largest, the EDD will schedule $(i+1)$ job the last. Since the total processing time is *exactly* q, feasibility implies $q \leq d_{i+1}$, that is case A. Also, it must be the case that the weight of $T[i,q]$ is at most the weight of $T[i+1,q]$. Thus, $t[i,q] \leq w_{i+1} + t[i+1,q] - w_{i+1}$. But the set $T[i+1,q] \setminus \{(i+1)\}$ is also a feasible subset of jobs $\{1,\ldots,i\}$ having total weight $t[i+1,q] - w_{i+1}$ and total processing time exactly $q - w_{i+1}$. By definition this must be less than $t[i, q-w_{i+1}]$. This implies case B.

A similar argument shows that case A and B implies $T[i+1,q] = T[i, q-w_{i+1}] \cup \{(i+1)\}$. This is because, A shows that the RHS is feasible and B shows that it has more weight than the case when $(i+1)$ is not in the set. □

The above implies a dynamic program which finds the optimum solution in time $O(n \sum_j p_j)$. Note that it is not necessary to have the index of q go all the way to $\sum_j p_j$ but only to d_{max} suffices.