Continued: Solving $1|\text{prec}|\sum w_jC_j$

Recall from last time that we are working towards a 2-approximation algorithm for $1|\text{prec}|\sum w_jC_j$. We will focus on initial sets $S$, which have the property that if $j \in S$ and $k \rightarrow j$ then $k \in S$. We defined $\rho(S) = \frac{w(S)}{p(S)}$ and $S^*$ to be an initial set with maximum $\rho(S)$ over all initial sets $S$. Our algorithm finds $S^*$, finds a 2-approximate schedule for $S^*$ respecting precedence constraints, and then recurses on $N - S^*$, appending the recursively constructed solution. Last time we showed that any feasible schedule of $S^*$ was within a factor of 2 to the optimal, so now all we need to prove is the following result.

The Sydney Decomposition Lemma

**Theorem 0.1** There exists an optimal schedule in which $S^*$ precedes $N - S^*$.

**Proof.** Define $f(S) = w(S) - \rho^*p(S)$ where $\rho^* = \rho(S^*)$. Note that $f(S^*) = w(S^*) - \rho^*p(S^*) = 0$. By our definition of $\rho^*$, $f(S) \leq 0$ for all initial sets $S$. Observe that maximizing $f$ is equivalent to maximizing $\rho$, and the maximum will achieve the value 0.

Given $S^*$, consider relaxing our input by deleting all precedence constraints from $S^*$ to $N - S^*$. We will look at an optimal solution to this relaxed problem. We claim that $S^*$ still maximizes $f(S)$ over all initial sets (including those new sets that have become initial because of the relaxation).

To show this, we need the following facts about initial sets:

**Fact 0.2** If $A$ and $B$ are initial, then both $A \cup B$ and $A \cap B$ are initial as well.

To see this, suppose $A \cup B$ was not initial. Then some precedence constraint must go from $x \notin A \cup B$ to $y \in A \cup B$. Without loss of generality, assume that $y \in A$. Clearly $x \notin A$, but this contradicts the assumption that $A$ is initial. We can argue similarly about $A \cap B$.

**Claim 0.3** $S^*$ maximizes $f(S)$ over all initial sets in our relaxed problem.

**Proof.** Suppose not. Then there must exist some $S'$ that is initial for our relaxed problem and for which $f(S') > 0$. Define $U = S^* \cup S'$ and $V = S^* \cap S'$. Observe that

$$f(U) + f(V) = f(S^*) + f(S') > 0.$$ 

Now, since $V \subseteq S^*$, none of the precedence constraints we deleted during the relaxation of this problem end in $V$. Thus $V$ must be initial in the original input, and hence $f(V) \leq 0$. Similarly, $U$ must be initial, since any deleted precedence constraint must have started in $U$, and so $f(U) \leq 0$, a contradiction. 

We can now complete the proof of the Sydney Decomposition Lemma. View some optimal schedule as an alternation of blocks from $S^*$ and $N - S^*$. So we have

$$T_0, S_1, T_1, \ldots, S_k, T_k.$$
where all of these sets are disjoint, the union of all the \( S_i \) sets is \( S^* \), and \( T_0 \) and \( T_k \) might be empty. Now since this schedule is obviously feasible, and \( S^* \) is initial, we can consider using an interchange argument on these blocks of jobs. As when we considered interchanging individual jobs, we know that \( \rho \) for each block must be non-increasing on sequential blocks. In particular, we have that

\[
\rho^* \geq \rho(T_0) \geq \rho(S_1) \geq \rho(T_1) \geq \ldots \geq \rho(S_k) \geq \rho(T_k).
\]

Note that the first inequality follows since \( T_0 \) must itself be initial, and \( \rho^* \) is the largest \( \rho \) value of any initial set, even in the relaxed problem instance. We know that \( S_1 \cup S_2 \cup \ldots \cup S_{k-1} \) is initial, since \( S^* \) is initial and the given optimal schedule demonstrates that \( S_k \) can follow the rest of \( S^* \). Therefore \( \rho(S_1 \cup S_2 \cup \ldots \cup S_{k-1}) \leq \rho^* \). The following fact will allow us to complete the proof.

**Fact 0.4** If \( A \) and \( B \) are initial, then \( \rho(A \cup B) \) is a linear combination of \( \rho(A) \) and \( \rho(B) \).

This follows from the definition of \( \rho \) and the fact that the functions \( w(\cdot) \) and \( p(\cdot) \) are additive. In particular,

\[
\rho(A \cup B) = \frac{w(A \cup B)}{p(A \cup B)} = \frac{p(A)}{p(A \cup B)} \rho(A) + \frac{p(B)}{p(A \cup B)} \rho(B).
\]

Plugging \( S_k \) in for \( A \) and \( S^* - S_k \) in for \( B \) implies that \( \rho^* \) is a linear combination of \( \rho(S_k) \) and \( \rho(S_1 \cup S_2 \cup \ldots \cup S_{k-1}) \) The only way this is possible is if \( \rho(S_k) \geq \rho^* \). But this forces the \( \rho \) values of all blocks (except possibly \( T_k \)) to be equal. Therefore we can swap blocks so that all \( S_i \) blocks are before all \( T_i \) blocks with no change to the objective function.

**A New Linear Programming Formulation**

We now consider a weak linear ordering formulation of the problem.

\[
\delta_{jk} = \begin{cases} 
1 & j \text{ comes before } i \text{ in the schedule} \\
0 & \text{otherwise}
\end{cases}
\]

Given this notation, we can formulate the following linear program.

\[
\min \sum_{j=1}^{n} w_j p_j + \sum_{j \neq k} w_k p_j \delta_{jk} \\
\text{s.t.} \quad \delta_{jk} + \delta_{kj} = 1 \quad \forall j, k \\
\delta_{jk} \geq 0 \quad \forall j, k \\
\delta_{jk} = 1 \quad \forall j \rightarrow k \\
\delta_{ik} + \delta_{kj} \geq 1 \quad \forall k, i \rightarrow j
\]

It is an open problem as to whether or not integral solutions to this LP correspond to real solutions. What is known is that since each inequality in this LP has at most 2 variables, there is a half integer solution.

**Series Parallel Precedence Constraints**

We now introduce a class of precedence constraints, called series parallel constraints. We’ll consider \( 1|\text{sepa}| \sum w_j C_j \). Series parallel constraints are constructed inductively as follows. Any singleton job is series parallel. Then we have two build operations. Given two series parallel constraints \( G_1 \)
and $G_2$, the parallel operation is to form $G_1 \cup G_2$. In other words, there are no constraints between elements of $G_1$ and $G_2$. The series operations forms $G_1 \cup G_2 \cup \{g_1 \rightarrow g_2 \mid g_1 \in G_1, g_2 \in G_2\}$. In other words, we inherit the constraints of $G_1$ and $G_2$, but also insist that every element of $G_1$ come before every element of $G_2$. Series parallel constraints can be easily described with a binary tree, in which leaves correspond to jobs and internal nodes correspond to either series or parallel operations.

It turns out that there is a polytime algorithm for solving $1|\text{sepa}|\sum w_j C_j$. In particular, the weak linear ordering LP described above has an integer optimum corresponding to a real schedule for any series parallel input.

Finally, we consider another linear program formulation for series parallel precedence constraints.

$$\min \sum_{j=1}^{n} w_j C_j$$

s.t. $C_k \geq C_j + p_k \quad \forall j \rightarrow k$

$$\sum_{j \in S} p_j C_j \geq \frac{1}{2}p^2(S) + \frac{1}{2}p(S)^2 \quad \forall S \subseteq N$$

We have seen the last set of constraints before, and we can think of this as the parallel inequalities. As is, this LP yields non-schedules with series parallel precedence constraints. Consider adding the following definition for all jobs $j$

$$M_j = C_j - \frac{P_j}{2}$$

Intuitively $M_j$ should be the time at which job $j$ is half done. We can now add the following inequalities for all sets of jobs $A$ and $B$ that were merged ($B$ following $A$) by a series operations.

$$\frac{1}{p(B)} \sum_{j \in B} p_j M_j - \frac{1}{p(A)} \sum_{j \in A} p_j M_j \geq \frac{1}{2}p(A) + \frac{1}{2}p(B).$$

In other words, between the midpoint of all the jobs in $A$ and the midpoint of all the jobs in $B$, we had better take the time to process at least half the stuff in $A$ and half the stuff in $B$. It turns out that adding these constraints to the LP lets us find integral solutions corresponding to real schedules.