Homework 2

Due: May 21st, 2009

1. Suppose the optimal schedule S is not SPT. Then there must be a job k and a job h such that $p_k > p_h$ and the machine processes k before h. We may assume that k and h are consucutive in the schedule. Now consider the schedule S' obtained from S by swapping jobs k and h. Note that the completition times of all jobs $j \neq k, h$ is the same in S and S', that is $C_j^{S'} = C_j^S$ for all $j \neq k, h$. It is not hard to see that $C_k^{S'} = C_h^S$ and $C_h^{S'} = C_k^S - p_k + p_h < C_k^S$. Therefore

$$\sum_j f(C_j^S) - \sum_j f(C_j^{S'}) = f(C_h^S) - f(C_k^{S'}) + f(C_k^S) - f(C_h^{S'}) > 0,$$

contradicting the fact that S is an optimal schedule.

Applying SPT to the given problem we obtain the optimal schedule 35718246, of value $\sum_j f(C_j) = 1 + 3^2 + 5^2 + 8^2 + 11^2 + 15^2 + 20^2 + 26^2 = 1521$.

2. Running the SRPT algorithm we see that in this instance we do not need to use preemption and we process the jobs in the order 641235. As we do not use preemption in this schedule, the schedule obtained from CFP is the same as the one for SRPT. Note that every valid schedule for $(1 \mid r_j \mid \sum C_j)$ is also valid for $(1 \mid r_j, pmtn \mid \sum C_j)$. Therefore the optimal value for $(1 \mid r_j \mid \sum C_j)$ is at least the optimal value for $(1 \mid r_j, pmtn \mid \sum C_j)$. Thus the schedule found by the algorithm is optimal.

3. Let P be an optimal schedule for $(1 \mid r_j, pmtn \mid \sum C_j)$, of value OPT^P. If we show that $CFP \leq 1.5\ OPT^P$, then we will be done. With an interchange argument, it is easy to see that we may assume that with two jobs at most one job is preempted. Moreover, if an optimal schedule for $(1 \mid r_j, pmtn \mid \sum C_j)$ does not use preemption, then $CFP = OPT^P = OPT$ and we are done. So we may assume $r_1 = 0, r_2 > 0$ and SRPT produces a schedule in which we process job 1 until time $t = r_2$, then we process and complete job 2 and finally we complete job 1. Thus we have $OPT^P = r_2 + 2p_2 + p_1 \geq 2p_1$ and, by the SRPT rule, $p_2 \leq p_1 - r_2$. The corresponding CFP schedule will process job 2 first and then job 1, so $CFP = 2r_2 + 2p_2 + p_1 \leq 2(p_1 - p_2) + 2p_2 + p_1 = 3p_1 \leq 1.5(2p_1) \leq 1.5OPT^P$.

To construct the tight examples, consider the two job instance with job 1 having $p_1 = T$ and $r_1 = 0$, job 2 having $p_2 = 1$ and $r_2 = T - 2$. Now the optimum schedule is to process job 1 followed by 2 giving $OPT = 2T + 1$. On the other hand, the CFP algorithm will remain idle till time $T - 2$, process job 2 and then process job 1. Thus, $CFP = T - 1 + 2T - 1 = 3T - 2$. For any $\delta > 0$, we can find T large enough such that $(3T - 2)/(2T + 1) > (1.5 - \delta)$.

To construct an example which shows the tightness of $(2 - \delta)$ consider the following data generalizing the previous example.

Jobs	1	2	3	...	k
p_j	T	1	2	...	2
r_j	0	T-2	T-1	...	T-1

The optimum schedules jobs in order $\{1, 2, \ldots, k\}$ giving $OPT = T+T+1+T+3+\ldots+T+(2k+1) = kT+\Theta(k^2)$. Running CFP will process the jobs in order $\{2, 1, 3, 4, \ldots, k\}$ which leads to $CFP = T-1+2T-1+2T+1+\ldots(2T+2k-5) = 2Tk+\Theta(k^2)$. For any fixed δ it is possible to choose large enough k and T such that $CFP/OPT > 2-\delta$.

4. At the first LCL iteration we have $\sum p_j = 55$, so the minimum of $f_j(\sum_{k\in X} C_k)$ is 77, attained by job 2 and 7. We pick job 2, so we have $X = \{1, 3, 4, 5, 6, 7\}$ and $\sum_{k\in X} p_k = 47$. Now the minimum of $f_j(\sum_{k\in X} C_k)$ is attained by job 7. We repeat this process and we obtain the order $3, 5, 1, 6, 4, 7, 2$, with $f_{max} = 144$. An alternative solution is the order $3, 2, 1, 5, 6, 4, 7$ (obtained by choosing job 7 in the first iteration).

5. We only need to slightly modify the algorithm seen in class for $(1||f_{max})$. At each iteration we consider the set of jobs j such that j can be processed after all the jobs in $X \setminus \{j\}$. This corresponds to the nodes in D with no outgoing arc to a node in X. Hence the algorithm is as follows.

   - Initialize R to be the set of nodes in D with no outgoing arcs and $X = J$. Let S be an empty stack.
   - Find j in $X \cap R$ which minimizes
   $$j = \arg\min_{k\in X\cap R} f_j(\sum_{k\in X\cap R} p_k)$$
   - Append j to the front of S and let $X = X \setminus \{j\}$. Remove j from R and add to R all nodes in X with no outgoing arcs to nodes in X.

   If n is the number of jobs and m is the number of arcs in D, the running time of the algorithm is $O(n^2m)$. The proof of correctness is very similar to the proof for LCL for $(1||f_{max})$.

6. (a) In the best possible case we can fit all the items in the backpack. Hence the maximum profit is at most $\sum_j p_j$. This is at most nP_{max}.
   (b) From part (a), p should range from 1 to nP_{max}. Moreover, i indexes the items, so i should range from 1 to n. So the table has n^2P_{max} entries.
   (c) The optimum set of items for the problem is given by $T[n, \hat{p}]$, where $\hat{p}$ is the maximum p such that $T[n, p] \neq null$ and $t[n, p] \leq B$.
   (d) We have $T[0, 0] = \emptyset$ (so $t[0, 0] = 0$), while $T[0, p] = null$, $t[o, p] = \infty$ for every $p > 0$. Moreover $T[i, 0] = \emptyset$ and $t[i, 0] = 0$.
   (e) We have $T[i + 1, p]$ equal to:
      (i) $null$, if $\sum_{j=1}^{i+1} p_j < p$,
      (ii) $T[i, p]$, if $t[i, p] \neq \infty$ and $t[i, p] \leq t[i, p - p_{i+1}] + w_{i+1}$,
      (iii) $T[i, p - p_{i+1}] \cup \{i + 1\}$ otherwise.

      The first case occurs when with the items in $\{1, \ldots, i + 1\}$ it is not possible to obtain a profit p. Case (ii) occurs when the $(i + 1)$th item is not in $T[i + 1, p]$, and the last case occurs when item $(i + 1)$ is in $T[i + 1, p]$.