C&O 454 Scheduling — Spring 2013

Assignment 1

Due: Thu, May 30 in class *before lecture*

Write your name and ID# clearly, and underline your last name.

Unless otherwise stated, all algorithms should be accompanied with a proof of correctness and a brief analysis of running time.

Problem 1: LCL rule (Pinedo, Ex. 3.4, 3.5, pg. 63) **(15 marks)**

(a) Consider the following instance of $1||f_{\max}$.

Jobs	1	2	3	4	5	6	7
p_j	4	8	12	7	6	9	9
$f_j(x)$	$3x$	77	x^2	$1.5x$	$70+\sqrt{x}$	$1.6x$	$1.4x$

Find an optimal schedule by running the LCL (least-cost-last) algorithm, and determine its objective value. Show all steps, including the f_j values computed and the job selected in each iteration. (5 marks)

(b) Recall the following modification of LCL discussed in class for the problem $1|prec|f_{\max}$: given the current set J of jobs, let $L(J)$ be the jobs in J having no successors in J, i.e., $L(J) := \{j \in J : \nexists k \in J \text{ s.t. } j \to k\}$. Find a job $\ell \in L(J)$ such that $f_\ell(\sum_{j\in J} p_j) = \min_{k\in L(J)} f_k(\sum_{j\in J} p_j)$; schedule ℓ last and recurse on the remaining set $J \setminus \{\ell\}$ of jobs. Prove that this algorithm computes an optimal schedule for $1|prec|f_{\max}$. (5 marks)

(c) Run the (modified) LCL algorithm to find an optimal schedule for the instance of $1|prec|f_{\max}$ specified by the data given in part (a), and the following precedence constraints: $1 \to 7 \to 6$, $5 \to 7$, $5 \to 4$. Show all the steps of the algorithm, and specify the optimal value. (5 marks)

Problem 2: Running time analysis and $O(.)$-notation **(10 marks)**

Given two functions $f : \mathbb{R}_+ \mapsto \mathbb{R}_+$ and $g : \mathbb{R}_+ \mapsto \mathbb{R}_+$, recall that $f(n) = O\big(g(n)\big)$ if there exist constants $c > 0, n_0 \geq 0$ such that $f(n) \leq c \cdot g(n)$ for all $n \geq n_0$. We say that $f(n) = \Theta\big(g(n)\big)$ if $f(n) = O\big(g(n)\big)$ and $g(n) = O\big(f(n)\big)$. (For example, $2n + 5 = \Theta(n)$.)

Consider the following problem. You are given an array A containing n integers, $A[1], \ldots, A[n]$. You need to output an $n \times n$ array B where $B[i,j]$ (for $i \leq j$) is equal to $\max\{A[i], A[i+1], \ldots, A[j]\} - \min\{A[i], A[i+1], \ldots, A[j]\}$. (So $B[i,i] = 0$; the value of $B[i,j]$ for $i > j$ is left unspecified, and you may store anything in these entries.)

(a) Consider the following simple algorithm for the above problem.

For $i = 1, 2, \ldots, n$
  For $j = i, i+1, \ldots, n$
    Step through the array entries $A[i], A[i+1], \ldots, A[j]$ to find their maximum MAX
    Step through the array entries $A[i], A[i+1], \ldots, A[j]$ to find their minimum MIN
    Set $B[i,j] = MAX - MIN$.

Analyze the running time of this algorithm. That is, you should give a function $f(n)$ and show that the running time of the algorithm is $\Theta\big(f(n)\big)$. (5 marks)

(b) Design an algorithm to solve the above problem with an asymptotically better running time. That is, if the running time of your new algorithm is $O\big(g(n)\big)$, then it should be that $\lim_{n\to\infty} g(n)/f(n) = 0$. (This is often denoted by $g(n) = o\big(f(n)\big)$.) Show that your algorithm is correct and analyze its running time.

(5 marks)

<u>Problem 3:</u> $1|prec|f_{\max}$ and EDD rules for $1|r_j, pmtn|L_{\max}$ and $1|r_j, pmtn, prec|L_{\max}$ (25 marks)

In this problem, we will give a different algorithm for $1|prec|f_{\max}$ and see its applications to $1|r_j, pmtn|L_{\max}$ and $1|r_j, pmtn, prec|L_{\max}$ (that is, minimizing maximum lateness on a single machine in the presence of release dates, and possibly precedence constraints, while allowing preemption).

In answering a part, you may use the results of the previous parts even if you did not manage to solve them.

(a) In class, we argued that the LCL rule produces an optimal schedule for $1||f_{\max}$. We now describe a simple way of modifying an instance $\mathcal{I}$ of $1|prec|f_{\max}$ to obtain an instance $\mathcal{I}'$ of $1||f_{\max}$ so that the LCL rule when run on $\mathcal{I}'$ will *produce a schedule that is optimal also for* $\mathcal{I}$. (Thus, this gives an alternate algorithm for $1|prec|f_{\max}$.)

Let $D = (J, A)$, be the directed acyclic graph (DAG) representing the precedence constraints, where J is the set of jobs. Assume that $p_j > 0$, and f_j is a strictly increasing function for all $j \in J$. The modification will consist of modifying the f_j-cost functions for the jobs suitably.

Initialize the cost function f'_j for every job $j \in J$ to be f_j.
Repeat until $A = \emptyset$: for every arc $a = j \to k$ in A, where k has no successors (i.e., k has no outgoing arcs in A), define $f'_j(t) = \max\{f'_j(t), f'_k(t + p_k)\}$ for every time t; remove a from A.

The instance $\mathcal{I}'$ is defined by the new f'_j cost functions (and the same p_js). (Note that the f'_js are also strictly increasing.) Observe that the above procedure ensures that if $j \to k$ then $f'_j(t) > f'_k(t)$ for all times t.

Prove that any schedule S that is feasible for $\mathcal{I}$ has the same objective value for both $\mathcal{I}$ and $\mathcal{I}'$; that is, $\max_{j\in J} f_j(C_j) = \max_{j\in J} f'_j(C_j)$. Next, prove that executing the LCL rule on $\mathcal{I}'$ produces a feasible schedule for $\mathcal{I}$. Thus, argue that running LCL on $\mathcal{I}'$ produces an optimal schedule for $\mathcal{I}$. (6 marks)

(b) Now consider the problem $1|prec|L_{\max}$, which is a special case of $1|prec|f_{\max}$ (where $f_j(t) = t - d_j$ is a strictly increasing function). Given an instance $\mathcal{I}$ of $1|prec|L_{\max}$, give a simplified description of the execution of the procedure in part (a) on $\mathcal{I}$. That is, you should interpret the procedure in part (a) as one that modifies the due dates of the jobs suitably (so that executing EDD on the resulting instance $\mathcal{I}'$ of $1||L_{\max}$ yields an optimal schedule for $\mathcal{I}$).

(It might be useful to see first how the procedure in part (a) operates on small examples, such as the following instance with 4 jobs: we have $d_1 = 1,\ d_2 = d_3 = 3,\ d_4 = 4$, $p_1 = p_4 = 1,\ p_2 = p_3 = 2$, and the precedence constraints are: $1 \to 2 \to 3$, $2 \to 4$. Note that an instance of $1||L_{\max}$ or $1|prec|L_{\max}$ may have due dates that are positive or negative, or zero.) (4 marks)

(c) Now consider the problem $1|r_j, pmtn|L_{\max}$. Devise an algorithm to solve this problem. Prove that your algorithm always returns an optimum schedule and analyze its running time. (9 marks)

(**Hint:** Consider modifying EDD in a fashion similar to the way we modified SPT to obtain the SRPT rule. Use an interchange argument to prove the optimality of the resulting preemptive EDD rule.)

(d) Now combine the results of parts (b) and (c) to devise an algorithm that finds an optimum schedule for $1|r_j, pmtn, prec|L_{\max}$. As always, prove the correctness of your algorithm and analyze its running time.

(6 marks)

Problem 4: $1|r_j, pmtn|\sum_j w_j C_j$ **(5 marks)**

Consider the following extension of the WSPT rule for the problem of minimizing total weighted completion time with release dates and allowing for preemption. At each point of time t, schedule the job available with highest weight/(remaining processing time) ratio, preempting a job if a job with higher ratio is released. Call this the weighted-shortest-remaining-processing-time (WSRPT) rule.

Give an example to show that the WSRPT rule does not always produce an optimal schedule.

(**Hint:** You need at least 3 jobs, and there is an example with 3 jobs that only uses weights 1 and $2 + \epsilon$, where $\epsilon > 0$ is very small.)