Answers to Exam Sequencing and scheduling, 2P450,
May 4 2005, 09:00-12:00

Question 1.
Consider the *head-body-tail*-problem in which job J_j has release date $r_j \geq 0$, processing time $p_j > 0$, and 'delivery time' $q_j \geq 0$. Interpreting $d_j = -q_j$ as a deadline, we consider the problem of scheduling jobs on a single machine, *minimizing lateness.* In fact, the lateness is the first moment that all jobs have been delivered.

Note that for an arbitrary subset $S \subseteq \{1, \dots, n\}$ and for an arbitrary schedule σ (in which C_j^σ denotes the completion time of job J_j), the lateness $L_{\max}(\sigma)$ $(= \max_j[C_j^\sigma + q_j])$ is at least $f(S) := (\sum_{j \in S} p_j) + \min_{j \in S} r_j + \min_{j \in S} q_j$, even if preemption is allowed.

(a) Formulate the pre-emptive EDD rule.
At any time $t \geq 0$, schedule of all available jobs (not completed yet, and $r_j \leq t$), the one with earliest due date d_j, or in this case, with highest delivery time q_j. In case a job arrives with lower due date than the one currently processed, preempt processing and star work on the more important job.

(b) Give a tight upper bound on the number of pre-emptions in a schedule obtained by the pre-emptive EDD rule.
As preemption only occurs at the arrival time of a job, and not at the very first arrival time, the number of preemptions is never more than $n-1$, and this bound is best possible.

(c) Give an optimal schedule in case *preemption is allowed*, for the following instance:

j	1	2	3	4	5
r_j	0	2	5	7	8
p_j	5	5	4	3	3
q_j	3	5	9	10	11

What is the lateness of this schedule? For which subset S is $f(S)$ equal to the lateness?
The pEDD schedule is as follows: $J_1 : [0, 2)$, $J_2 : [2, 5)$, $J_3 : [5, 7)$, $J_4 : [7, 8)$, $J_5 : [8, 11](L_5 = 22)$, $J_4 : [11, 13], (L_4 = 23)$, $J_3 : [13, 15], (L_3 = 24)$, $J_2 : [15, 17], (L_2 = 22)$, $J_1 : [17, 20], (L_1 = 23)$.
Its lateness is $L_{\max} = L_3 = 24$, *which equals* $f(S)$ *for* $S = \{3, 4, 5\}$.

(d) Give an optimal schedule, in the case that preemption is not allowed.
The non-preemptive EDD rule gives the following schedule: $J_1 : [0, 5](L_1 = 8)$, $J_3 : [5, 9](L_3 = 18)$, $J_5 : [9, 12](L_5 = 23)$, $J_4 : [12, 15](L_4 = 25)$, $J_2 : [15, 20](L_2 = 25)$.
Give convincing arguments, why your schedule is optimal.
An optimal schedule has integral completion times, so if it has lower lateness it must be 24. Then jobs 3,4,5 have to be scheduled inside interval $[5, 15]$, *and job 3 has to be the first and the last of these three. This is impossible.*

Question 2.

(a) Formulate the BALANCING-problem (which is an NP-complete problem used in class as an example of an intrinsically 'hard' problem).
Given positive integral numbers $a_1, \dots, a_n$, and b such that $\sum_{i=1}^n a_i = 2b$, the questions is whether there exists a set $S \subset \{1, \dots, n\}$, such that $\sum_{i \in S} a_i = b$.

(b) Consider the head-body-tail problem (see question 1). Prove that the head-body-tail problem is also NP-complete ('hard'), by showing that one of the problems is a special case of the other problem.
Given a BALANCE instance, we turn this into a head-body-tail problem by defining $n+1$ *jobs, where* J_i *had* $r_i = q_i = 0$*, and* $p_i = a_i$*, for* $i = 1, \ldots, n$*, and a special job* j_0 *has* $r_0 = q_0 = b$*, and* $p_0 = 1$*. A schedule of lateness (or latest delivery) equal to* $2b+1$ *exists if and only if there is a subset* $S \subset \{1, \ldots, n\}$ *with* $\sum_{i \in S} a_i = b$*.*

Question 3.
For a single machine we would like to schedule jobs $J_1, \ldots, J_n$, with processing times $p_1, \ldots, p_n$ and weights $w_1, \ldots, w_n$, respectively, so as to minimize the total weighted completion time.

(a) Describe the *ratio rule* for obtaining a good schedule.
Schedule jobs in order of non-decreasing $\frac{p_i}{w_i}$.

(b) Does this rule always give an optimal schedule? If yes, prove, if no, give an example where it does not work optimally.
Yes, this rule always gives a schedule with lowest value of $\sum_j w_j C_j$*. Proof: consider an optimal schedule. It has no idle time. Consider the first two consecutive jobs that are not in ratio rule order. Suppose these are* J_j *and* J_k*. Interchange the two jobs. The other jobs keep their original completion times,* C_k *drops by* p_j*,* C_j *increases by* p_k*, so the total weighted completion time drops by* $w_k p_j - w_j p_k$ *which is positive by assumption.*

(c) Consider an instance I_1 of the problem, with processing times $p_1^1, \ldots, p_n^1$ and weights $w_1^1, \ldots, w_n^1$, and another instance I_2, with the same number of jobs, but with processing times $p_j^2 := w_j^1$, and weights $w_j^2 := p_j^1$ (hence processing times weights switched). Let σ be an arbitrary schedule for I_1 without any idle time. Here σ is not necessarily optimal. Schedule the jobs for I_2 in the *opposite order*.
Prove that the two schedules have the same objective value.
Let the schedule for the new instance be denoted by τ*. Assume the jobs are labeled in the order they are processed in* σ*. Then the value of* σ *is* $\sum_{j=1}^n w_j C_j^\sigma = \sum_{j=1}^n w_j (\sum_{k=1}^j p_k) = \sum_{j,k:k \leq j} w_j p_k$*.*
On the other hand, the value for schedule τ *equals* $\sum_{k=1}^n p_k C_k^\tau = \sum_{k=1}^n p_k (\sum_{j=k}^n w_j) = \sum_{j,k:j \geq k} w_j p_k$*, hence these two values are the same.*

Question 4.
In the flow shop problem on m machines, each job is processed by machines $M_1, M_2, \ldots, M_m$, in this order. A schedule is called a *permutation*-schedule if the jobs are scheduled on each machine in the same order. The usual objective is to minimize makespan.

(a) Prove that when looking for a schedule of minimum length for the **two**-machine flow shop we only have to consider permutation schedules.
Consider any schedule. Without loss of generality, jobs on machine 1 are scheduled as early as possible, so there is no idle time on machine 1. Consider the first two consecutive jobs on machine 1 in order different from their order on machine 2, say J_j *and* J_k*. Let* t *be the finish time of job* J_k *on machine 1, then on machine 2, both jobs are scheduled not earlier than* t*, and on machine 1 both are processed not later than* t*. Hence we can interchange the jobs on machine 1 without increasing the makespan of the schedule.*

(b) Describe an efficient algorithm to find a shortest schedule for the two-machine flow-shop.
Since we only have to establish an ordering of the jobs (as a permutation schedule is among the optimal schedules) we can do this by selecting the operation with shortest processing time. If it is an operation on the first machine, schedule this job first, if it is an operation on the second machine schedule it last. Forget about this job and repeat the argument to find the order of the remaining jobs.
An alternative rule giving the same schedule: first take all jobs with processing time on machine 1 smaller than on machine 2, process them in order of machine 1 processing time, next process remaining jobs in decreasing order of machine 2 processing time.

(c) Extend your algorithm to the three-machine flow-shop. Does it still produce the optimal schedule? If yes, prove, if no, give an example where it does not work optimally.
For three machines a permutation schedule is also among the optimal schedules. The above rule can be 'extended' to the three machine case by inspecting only operations on either the last or the first machine. Another extension could be to work with processing times $(a_j + b_j)$ *and* $(b_j + c_j)$. *(However, NO simple rule can ever guarantee optimality, as the flow shop on three machines is NP-complete.) The first extension works out wrongly on the following instance:* $J_1 : 1, 4, 3; J_2 : 2, 1, 4$. *The alternative mentioned is spoilt by instance:* $J_1 : 2, 3, 3; J_2 : 1, 5, 3$.