Exam Sequencing and scheduling, 2P450,
March 18 2005, 14:00-17:00

This test contains four questions on two pages, for which a total of 40 points can be scored. The final grade is obtained by dividing the amount of points by four. Questions may be answered in English and/or in Dutch. Please explain your answers.

Each good answer scores 3 points except for sub-questions 2(a) and 2(c), which give two points each.

Question 1.

(a) Formulate the BALANCING-problem (which is an NP-complete problem used in class as an example of an intrinsically 'hard' problem).
Given n positive integers $a_i, i = 1, \ldots, n$, with $\sum_i a_i = 2b$, is there a subset $S \subset \{1, \ldots, n\}$, such that $\sum_{i \in S} a_i = b$?

(b) Formulate the job-shop problem.
We are given m machines $M_1, \ldots, M_m$, and n jobs $J_1, \ldots, J_n$, where job J_j is a sequence of tasks $O_{j1}, O_{j2}, \ldots, O_{jk_j}$. Task O_{ji} has to be processed by (fixed) machine $M_{ji} \in \{M_1, \ldots, M_m\}$, for an uninterrupted time period of length $p_{ji} \in \{0, 1, 2, \ldots\}$. A machine can handle one job at a time and the tasks per job have to be processed in the given order. Is there a schedule with makespan L, or what is the minimum makespan of any schedule?

(c) Prove that the job-shop problem is also NP-complete ('hard'), by by showing that one of the problems is a special case of the other problem.
The point is to see that every balance problem can be viewed as a job-shop problem: given the balance problem with numbers a_i, create the following job shop problem: we have $n+1$ jobs, and 3 machines. For $1 \leq j \leq n$, job J_j visits machines M_1, M_2, M_3 for a time period of 0, a_j, and 0 time units, respectively, in this order. The dummy job J_{n+1} consists of visits to these three machines for periods of b, 1, and b time units, respectively. There is a subset of size b if and only if there exists a schedule of makespan $2b+1$. If you dislike jobs of length 0, they may be given a small length $\epsilon = \frac{1}{4n}$. Then a schedule with makespan at most $2b + 1.5$ corresponds to a subset of size b.

Question 2.

(a) For minimization of the makespan (length) of a schedule for n jobs $J_1, \ldots, J_n$ on parallel identical machines $M_1, \ldots, M_m$ we can use the so-called LPT-rule. Explain this rule.
LPT stands for Longest Process Time. Order jobs by non-increasing length, and assign them one by one, in this order to the machines. A job is assigned to that machine that has lowest current load.

(b) Does this LPT-rule always provide an optimal schedule? Is yes, prove, if no, give an example where it does not work optimally.
NO, it does not always give an optimal solution. For example, consider a two machine problem with 5 jobs of length $3, 3, 2, 2, 2$. LPT gives a schedule of length 7, whereas the optimal schedule has length 6.

(c) In the same context (n jobs $J_1, \ldots, J_n$ scheduled on parallel identical machines $M_1, \ldots, M_m$) we may use the SPT-rule. Explain what it is, and for what objective it is used (in other

words: what do you want to optimize when using the SPT-rule?).
SPT stands for Shortest Process Time. Order jobs by non-decreasing length, and assign them one by one, in this order to the machines. A job is assigned to that machine that has lowest current load. The rule is used for minimizing average or total completion time ($\sum_j C_j$).

(d) Does the SPT-rule always give an optimal schedule? Is yes, prove, if no, give an example where it does not work optimally.
YES, the SPT rule always gives an optimal schedule. Proof: suppose jobs are already in SPT order. Let there be n *jobs,* m *machines. Without loss of generality, job* J_k *is assigned to machine* M_i*, with* $i \equiv k$*, modulo* m*. After assigning* J_k*, machine* i *with* $i \equiv k$ modulo m*, has the highest load. Next highest are* $M_{i-1}, M_{i-2}, \ldots, M_{i+1-m}$*, and the loads differ no more than* p_k*. Here machine subscripts should be interpreted modulo* m*. In the total sum of completion times, the length of* J_n *contributes once, and similar for jobs* $J_{n-1}, \ldots, J_{n-m+1}$*. The* m *jobs prior to these* $(J_{n-m}, \ldots, J_{n-2m+1})$ *contribute to the sum of completion times with 2 times their job length, etcetera. As in each schedule there are* m *jobs contributing with their processing time (at least) once,* m *are contributing at least twice, etcetera, the schedule found indeed has minimum total completion time.*

(e) We want to minimize the schedule for n jobs $J_1, \ldots, J_n$, with processing times $p_1, \ldots, p_n$, respectively, on parallel identical machines $M_1, \ldots, M_m$) and we allow *preemption*. Give a closed form expression for the optimal makespan in terms of the processing times p_j.
By McNaughton's rule we find a schedule of length $\max\{p_{\max}, \sum_j p_j/m\}$.

Question 3.

(a) In the flow shop problem on three machines, each job is processed by machines M_1, M_2, M_3, in this order. Give an example of a *permutation*-schedule and one of a *non-permutation*-schedule for a three job example with processing times given in the following table

	machine		
job	M_1	M_2	M_3
J_1	2	4	1
J_2	1	3	5
J_3	4	1	3

Permutation schedule: on each machine process jobs in order J_1, J_2, J_3*; Non-permutation schedule: on machines* M_1, M_2*, process jobs in order* J_1, J_2, J_3*, whereas on machine* M_3*, process jobs in order* J_3, J_2, J_1.

(b) Prove that when looking for a schedule of minimum length for the **three**-machine flow shop we only have to consider permutation schedules.
By an exchange argument one can show that for any flow shop schedule of minimum makespan, jobs can be processed on the FIRST machine in the same order as on the SECOND machine (without increasing makespan). By symmetry, jobs on the LAST machine can be processed in the order they have on the LAST BUT ONE machine. As a result, for flow shop problems with two or three machines we may restrict ourselves to investigating schedules in which the jobs are processed on each machine in the same order (permutation schedule).

(c) Show an example of a **four**-machine flow shop for which the optimal schedule is not a permutation schedule
Take two jobs, J_1 with processing times 4,4,4,4, and J_2 with processing times 4,1,1,4. Any schedule starting or ending with J_2 has length at least 4 (J_2) plus 16 (length J_1) is 20. Scheduling on $M_1 : J_1, J_2$, $M_2 : J_1, J_2$, $M_3 : J_2, J_1$, $M_4 : J_2, J_1$, leads to schedule of length 18.

Question 4.

(a) 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.*

Prove that for an arbitrary subset $S \subseteq \{1, \ldots, n\}$ and for an arbitrary schedule σ (in which C_j^σ denotes the completion time of job J_j), the lateness $(\max_j[C_j^\sigma + q_j])$ 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.
Consider a (possibly preemptive) schedule σ and a subset S of jobs. Let J_l be the job in S that is completed last in schedule σ. Let F be the first moment in schedule σ a job from S starts being processed. Then $F \geq \min_{j \in S} r_j$. As all jobs in S are processed in between F and C_l, we have $C_l - F \geq \sum_{j \in S} p_j$. The lateness of schedule σ equals $\max_j(C_j - d_j) = \max_j(C_j + q_j) \geq C_l + q_l$. This in turn is at least $C_l + \min_{j \in S} q_j = F + (C_l - F) + \min_{j \in S} q_j \geq \min_{j \in S} r_j + (\sum_{j \in S} p_j) + \min_{j \in S} q_j = f(S)$.

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

j	1	2	3	4	5
r_j	0	3	6	7	11
p_j	5	4	2	3	3
q_j	4	6	10	7	5

What is the lateness of this schedule? For which subset S is $f(S)$ equal to the lateness?
The optimal preemptive schedule obtained by the preempted EDD-rule is
$J_1[0, 3]; J_2[3, 6]; J_3[6, 8]; J_4[8, 11]; J_2[11, 12]; J_5[12, 15]; J_1[15, 17]$.
The lateness is $\max\{L_1, \ldots, L_5\} = \max\{17+4, 12+6, 8+10, 11+7, 15+5\} = L_1 = 21$, *and is equal to $f(S)$ for $S = \{1, 2, 3, 4, 5\}$.*

(c) Give an optimal schedule, in the case that preemption is not allowed.
Process jobs in order $1[0, 5], 2[5, 9], 3[9, 11], 4[11, 14], 5[14, 17]$. *The lateness* $= 22 = L_5$. *To see that it cannot be shorter one should realize that a schedule cannot have J_1 both as first and as last job. A schedule in which J_1 is not last has lateness at least* $0 + 17 + \min\{6, 10, 7, 11\} = 22$. *A schedule in which J_1 is not first has lateness at least* $\min\{3, 6, 7, 11\} + 17 + 4 = 24$.