Exam Sequencing and scheduling, 2P450,
January 23, 2015, 13:30-16:30

This test contains four questions on two pages, with 10 sub-items, for which a total of 50 points can be scored. Each good answer scores 5 points. The final grade is obtained by dividing the amount of points by five. Questions may be answered in English and/or in Dutch. Please explain your answers. You are NOT allowed to use the handouts distributed in class/or on the web.

Question 1. Consider the problem of minimizing makespan on m parallel identical machines $\mathrm{P}m||C_{\max}$.

(a) Describe the LPT-rule and state its worst-case ratio α (in terms of m). *Sort the list of jobs by non-increasing process time; in order of the sorted list append a job to the machine with lowest current load. The worst case ratio, depending on m, is $\alpha_m = \frac{4}{3} - \frac{1}{3m}$.*

(b) Sketch the proof that for any instance I the LPT-schedule has a makespan $C^{\mathrm{LPT}}_{\max}(I)$ at most $\alpha C^{\mathrm{OPT}}_{\max}(I)$. *Assume jobs are already LPT-sorted: $p_1 \geq p_2 \geq \ldots \geq p_n$. Let I_k denote the instance restricted to the first k jobs. Consider the LPT-schedule; let ℓ be the job with $C^{LPT}_{\ell}(I_n) = C^{LPT}_{\max}(I_n)$. At the start time S of ℓ all machines have been busy from time zero onward, hence $mS \leq \sum_{j<\ell} p_j$.*

A lemma told in class states that if an optimal schedule exists with at most two jobs per machine (*), then LPT will provide an optimal schedule. In case (*) applies to I_ℓ, evidently $C^{LPT}_{\max}(I_n) = C^{LPT}_{\max}(I_\ell) = C^{OPT}_{\max}(I_\ell) \leq C^{OPT}_{\max}(I_n) \leq \alpha_m C^{OPT}_{\max}(I_n)$.

Otherwise, we know that each optimal schedule for I_ℓ contains at least one machine with three or more jobs, hence $C^{OPT}_{\max}(I_\ell) \geq 3p_\ell$. In this case we find: $C^{LPT}_{\max}(I_n) = C^{LPT}_{\max}(I_\ell) = S + p_\ell \leq \sum_{j<\ell} p_j/m + p_\ell = \sum_{j\leq\ell} p_j/m + (1-\frac{1}{m})p_\ell \leq C^{OPT}_{\max}(I_\ell) + (1-\frac{1}{m})\frac{1}{3}C^{OPT}_{\max}(I_\ell) = \alpha_m C^{OPT}_{\max}(I_\ell) \leq \alpha_m C^{OPT}_{\max}(I_n)$.

(c) Provide for each m an instance I_m for which $C^{\mathrm{LPT}}_{\max}(I_m)$ **equals** $\alpha C^{\mathrm{OPT}}_{\max}(I_m)$. *Consider for I_m an instance with $2m+1$ jobs, three with process time m (jobs $2m+1, 2m$ and $2m-1$), and further $p_{2i-1} = p_{2i} = 2m-i$, for $i = 1, \ldots, m-1$. The LPT rule yields a schedule of length $4m-1$, where one machine has three jobs with total load $4m-1$, and all other machines each have two jobs with load $3m-1$. An optimal schedule has length $3m$, and has one machine containing the three jobs of length m, whereas the other machines pair up jobs in pairs of total length $3m = (2m-i)+(m+i)$.*

Question 2. Consider the problem $1|r_j|\hat{L}_{\max}$ of minimizing maximum *delivery time* of jobs scheduled on a single machine under release dates (also known as *head-body-tail*-problem). Here the delivery time is given als $\hat{L}_j := C_j + q_j$. In particular consider the instance given by

jobs j	1	2	3	4	5	6
p_j	8	3	4	3	4	4
r_j	0	6	16	19	12	10
q_j	1	2	3	4	5	6

(a) Formulate and apply an appropriate rule ρ for solving the PREEMPTIVE case. Compute the optimal *preemptive* schedule for the instance given above, and list the results in a table with the following entries:

jobs j	1	2	3	4	5	6
C_j^ρ						
$\hat{L}_j^\rho$						

The general rule is an adaptation of the preemptive EDD rule: at each point in time t, process the (remainder of the) available, unfinished job that has the longest tail q_j. If a job with higher tail arrives at time t preempt current processing. In the given instance, job 1 will be preempted twice (at $t = 5$, by job 2; and $t = 10$, by job 6 and job 3 will be preempted at time $t = 19$ by job 4. The resulting completion times and latenesses are given in

jobs j	1	2	3	4	5	6
C_j^ρ	26	9	25	22	18	14
$\hat{L}_j^\rho$	27	11	28	26	23	20

The schedule has maximum delivery time $L_{\max} = L_3 = 28$.

(b) Verify optimality of the preemptive schedule by providing a *compact proof of optimality* in the form of a subset $S \subseteq \mathcal{J}$, for which $\min_{j\in S} r_j + p(S) + \min_{j\in S} q_j = \hat{L}^\rho_{\max}$. Explain how you found this set S. *The set $S := \{3, 4, 5, 6\}$ has the property that $r_{\min}(S) + p(S) + q_{\min} = 10 + 15 + 3 = 28 = L^{pEDD}_{\max}$. The set was found as follows: the critical job is job 3, finishing at $C_3 = 25$. Between its release date $r_3 = 16$ and C_3 jobs $4, 5$ and 3 are being processed (all with tails at least q_3). Job 5 was released earlier; between $r_5 = 12$ and C_3 jobs $3, 4, 5$ and 6 were processed. Job 6 was released even earlier, and has $q_6 \geq q_5 \geq q_3$. Between $r_6 = 10$ and C_3 (only) jobs $3, 4, 5, 6$ have been processed; they all have $r_j \geq r_6$ and $q_j \geq q_3$. There is no idle time; so $S = \{3, 4, 5, 6\}$.*

Question 3. Consider the problem $Q3||\sum_j C_j$ of minimizing total completion time, on three parallel machines with distinct speeds $\sigma_1, \sigma_2, \sigma_3$.

(a) Describe for the general case, how to find the optimal solution.

*Consider the set of ratios $\frac{k}{\sigma_i}$, over all $k = 1, 2, \ldots, n$, and $i = 1, \ldots, m$; select the n smallest elements (duplicate values are allowed). Here k denotes a position of a job on a machine, counting **backwards**. Sort jobs from big to small, and match the ℓ-th biggest job j_ℓ to the ℓ-th lowest ratio $\frac{k_\ell}{c_{i_\ell}}$, which means schedule job j_ℓ as the k_ℓ latest job on machine i_ℓ. The rationale behind this rule is that job j scheduled at machine i on the k-latest position will contribute kp_j/σ_i to the objective $\sum_j C_j$.*

(b) In particular we may consider the instance given by the following parameters

jobs j	1	2	3	4	5	6	7
p_j	6	8	5	7	9	4	10

machines i	1	2	3
σ_i	2	3	4

Apply the method described in (a) to find the optimal schedule for this instance. *Note that the nine smallest ratios are:* $\{\frac{1}{4}, \frac{1}{3}, \frac{1}{2}, \frac{2}{4}, \frac{2}{3}, \frac{3}{4}, \frac{2}{2}, \frac{3}{3}, \frac{4}{4}\}$. *In fact different orderings (*$\frac{1}{2}$ *and* $\frac{2}{4}$ *reversed, for instance) may yield a total of 6 different optimal schedules, one of which is: machine 1: jobs 6, 2; machine 2: jobs 1,5; machine 3: jobs 3,4,7.*

(c) If machine i ($i = 1, 2, 3$) is **not able** to handle jobs j with $j = 0$ modulo $i + 1$ (that is, if $i + 1$ divides j), what would then be an appropriate rule or method for the general case? *In the case that not every machine i can process each job j (in a time p_j/σ_i) we are dealing with the machine setting of* ***unrelated*** *machines $Rm||\sum_j C_j$ rather than uniform machines $Qm||\sum_j C_j$). To match the job j with the proper position k on machine i we have to solve a* ***weighted bipartite matching*** *problem with a connection between job j and position (i, k) with weight $p_j k/\sigma_i$ only if j can be processed on i.*

Question 4. Consider the problem of scheduling n jobs on m machines, where each job consists of m subtasks, and where for each job j its i-th sub-task has to be processed on machine i. Here the processing time required is p_{ij}. We consider as objective function $\sum_j C_{mj}$, where C_{mj} denotes the completion time of the last subtask of job j (hence on machine m).

(a) Prove or disprove that there always exists an optimal schedule in which the jobs are processed on machine 2 in the same order as on machine 1. *The claim is true; without loss of generality jobs on machine 1 are scheduled actively, without idle time. Now consider an optimal schedule in which the job order on machine 1 is not equal to the order on machine 2. Then there is a pair of jobs j, k with j and k consecutively processed on machine 1, that is $C_{1j} = S_{1k} = C_{1k} - p_{1k}$, whereas on machine 2 we have $C_{2k} < C_{2j}$. Note that $C_{1k} \leq S_{2k} = C_{2k} - p_{2k} \leq S_{2j} - p_{2k}$, hence we can interchange on machine 1, jobs j and k. This will not interfere with feasibility on machines 1 and 2, nor with the objective value of the schedule. The interchange does decrease the number of dissimilarities between the orders on machine 1 and 2. It follows that there exists also an optimal schedule with these orders being equal.*

(b) Prove or disprove that there always exists an optimal schedule in which the jobs are processed on machine m in the same order as on machine $m - 1$. *It looks as if flipping the schedule upside down the same argument will hold again. HOWEVER, the objective is $\sum_j C_{mj}$, so modifying the order on machine m between consecutive jobs j and k (scheduled differently on machine $m - 1$) may influence the value of the objective function. MOREOVER, it does not make sense to assume that the jobs on machine m are scheduled AS LATE AS POSSIBLE and without idle time, because that would also deteriorate the object value of the 'optimal' schedule.*

A particular counter-example to the claim is the following

job\mach	1	2	3
A	10	10	10
B	1	10	90
C	10	100	1

The permutation schedule with lowest $\sum_j C_{3j}$ is ABC, with $\sum_j C_{3j} = 281$, whereas the schedule with job order BAC on machines 1 and 2; and order ABC on machine 3 yields $\sum_j C_{3j} = 274$.