Name: .... ID: .... University: ....

Exam for LNMB/Mastermath Course on Scheduling

22 May 2017

This exam consists of:

- **12 pages.**
- **6 questions.**
- You can obtain a total of 100 points. Your exam grade will be the points you obtained divided by 10.

You are allowed to make use of one Din-A4 paper with handwritten notes (on both sides) and a non-programmable calculator. You can answer the questions in Dutch or English.

Please write your answers, short but precise, in the space provided for it, directly after the question. If you need more space, please ask for another sheet for the specific question.
Do not forget to write your name, ID number and university on each page.

When a proof is asked, please provide a mathematically sound proof, short but precise. Unless stated otherwise, you are always expected to (briefly) explain your answer.
In case the objective function is not explicitely specified, it is supposed to be a regular objective function, unless stated otherwise.

Exercise 1 (15 points)**.**
Consider the problem $1 \mid \mid \sum w_j T_j$, where T_j denotes the *tardiness* of job J_j; we have that $T_j = \max\{C_j - d_j, 0\}$. Formulate a polynomial-time algorithm that solves this problem to optimality, or show that this problem is $\mathcal{NP}$-hard. In the latter case, you can use that the PARTITION problem is $\mathcal{NP}$-complete. The PARTITION problem is defined as follows:

Given a set of n nonnegative integers $a_1, \ldots, a_n$, does there exist a subset S of the index set $\{1, \ldots, n\}$ such that

$$\sum_{j \in S} a_j = \sum_{j=1}^{n} a_j/2 \equiv A?$$

Length **indication**: 500 words.

Continue on next page.

Continue your answer of Exercise 1.

Exercise 2 (15 points)**.**
Consider the problem $1\,|\,\bar{d}_j, prec\,|\,f_{\max}$. The maximum cost $f_{\max}$ is defined as $\max_j f_j(C_j)$, where $f_j(t)$ is the cost function of job J_j; it is non-decreasing in t. Formulate a polynomial-time algorithm that solves this problem to optimality, and prove its correctness.
Length **indication**: 350 words.

This page is intentionally left blank.

Exercise 3 (20 points)**.**
One possible solution approach to solve the job shop scheduling problem $J||C_{\max}$ is *Constraint Satisfaction*. The idea is to put an upper bound y on $C_{\max}$ and then show that there is no feasible schedule with makespan $\leq y$, after which y is increased. We will look at the feasibility problem resulting from the constraint $C_{\max} \leq y$. To obtain useful information we further look at the relaxation of the job shop problem to a single-machine scheduling problem. We select one *bottleneck* machine, say M_i. All machines other than M_i get infinite capacity, that is, they can process all available jobs at the same time. Now consider any operation on M_i, suppose this is O_k, which belongs to job J_j. Since operations within the same job are not allowed to overlap, and because of the infinite capacity of the machines other than M_i, we know that O_k will become available at M_i at time r_k, which is equal to the total processing time of all predecessors of O_k in job J_j. Similarly, after M_i has completed O_k, it requires a time q_k, which is equal to the total processing time of the successors of O_k in J_j, to finish job J_j. Since each job must be finished at time y, this implies that M_i must have completed O_k at time $y - q_k \equiv \bar{d}_k$. In this way, after this relaxation, we have to decide whether there exists a feasible schedule for the operations on M_i such that the release dates r_k and deadlines $\bar{d}_k$ are obeyed; keep in mind that M_i can only handle one operation at a time. From now on we assume that we are looking at this single machine scheduling problem and that the jobs refer to the operations that have to be executed on this machine. We assume that each machine has to process at most one operation of each job.
For any subset S of the set containing the indices of all jobs we define

$$r(S) = \min_{j \in S} r_j; \qquad p(S) = \sum_{j \in S} p_j; \qquad \bar{d}(S) = \max_{j \in S} \bar{d}_j.$$

(a) Suppose that for some set of indices S and job J_0 with $0 \notin S$ we have that

$$r(S) + p(S \cup 0) > \bar{d}(S \cup 0).$$

What conclusion can you draw then? Proof your answer.

(b) Give an algorithm with minimum running time to find the set S that maximizes $f(S)$, which is defined as

$$f(S) \equiv r(S) + p(S) - \bar{d}(S).$$

You do not have to show its correctness.

(c) Suppose that for a given job J_0 you want to find a set S to derive the type of result asked for in the situation of part (a). Indicate how you can use the algorithm of (b) to do so (if you did not solve (b), then just assume that Algorithm ALGO is the desired algorithm).

Length **indication**: 300 words.

Continue on next page.

Continue your answer of Exercise 3.

Exercise 4 (15 points)**.**
Consider the problem Pm | | $\sum w_jC_j$, which is known to be $\mathcal{NP}$-hard. We want to solve this problem using branch-and-price (a variant of branch-and-bound in which the lower bound is determined by solving the LP-relaxation using column generation). To that end, we formulate the ILP-formulation of the problem using *single-machine schedules*. A single-machine schedule corresponds to a subset of jobs that will be put on one (arbitrary) machine; each feasible schedule can be decomposed into m single-machine schedules that together contain all jobs. Given a single-machine schedule s, we use the binary decision variable x_s to indicate whether s is chosen (then $x_s = 1$) or not (then $x_s = 0$). Let S denote the set of all possible single-machine schedules; let c_s denote the total weighted completion time of all jobs included in the single-machine schedule s; and let a_{js} be a parameter that has value 1 if job J_j is included in the single-machine schedule s, and 0, otherwise. Then we can formulate the problem as an ILP as follows:

$$\min \sum_{s\in S} c_s x_s$$

subject to

$$\sum_{s\in S} a_{js}x_s = 1 \quad \forall\ j = 1,\dots,n$$

$$\sum_{s\in S} x_s = m$$

$$x_s \in \{0,1\} \quad \forall\ s \in S$$

(a) Prove that knowing the set of selected single-machine schedules (which only consist of subsets of indices) is sufficient to construct the corresponding solution of Pm | | $\sum w_jC_j$.

(b) The LP-relaxation can be obtained by replacing the integrality constraints with $x_s \geq 0$ for all $s \in S$. Since we do not want to enumerate all single-machine schedules in S, we just use a subset S' of S and solve the LP-relaxation for this subset, after which we find the values of the dual multipliers (shadow prices) π_j and λ for the first and second constraint, respectively. We then check whether it is useful to add single-machine schedule s to S' by looking at the *reduced cost* of s. The reduced cost of single-machine schedule s is equal to

$$c_s - \sum_{j=1}^{n} \pi_j a_{js} - \lambda$$

Formulate the pricing problem and show how to solve it to optimality using dynamic programming. Briefly explain why it is useful to solve it.

(c) Formulate a branching rule that can be used in the branch-and-price algorithm.

Length **indication**: 400 words.

Continue on next page.

Continue your answer of Exercise 4.

Exercise 5 (20 points)**.**
Consider the problem P3 | | $C_{\max}$.

(a) Construct a dynamic program that solves this problem to optimality in pseudo-polynomial time. Note that the lower the running time (measured in $O(\cdot)$ notation), the better your algorithm.

(b) Construct an FPTAS for this problem.

Length **indication**: 550 words.

Continue on next page.

Continue your answer of Exercise 5.

Exercise 6 (15 points)**.**
Consider the stochastic scheduling problem $\mathrm{Pm} \mid \mid \mathbb{E}\left[\sum w_j C_j\right]$.

(a) Consider the following instance with 4 jobs. Jobs 1 and 2 are deterministic jobs with $w_j = 1$ and $\mathbf{Pr}[P_j = 1] = 1$ $(j = 1, 2)$. Jobs 3 and 4 have $w_j = 10$ and processing time distribution $\mathbf{Pr}[P_j = 1] = 1 - \epsilon$ and $\mathbf{Pr}[P_j = \frac{19+\epsilon}{\epsilon}] = \epsilon$, and thus $\mathbf{E}[P_j] = 20$ $(j = 3, 4)$, for $\epsilon > 0$ small.

Determine an optimal policy for $m = 1$ for this instance. What is the expected total weighted completion time for $\epsilon \to 0$?

(b) Consider the same jobs as in part (a). Determine an optimal policy for $m = 2$. What is the expected total weighted completion time for $\epsilon \to 0$?

You do not need to prove the optimality of your policy.

(c) Give an instance for $m = 2$ such that an optimal policy leaves a machine deliberately idle, i.e., it leaves a machine idle although there is still one or more jobs to be processed.

You only need to give the instance and the optimal policy. You do not need to prove the optimality of the policy.

(d) Can deliberate idleness also occur in an optimal policy for the case that jobs may be preempted?

Length **indication**: 400 words.

This page is intentionally left blank.