Utrecht University
Faculty of Science
Department of Information and Computing Sciences

Final Exam Algorithms for Decision Support, Monday November 5, 2018, 13.30-16.30 hr.

Solutions are added in blue

- Switch off your smart phone, PDA and any other mobile device and put it far away.
- A calculator is not necessary and you are **not** allowed to use one.
- This exam consists of **4** questions and a bonus question.
- Answers may be provided in either Dutch or English.
- All your answers should be clearly written down and provide a clear explanation. Unreadable or unclear answers may be judged as false.
- Please write down your name and student number on every exam paper that you hand in.
- The maximum score is indicated at each of the questions.
- If you want to go to the bathroom, you have to hand in your smart phone.

Good luck, veel succes !

Question 1: The home care company

(a: 1 pt., b: 0.5 pt., c: 0.75 pt., d: 0.75 pt, total: 3 points.)

We consider a home care company in the center of the Netherlands. It employs 10 nurses and 20 assistants. There are n customers requiring care from the company. The care for a customer is performed in two steps:

1. Service from an assistant. The required time equals the maximum of 10 minutes and an amount following a normal distribution with an average of p_i. The average assistant service time equals 20 minutes and the variance equals v_i^2.

2. Service from a nurse. This takes an amount of time which equals the maximum of 5 minutes and an amount following a normal distribution with an average of q_i. The average nurse service time equals 10 minutes and the variance equals w_i^2.

The steps have to be performed consecutively, so a nurse can only start after the assistant has finished.

You may assume that the travelling time from one customer to the next follows a gamma distribution with an average of 15 minutes and a shape parameter $\alpha = 3$.

Furthermore, customers are assigned to assistants in increasing order of their index i. This implies that, if an assistant has finished her/his work, she/he is going to travel to the customer with lowest index which has not been assigned to an assistant yet. The company wants that the nurse arrives shortly after the assistant has finished. Therefore, a nurse is requested to start travelling 5 minutes after the start of the work of the assistant. If, at this point in time, there is no nurse available, the request is put into a FIFO queue.

The company wants to perform a simulation study to determine the *makespan*, i.e., the total time required to serve the complete set of the customers. Moreover, they want to determine the average service time of the customers.

(a) Which events are included in the event-scheduling model for this problem? Draw an event graph and for each arc give the corresponding time delay.

Solution

We denote the distribution of the service time by the assistant by $\max(10, N(p_i, \sigma^2_{assistant}))$ and the service time of the nurse by $\max(5, N(q_i, \sigma^2_{nurse}))$. We obtain the event graph given in Figure 1.

(b) Describe two *different* possible actions to validate the simulation model of the home care company. The descriptions of the actions should be as specific as possible and include which aspects of the model are adressed. A general remark like: "discussion with an expert" will not result in any credit points.

Solution

Use Chapter 5 of the book of Law.

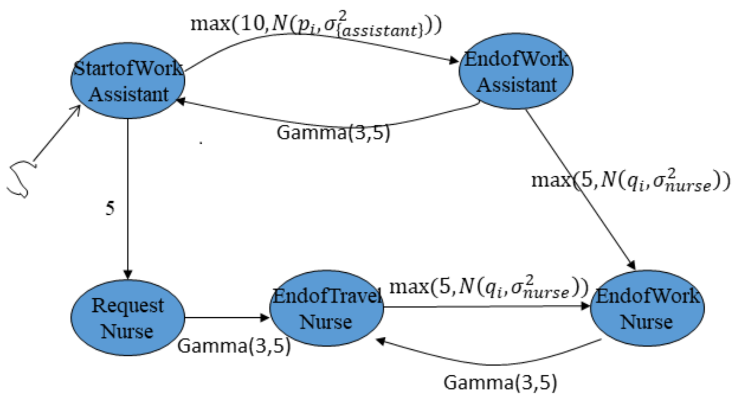

Figure 1: The event graph.

(c) Describe how we can generate the travelling times of the nurses and assistants in a program written in an imperative programming language like Java or C♯ without using **any** specific random generation libraries or functions.
Note: You do not have to give a program, but just a description or pseudo-code.
Solution
We denote the travelling time by T. T follows a Gamma(3,5) distribution so $t = X_1 + X_2 + X_3$, where X_1, X_2, X_3 are independent and x_i follows an exponential distribution with an average of 5. Hence we can generate T is the following way:

(1) Generate U_1, U_2, U_3 from $U[0;1]$ with the Mersenne Twister.

(2) Generate X_1, X_2, X_3 from the exponential distribution with inverse transform. The distribution function of the exponential distribution is $F(x) = 1 - e^{-\frac{x}{5}}$. You can find the inverse as follows:

$$y = 1 - e^{-\frac{x}{5}} \Leftrightarrow e^{-\frac{x}{5}} = 1 - y \Leftrightarrow x = -5\ln(1-y).$$

So, $F^{-1}(y) = -5ln(1-y)$.
So $X_i = -5ln(1-U_i) \quad (i = 1, 2, 3)$.

(3) Return $T = X_1 + X_2 + X_3$

(d) Suppose we have the following 5 samples of the service time of the nurses: 6, 7, 8, 10, 13. A consultant claims that the service times are uniformly distributed on $[5; 15]$. We want to test this claim with a Q-Q plot. What are the points in the Q-Q plot? Draw the Q-Q plot. Does the Q-Q plot support the claim or not? Explain your answer.
Observation: the number of samples is chosen artifically small for educational reasons.
Solution

In a Q-Q plot we draw the points $(X_i, F^{-1}(\frac{i-0,5}{n}))$. If this plot resembles the line $y = x$ we believe that the observations are indeed from distribution F. For $U[5;15]$, we have $F(x) = \frac{x-5}{10}$ so $F^{-1}(y) = 10y + 5$. The points that you have to draw are: $(6, F^{-1}(0,1))$, $(7, F^{-1}(0,3))$, $(8, F^{-1}(0,5))$, $(10, F^{-1}(0,7))$, $(13, F^{-1}(0,9))$; in other words $(6,6), (7,8), (8,10), (10,12), (13,14)$. If you draw the graph you will see that it does not look like the line $y = x$.

Question 2: Beun de Haas *(a: 1 pt., b: 1 pt total: 2 points.)*
Beun de Haas is an independent entrepeneur, who performs small jobs for clients. We consider the planning for the next T days. On day t ($t = 1, \dots, T$) Beun can spend at most Q_t time units on jobs for clients. Given is a set of n possible jobs where job j ($j = 1, \dots, n$) has a reward of c_j when completed and takes a_j time units. If Beun decides to perform a job, he has to do the complete job on a single day. Because of the agenda of the clients, a job can only be performed on a given subset of days. We denote by J_t, the set of jobs that are available on day t. The objective is to make a workplan for Beun for T days that maximizes his reward.

(a) Give an integer linear programming formulation for this problem. Clearly describe the decision variables, objective, and constraints.

Solution

We use binary decision variables x_{jt}, where x_{jt} equals 1 of job j is performed on day t and 0 otherwise.

Parameter b_{jt} equals 1 if $j \in J_t$ and 0 otherwise. We now obtain the following ILP formulation

$$\max \sum_{t=1}^{T} \sum_{j=1}^{n} c_j x_{jt}$$

$$\sum_{j=1}^{n} a_j x_{jt} \leq Q_t \quad (t = 1, \dots, T)$$

$$\sum_{t=1}^{T} x_{jt} \leq 1 \quad (j = 1, \dots n)$$

$$x_{jt} \leq b_{jt} \quad \forall_{j,t}$$

$$x_{jt} \in \{0, 1\} \quad \forall_{j,t}$$

The first constraint ensures that the total processing time of the jobs performed on day t fits within the available time on that day. The second constraint states that each job is processed at most once. The third constraints models that a job can only be processed on a day on which it is available.

(b) For some combinations of jobs, it is more efficient if they are planned on the same day. Let S be a set of job pairs which require less time if they are planned on the same day. For $(i,j) \in S$, we have that if job i and j are planned on the same day, this saves an amount of time equal to s_{ij}. Extend the model of part (a) to an integer linear programming model including this option. Clearly describe the decision variables, objective, and constraints.

Solution

For $(i,j) \in S$ we add a decision variable y_{ijt} which is equal to 1 if job i and j are both processed on day t, and 0 otherwise. We change the first constraint into:

$$\sum_{j=1}^{n} a_j x_{jt} - \sum_{(i,j)\in S} s_{ij} y_{ijt} \leq Q_t \quad (t = 1, \ldots, T)$$

To ensure that y_{ijt} can only be set equals to 1 if job i and j are processed on day t we add

$$2y_{ijt} \leq x_{it} + x_{jt}.$$

Note that this implies that y_{ijt} can only be 1 if i and j are both in J_t.

Question 3: Load balancing
(a: 0.5 pt., b: 1 pt. c: 0.5 pt. d:1 pt. e: 0.5 pt. total: 3.5 points.)
In this question you may use the fact that the following problems are $\mathcal{N}P$-complete: PARTITION, SUBSET SUM, CLIQUE, INDEPENDENT SET,VERTEX COVER, HAMILTONIAN PATH, HAMILTONIAN CYCLE, TRAVELLING SALESMAN PROBLEM
We consider the following LOAD BALANCING problem. We are given n computation jobs. Job j has processing time p_j ($j = 1, \dots, n$). These jobs have to be divided over m processors. The goal is to assign the jobs to the processors in such a way that $L_{\max}$ is minimized, where $L_{\max}$ equals the maximum workload of a processor. The workload of a processor is defined as the sum of the processing times of the jobs on the processor.

(a) Formulate the decision problem corresponding to LOAD BALANCING.
Solution
Given a number $Q > 0$. Is there an assignment of the jobs to the processors with $L_{\max} \leq Q$?

(b) Prove that the decision variant of LOAD BALANCING is $\mathcal{N}P$-complete.
Solution
The complete prove is not given, only the reduction. The complete proof proceeds according to the lines of the lecture slides.
We use a reduction from PARTITION.
Let $n, a_1, a_2, \dots a_n$ be an instance of PARTITION. This is a 'yes'-instance if there is a subset S of $\{1, 2, \dots, n\}$ such that $\sum_{j \in S} a_j = \frac{1}{2} \sum_{j=1}^{N} a_j$.
We define the following instance n', m, Q, $p_1, \dots, p_{n'}$ of LOAD BALANCING:

- $n' = n$
- $m = 2$
- $Q = \frac{1}{2} \sum_{j=1}^{n} a_j$
- $p_j = a_j \ \forall_j$.

Now you can show that we have 'yes' in PARTITION if and only if we have 'yes' in LOAD BALANCING.

(c) Describe two **construction** heuristics for LOAD BALANCING.
Solution

- Assign job 1 to machine 1, job 2 to machine 2, ..., job m to machine m, job $m+1$ to machine 1 etc.
- Sort the jobs in decreasing order of their processing times. Assign the jobs to the machines in this order and assign each job the machine with the lowest workload.

(d) Give an integer linear programming formulation for LOAD BALANCING.

Solution

We define decision variables x_{ij} equal to 1 of job j is assigned to machine i and 0 otherwise. Moreover, we use a decision variable for $L_{\max}$. We obtain the following ILP-formulation:

$$\min L_{\max}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \quad (j = 1, \ldots, n)$$

$$\sum_{j=1}^{n} p_j x_{ij} \leq L_{\max} \quad (i = 1, \ldots, m)$$

$$x_{ij} \in \{0,1\} \forall_{i,j}$$

The first constraint ensures that each job is processed exactly once, the second constraint ensures that $L_{\max}$ is the maximum workload of the processors.

(e) Now we are allowed to buy two additional processors. The first additional processor costs Q_1 and the second processor Q_2, where $Q_2 < Q_1$. The second additional processor can only be used in combination with the first. Extend the model of part (d) with the possibility of extra processors. The goal is to minimize the sum of $L_{\max}$ and the cost of the extra processors.

Solution

We add binary variables y_1, y_2 to signal if we buy the first and second additional processor, respectively. Morever we define $x_{m+1,j}$ equal to 1 if job j is assigned to the first additional processor and 0 otherwise. $x_{m+2,j}$ is defined in the same way. We now have the following model:

$$\min L_{max} + Q_1 y_1 + Q_2 y_2$$

$$\sum_{i=1}^{m+2} x_{ij} = 1 \quad (j = 1, \ldots, n)$$

$$\sum_{j=1}^{n} p_j x_{ij} \leq L_{max} \quad (i = 1, \ldots, m+2)$$

$$\sum_{j=1}^{n} x_{m+1,j} \leq n y_1$$

$$\sum_{j=1}^{n} x_{m+2,j} \leq n y_2$$

$$y_2 \leq y_1$$

$$x_{ij} \in \{0,1\} \forall_{i,j}$$

$$y_1, y_2 \in \{0,1\}$$

Observe that third and fourth constraint are forcing constraints to make sure that a processor is only used when it is bought. The fifth constraint models that we can only buy processor 2 if we have already bought processor 1.

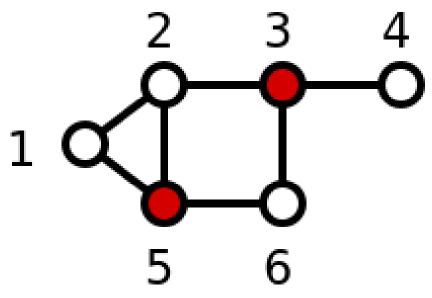

Figure 2: An example of a dominating set

Question 4: Dominating set
(total: 1.5 points)
In this question you may use the fact that the following problems are $\mathcal{N}P$-complete: PARTITION, SUBSET SUM, CLIQUE, INDEPENDENT SET,VERTEX COVER, HAMILTONIAN PATH, HAMILTONIAN CYCLE, TRAVELLING SALESMAN PROBLEM
We consider the problem DOMINATING SET. We are given a graph $G = (V, E)$. A subset D of the vertices is called a dominating set if each vertex not in D is connected to at least one vertex in D. Note that this is different from a vertex cover, which is a subset of the vertices such that each edge has one end point in the subset. In the example $\{3, 5\}$ is a dominating set, but not a vertex cover. Prove that the decision variant of DOMINATING SET is $\mathcal{N}P$-complete.

Solution
The complete prove is not given, only the reduction. The complete proof proceeds according to the lines of the lecture slides.
We use a reduction from VERTEX COVER. Given an instance K, $G = (V, E)$ of VERTEX COVER. This is a 'yes' instance if there is a subset W of V of size at most K such that for each $(u, v) \in E$ we have that $u \in W$ or $v \in W$. We construct an instance K', G' of DOMINATING SET in the following way.

- $K' = K$
- We construct the graph G' from G as follows. We include all the nodes and edges from G. For each $(u, v) \in E$ we add a new vertex w_{uv}. Now we add the edges (u, w_{uv}) and (v, w_{uv}). Note that this boils down to replacing each edge by a triangle.

Now we can prove that G has a vertex cover of size at most K if and only of G' has a dominating set of at most K'.

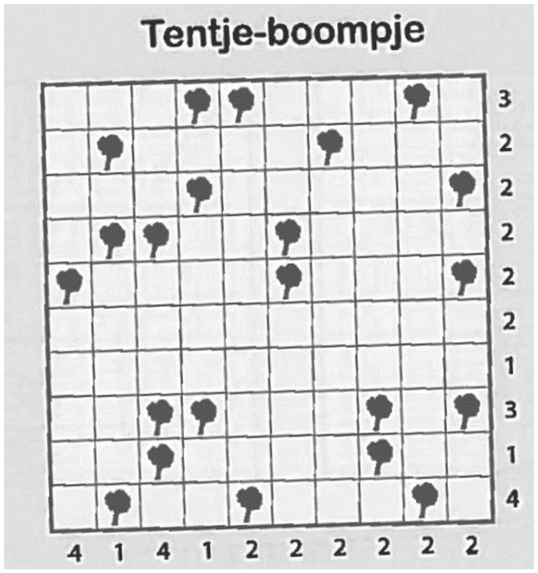

Figure 3: An example of Camp ground

Bonus question 5: Campground
(1 point)
We consider the puzzle called 'Campground', also known as *'Tentje-boompje'*. We have to find the positions of tents on a campground. The campground is modelled by a grid of m rows and n columns. When you want to stay at the campground, the owner directs you to 'your' tree and you must put your tent directly next to this tree (not diagonally). There can be at most one tent on a position. Different tents are not allowed to be on horizontally, vertically or diagonally adjacent positions. We are given numbers r_i and c_j, which denote the number of tents in row i and column j respectively. Formulate an Integer Linear Programming problem that solves the puzzle.