Answers ADS exam 2016-2017

Maxwell Hessey

October 2017

1 Simulation of an emergency department

(a) *Validation*

E.g. :

- Validate assumptions such as DN1 only works on patients of category A by setting scenarios to people from DN1 or their managers such as: "What if a patient comes in with a barely non-critical condition and you know they will have to wait for x minutes if you do not treat them, would you (partially) treat them?"
- Input validation: Compare the generated patient treatment times with real life patient treatment times. Do the distributions match?
- If such a department exists (or a similar one), do a simulation run and compare the results to the real world situation in terms of distribution of waiting times for each category, working hours of the staff, percentage of time the staff is busy etc. (Output validation)

(b) *Generating treatment times.* [1]

First note that as 6 is an integer, Gamma(6,10) is a 6-Erlang(10) distribution. This means we can generate outcomes using the sum of four exponential distributions with mean 10 (this is called convolution):

$$\text{6-Erlang(10)} = Exp(10) + Exp(10) + Exp(10) + Exp(10) + Exp(10) + Exp(10) \tag{1}$$

So far so good, but how do we sample from Exp(10)? To draw from an exponential distribution, we use inverse transform sampling. The CDF of the exponential function is: $1 - e^{-\frac{x}{\beta}}$; in this case $\beta = 10$. First we find the inverse:

$$\begin{aligned} F_{Exp}(x) = 1 - e^{-\frac{x}{\beta}} &= y \\ -e^{-\frac{x}{\beta}} &= y - 1 \\ e^{-\frac{x}{\beta}} &= 1 - y \\ -\frac{x}{\beta} &= \ln(1 - y) \\ x &= -\beta \ln(1 - y) = F^{-1}_{Exp}(y) \end{aligned}$$

Now we obtain the following algorithm:

1. For $i = 1, 2, \ldots, 6$ generate a number u_i from $U[0, 1]$ by a linear congruential generator, which applies $z_{j+1} = az_j \mod p$, p a large prime e.g. $2^{31} - 1$.
2. Define $x_i = -10 \ln(1 - u_i)$ for $i = 1, 2, \ldots, 6$.
3. Return $x = x_1 + x_2 + \ldots x_6$.

[1] See the slides on RNG on: http://www.cs.uu.nl/docs/vakken/mads/lectureRNG.pdf

(c) *Experimental setup* This question is about step 7 from the steps in a sound simualtion study discussed in the lecture Modelling and Simuation study.

The key here is that the emergency department works continuously. So we want to find information about the steady state of the department, not about the warm-up phase.

A possible technique is separate runs. These should be very large to minimize the effect of the warmup phase. Or if the warmup phase is known to last until time t_0, then data can be collected only after t_0 and the runs can be shorter. To find t_0 you should apply the moving average method decribed in lecture on output (at the exam you should give the formules for this).

After the prodcution runs you can do the following (this is part of step 9 of a sound simulation study so strictly speaking not included in the question). Suppose we choose to do n runs. For each run we obtain the maximum waiting time, let W_i be the maximum waiting time of run i. To estimate the maximum waiting time we compute

$$\bar{W}(n) = \frac{\sum_{i=1}^{n} W_i}{n}.$$

Then we compute the sample variance $S^2(n) = \frac{\sum_{i=1}^{n}(W_i - \bar{W}(n))^2}{n-1}$. We can use this to compute a confidence interval:

$$[\bar{W}(n) - t_{n-1,1-\frac{\alpha}{2}}\sqrt{\frac{S^2(n)}{n}};\ \bar{W}(n) + t_{n-1,1-\frac{\alpha}{2}}\sqrt{\frac{S^2(n)}{n}}],$$

where $t_{n-1,1-\frac{\alpha}{2}}$ is from the student's t-distribution with $n-1$ degrees of freedom and can be found in a statistical table. This means the maximum waiting time from the model is in this interval with probability $1-\alpha$.

2 Tutoring Lessons

(a) *Each student takes one lesson*

Decision variables: For each timeslot, X must decide which student to teach. So let x_{jt} be 1 if X teaches student j at timeslot t and 0 otherwise.

Objective function: Maximize the profit X makes, which is the sum of all prices payed during the slots he works in (2)

Constraints: Each student takes only one lesson (3). In each hour X can only do one tutoring lesson (4). Integrality (5).

Let J denote the set of all available students. This leads to:

$$\max \sum_{j \in J t \in T} p_{jt} x_{jt} \tag{2}$$

$$\sum_{t \in T} x_{jt} \leq 1 \quad \forall j \in J \tag{3}$$

$$\sum_{j \in J} x_{jt} \leq 1 \quad \forall t \in T \tag{4}$$

$$x_{jt} \in \{0,1\} \quad \forall j \in J, t \in T \tag{5}$$

$$\tag{6}$$

(b) *TUM*

Out of scope for the year 2017-2018

(c) *Each student potentialy takes several lessons*

X must make the same decisions as in question a, his objective also remains the same. Only the constraints change. We add the constraint that a student can only be given a lesson if they are available at that time (10). The constraint that each student can take at most one lesson becomes the constraint that each student can take at most q_j lessons (8).

Thus we have:

$$\max \sum_{j \in J t \in T} p_{jt} x_{jt} \tag{7}$$

$$\sum_{t \in T} x_{jt} \leq q_j \quad \forall j \in J \tag{8}$$

$$\sum_{j \in J} x_{jt} \leq 1 \quad \forall t \in T \tag{9}$$

$$x_{jt} \leq a_{jt} \quad \forall j \in J \forall t \in T \tag{10}$$

$$x_{jt} \in \{0,1\} \quad \forall j \in J, t \in T \tag{11}$$

(d) *Crash Courses*

Now all lessons must be in consecutive hours. Note 01.00-02.00 and 10.00-11.00 are consecutive hours, so all T of X's hours are consecutive. That is, we do not need to consider time of day and we can stick to numbering the numbering of timeslots we used before $(1, 2, \cdots, T)$.

This means we only have to decide which slot we start teaching student j at. Let y_{jt} be 1 if X starts teaching j a crash course at time t. The profit P_{jt} of y_{jt} now equals $\sum_{s=t}^{t+q_j-1} p_{js}$ Obviously we can only start teaching a set of lessons once (14). Moreover, X can teach only one student at the same time (15). We add the constraint that a student can only be given lessons if they are available at that time (15). As always, X wants to maximize his profit, so we have:

$$\max \sum_{j \in J t \in T} P_{jt} y_{jt} \tag{12}$$

$$\sum_{t \in T} y_{jt} \leq 1 \quad \forall j \in J \tag{13}$$

$$\sum_{j=1}^{n} \sum_{s=t-q_j+1}^{t} y_{js} \leq 1 \quad \forall t \in T \tag{14}$$

$$y_{jt} \leq b_{jt} \quad \forall j \in J \forall t \in T \tag{15}$$

$$y_{jt} \in \{0,1\} \quad \forall j \in J, t \in T \tag{16}$$

3 Distribution network

1. *Showing UCF is $\mathcal{NP}$-Complete*

   Recall that to show a problem P is $\mathcal{NP}$-Complete we need to show it is in $\mathcal{NP}$ and that there is a polynomial time reduction from a known $\mathcal{NP}$-Complete problem to P.

   First we show UCF is in $\mathcal{NP}$: Any yes solution y can be stored in polynomial space with respect to the input x. A solution can be stored as $x_{ij} = 1$ if customer j is supplied from depot i and 0 otherwise, with $y_i = 1$ if depot i is open and 0 otherwise. Clearly this is polynomial w.r.t. the space needed to define the cost functions c_{ij} and F_i.

   Any yes solution can be checked in polynomial time with respect to (x,y). Checking a solution may be done by looping once over all customers to check they are being supplied. During this loop, we can build a list of depots these customers are being supplied from and maintain the total cost required so far. Next we loop over the list of depots and check all these are open and maintain the total cost required (only counting each depot once). Finally we compare the total cost with Q. All this is done in $O(nm)$, which is polynomial w.r.t the input size.

   Next we show UCF is $\mathcal{NP}$ -hard by a reduction from Vertex Cover (VC).

   Let $(G(V, E), k)$ be any instance of Vertex Cover. Define an instance of UCF as follows:

   - For every edge $e \in E$ define a customer.
   - For every vertex $v \in V$ define a depot d_v.
   - Let the cost of supplying a customer from a depot equal 0 if in G,the vertex corresponding to the depot is incident to the edge corresponding to the customer and $Q + 1$ otherwise (basically, be impossibly large otherwise)
   - Let the cost of opening any depots be 1.
   - Let $Q = k$

   Clearly this is a polynomial time transformation.

   Consider any Yes instance of Vertex Cover. We show that the above transformation creates a Yes instance of UCF: Since we have a yes instance, there exists a vertex cover VC of size at most k. We will show that opening the depots corresponding to the VC is a feasible solution to UCF with cost at most Q. By construction, every customer corresponds to some edge e in G. Because VC is a Vertex Cover, there exists a vertex $v \in VC$ such that v is incident to e. Therefore there is an open depot (corresponding to v) from which e can be supplied with cost 0. The cost of opening all depots is equal to the size of the vertex cover, which is at most k. So the total cost for this solution is at most k.

   Conversely, consider any Yes instance of UCF resulting from the above transformation. We show that the instance transformed from must be a Yes instance of VC. A yes instance of UCF implies we have a feasible solution of cost at most Q. First we observe that to be a feasible solution with cost at most Q, no customer corresponding to edge e can be supplied from a depot corresponding to a vertex that is not incident to e (because the costs for this alone are $Q+1$ by construction). So the total costs of supplying customers from depots must be 0. Furthermore, since the cost of opening a depot is 1, the total number of openened depots must be at most $Q = k$. These at most k depots must supply all customers, so the corresponding at most k vertices must be incident to all edges in E. So the corresponding vertices are a vertex cover of size at most k.

   QED.

2. *Showing DT is $\mathcal{NP}$-Complete*

   As before, we show $DT \in \mathcal{NP}$ an then a polynomial time reduction from a known $\mathcal{NP}$-Complete problem (in this case HP).

DT $\in \mathcal{NP}$:

Any yes solution y can be stored in polynomial space with respect to the input x. Store the solution as an ordered list of locations. Every location l is visited at most once because it needs to be, and at most once for every other time a location l' needs to be visited (it might be that the shortest route to l' goes via l). We visit at most $O(n)$ locations because we need to, so at most $O(n^2)$ locations in total. So the list is at most $O(n^2)$ long, which is polynomial in the input parameter n.

Any yes solution can be checked in polynomial time with respect to (x,y). Make a list L of all locations that are yet to be visited . Follow the locations in order of the solution, whenever a location is visited, check if it is in the list and if so remove it (in $O(n)$). Keep track of the total cost. When the solution list has been exhausted, check $L = \emptyset$. And check the total cost is at most W. Total running time is $O(n^3)$, so polynomial w.r.t. input parameter n.

DT is $\mathcal{NP}$ -hard by reduction from Hamilton Path (HP). Let $G = (V, E)$, be any instance of the HP. Define an instance of DT as follows:

- For each vertex v in V create a pickup location p_v and delivery location d_v
- Define the distance function $w(u, v)$ as follows:
  - 0 between p_v and d_v $\forall v \in V$
  - $|V| + 2$ between p_v and d_u $\forall u \in V \neq v$
  - 1 between d_u and p_v $\forall (u, v) \in E$
  - $|V| + 2$ between d_u and p_v iff $(u, v) \notin E$
  - 1 between s and p_v $\forall v \in V$
  - 1 between d_v and s $\forall v \in V$
- Let $W = |V| + 1$

Clearly this is a polynomial time transformation.

Now assume we have a YES instance of HP, we show that the above transformation results in a YES instance of DT. Because we have a YES instance of HP, there exists a path in G that visits each vertex exactly once. Number the vertices in order of visiting in the path $(v_1, v_2, \cdots, v_W)$. Consider the route that follows this path: $s, p_{v_1}, d_{v_1}, p_{v_2}, d_{v_2}, \cdots, p_{v_W}, d_{v_W}, s$.

The cost of s to p_{v_1} is 1 by construction. The cost of going from p_{v_i} to d_{v_i} is 0 by construction. There must be an edge in G between v_i and v_{i+1} (otherwise they could never be subsequent vertices in the HP), so the cost of going from d_{v_i} to $p_{v_{i+1}}$ is 1 by construction.

Thus the total cost is 1 + the length of the path + 1, or $1 + (|V| - 1) + 1 = |V| + 1$

Now assume we have a YES instance of DT, we will show that we have a YES instance of HP. Because we have a YES instance of DT, there must exist a route, visiting all locations then returning to the depot with total length at most W. Because each location is visited at least once; there are W locations; the total route length is at most W, it must hold that each location is visited exactly once. Since $W = |V| + 1 < |V| + 2$, by definition of our distance function w, it is not possible that we go from d_u to p_v unless $(u, v) \in E$. Now name the locations in order of their position in the route $s, p_{v_1}, d_{v_1}, p_{v_2}, d_{v_2}, \cdots, p_{v_W}, d_{v_W}, s$. Each edge (v_i, v_{i+1}) must exist in E. Consider the corresponding path in G: $(v_1, v_2), (v_2, v_3), \cdots, (v_{W-1}, v_W)$. This is a path in G (because all the edges exist) and visits each vertex once (because each corresponding location is visited exactly once). So we have a YES instance of HP.

QED.

3. *Construction Heuristics for DT* Note the following relation with TSP: Build a graph as follows: Start with the bipartite graph with vertices of pickup locations and delivery location. Add vertex s and s'. Add arcs from s to all pickup locations, from all delivery locations to s' and from s' to s. Use the distance function w on the arcs, and give s' to s the distance 0.

For non-existing arcs we can add arcs with infinite cost. Now we are looking for the cheapest TSP tour on this graph. So any of the TSP construction heuristics will work. See the slides [2] for a complete list.

For example: Nearest Neighbour: from d_i go to the nearest unvisited pickup location. Repeat until all locations have been visitied.

Farthest insertion heuristic: Create a trivial (single vertex) cycle. Until all vertices are in the cycle, repeat: Add the vertex to the cycle that is furthest away.

4 Bayesian Networks

Out of scope for the year 2017-2018

[2] http://www.cs.uu.nl/docs/vakken/mads/lecture_tsp.pdf