Network Optimisation Homework Solutions

---

1. Consider a transshipment problem on the graph $G = (V, E)$, where there is a vertex corresponding to each input port and one corresponding to each output port, and there is an arc from every input port to every output port, but no arcs between input ports, or between output ports. In other words, $V = \mathcal{I} \cup \mathcal{O}$, and $E = \mathcal{I} \times \mathcal{O}$. A graph in which the vertex set can be partitioned into two subsets such that there are no edges within a subset is called bipartite. In this case, every edge between the two subsets is present. So we call G a *complete bipartite graph*. We associate a cost $-Q_{ij}$ with the edge $(i, j) : i \in \mathcal{I}, j \in \mathcal{O}$. We associate a unit supply (i.e., demand of -1) with each input port $i \in \mathcal{I}$, and a unit demand with each output port $j \in \mathcal{O}$. This is the input to the algorithm provided for solving the transshipment problem.

   Suppose the algorithm outputs an integer-valued solution, i.e., the flow on every edge is integer-valued. (This is what the question should have stated, rather than tree solution; in this problem, there are no feasible tree solutions. However, there are feasible solutions, e.g., to have a flow of $1/N$ on each edge.) If the flow is integer-valued, then since the supply or demand at each node is 1, the edge flows have to be either 0 or 1. Moreover, for each input port, there has to be exactly one edge incident on it that carries a flow of 1, and all others have to be 0. Likewise for each output port. But this means that the edges carrying a flow of 1 constitute a matching. Denote it M, i.e., M is the set of edges carrying unit flow.

   As these edges constitute a minimum cost solution to the transshipment problem, they minimise $\sum_{(i,j)\in M} -Q_{ij}$, or equivalently maximise $\sum_{(i,j)\in M} Q_{ij}$.

   If the algorithm outputs a solution that is not integer-valued, then that does not immediately yield a matching.

2. Associate a unit demand with each girl and a unit supply with each boy (or the other way round, if you prefer). Consider a bipartite graph, where there is an arc oriented from each boy to each girl that he knows, but no arcs between boys or between girls. There are also no arcs between girls and boys who don't know each other. Associate a unit cost with each arc.

   Consider the transshipment problem with the above graph, node demands and edge costs. An initial feasible solution is to assign a flow of $1/k$ on each arc. As there are exactly k arcs directed out of each boy, and exactly k arcs directed into each girl, these flows satisfy the demands. Now, the Integraility Theorem tells us that, as there is a feasible solution, there is an integer-valued feasible solution. Such a solution must necessarily assign flows of 0 or 1 to arcs (a flow of 2 or bigger would violate supply and demand constraints). Clearly, each boy node must have exactly one edge with a flow of 1 directed out of it, and each girl node exactly one edge with a flow of 1 directed into it. Thus, the edges carrying unit flow specify a matching.

3. As suggested in the hint, consider a bipartite graph with a node for each row and one for each column, and an edge directed from row node i to column node j if $x_{ij} > 0$. There are no edges between row nodes or between column nodes. Associate a unit supply with each row, a unit demand with each column, and a unit cost with each edge. Consider a transshipment problem with the above input.

   The doubly stochastic matrix X specifies an initial feasible solution for this transshipment problem; assign a flow x_{ij} to edge (i, j). As $\sum_j x_{ij} = 1$, the flows out of row node i add up to 1, which is the supply at that node. As $\sum_i x_{ij} = 1$, the flows into column node j add up to 1, which is the demand

at that node. Hence the schedule specified by X is feasible. Now, the Integraility Theorem tells us that there is an integer-valued feasible solution, i.e., a solution which takes only values 0 or 1; as in the last problem, values of 2 or more are impossible. Use this integer-valued solution to specify a matrix P by setting p_{ij} equal to the flow on the arc (i, j) in the solution. As the flow P is feasible, we must have $\sum_j p_{ij} = 1$ and $\sum_i p_{ij} = 1$. But this means that there must be exactly one 1 in each row and each column, so P must be a permutation matrix.

The permutation matrix P can only assign flow to arcs of the graph. The graph was constructed so that arc (i, j) is present if and only if $x_{ij} > 0$. Hence, p_{ij} can be 1 only if $x_{ij} > 0$ (obviously, it doesn't have to be 1 for every such x_{ij}), and is 0 otherwise.

4. The graph for this model will have a layered structure. The vertex set is the union of a single source node s corresponding to X_0, and a copy of $\mathcal{X}$ for each time slot $\{1, 2, 3, \ldots, n\}$, for a total of $n|\mathcal{X}| + 1$ vertices. Think of the source vertex as layer 0 and the elements of $\mathcal{X}$ corresponding to time slot k as the vertices of the k^{th} layer. There is an edge directed from each vertex in layer k to each vertex in layer $k + 1$.

   We want to associate edge lengths with the probabilities in the expression for the likelihood that we are trying to maximise. But edge lengths add up along a path, while the probabilities in the expression multiply. The way to deal with this is to take logarithms. Also, in the shortest path problem, we are minimising, while for the MLE, we are maximising, so we should change sign. This leads us to specify the edge lengths (the notation is explained just below):

$$d(i^k, j^{k+1}) = -\log p(i, j) - \log \pi(i, j; Y_{k+1}).$$

   Here i^k refers to node i in layer k (i.e., the node corresponding to the Markov chain being in state i at time k) and j^{k+1} to node j in layer $k + 1$. If the Markov chain was in state i at time k and we observed the output Y_{k+1} at time $k + 1$, how likely is it that the Markov chain moved to state j? The likelihood is exactly $p(i, j)\pi(i, j; Y^{k+1})$, and the edge length is the logarithm of this likelihood. Maximising the log-likelihood is equivalent to maximising the likelihood because the logarithm is a monotone increasing function (if the base is bigger than 1; we implicitly assume the base is e, though the choice of base doesn't matter, so long as it is bigger than 1).

   We now solve the shortest path problem on this graph to find the shortest paths from s to all vertices in layer n. If vertex i happens to have the minimum path length of all vertices in this layer, then the most likely state at time n is $X_n = i$.

   Remark: As probabilities lie in $[0, 1]$, their logarithms are zero or negative, so edge lengths are non-negative. If some probabilities are zero, then the corresponding edge lengths are infinite. We can simply allow $+\infty$ as a possible value for edge length, or say that those edges don't exist; in either case, they will play no role in the shortest path problem.