CSC473W18: Homework Assignment #3: Solutions

Question 1. (16 marks)

a. The main observation is that exists a vector $x \geq 0$ such that $Ax \leq b$ if and only if there exist vectors $x \geq 0$ and $s \geq 0$ such that $Ax + s = b$. This is in turn true if and only if there exists a vector $\tilde{x} \geq 0$ such that $\tilde{A}\tilde{x} = b$, where $\tilde{A} = (A\ I)$ and I is the $m \times m$ identity matrix. Applying the Farkas Lemma, we get that either such a $\tilde{x}$ exists, or there exists a $\tilde{y}$ such that $\tilde{A}^\intercal \tilde{y} \leq 0$ and $b^\intercal \tilde{y} > 0$, but not both. Observe that $\tilde{A}^\intercal \tilde{y} \leq 0$ is equivalent to the two sets of inequalities $A^\intercal \tilde{y} \leq 0$ and $\tilde{y} \leq 0$. So, in summary, we have that exactly one of the following holds:

- **i.** There exists a $x \in \mathbb{R}^n$ such that $x \geq 0$ and $Ax \leq b$.
- **ii.** There exists a $\tilde{y} \in \mathbb{R}^m$ such that $\tilde{y} \leq 0$, $A^\intercal \tilde{y} \leq 0$ and $b^\intercal \tilde{y} > 0$.

Taking $y = -\tilde{y}$, we see that the second condition is equivalent to the existence of a $y \geq 0$ such that $A^\intercal y \geq 0$ and $b^\intercal y < 0$.

b. As explained in the notes, we can apply what we just proved to the matrix and vector

$$\tilde{A} = \begin{pmatrix} A \\ -c^\intercal \end{pmatrix}; \quad \tilde{b} = \begin{pmatrix} b \\ -1 \end{pmatrix}.$$

We get that the system of inequalities in the question is infeasible if and only if there exists no $x \geq 0$ such that $\tilde{A}x \leq \tilde{b}$, which happens if and only if there exists a $\tilde{y} \geq 0$ such that $\tilde{A}^\intercal \tilde{y} \geq 0$ and $\tilde{b}^\intercal \tilde{y} < 0$. If the last coordinate of $\tilde{y}$ is 0, then we can write

$$\tilde{y} = \begin{pmatrix} y \\ 0 \end{pmatrix},$$

and we have $A^\intercal y \geq 0$ and $b^\intercal y < 0$ which implies that the system of inequalities $Ax \leq b, x \geq 0$ is infeasible. If the last coordinate of $\tilde{y} = s \neq 0$, we can write

$$\tilde{y} = \begin{pmatrix} sy \\ s \end{pmatrix}.$$

Then $\tilde{y} \geq 0$ is equivalent to $y \geq 0$, $s \geq 0$, and $\tilde{A}^\intercal \tilde{y} \geq 0$ is equivalent to $sA^\intercal y - sc \geq 0$, which, because $s \geq 0$, is equivalent to $A^\intercal y \geq c$. Finally, $\tilde{b}^\intercal \tilde{y} < 0$ is equivalent to $sb^\intercal y - s < 0$, which is equivalent to $b^\intercal y < 1$.

Question 2. (25 marks)

a. The flow problem is equivalent to the linear program

$$
\begin{aligned}
&\min \sum_{e \in E} w_e f_e \\
&\text{s.t.} \\
\forall u \in V \setminus \{s,t\} : &\sum_{v:(v,u)\in E} f_{vu} - \sum_{v:(u,v)\in E} f_{uv} = 0, \\
&\sum_{u:(u,s)\in E} f_{us} - \sum_{v:(s,v)\in E} f_{sv} = -1, \\
&\sum_{u:(u,t)\in E} f_{ut} - \sum_{v:(t,v)\in E} f_{tv} = 1, \\
\forall e \in E : &f_e \ge 0.
\end{aligned}
$$

Above we used the fact that the total flow going out of s, which is required to be 1, equals the total flow going into t. The constraint for t is not strictly necessary because it is implied by the other constraints, but it makes for a simpler dual program.

The dual program is

$$
\begin{aligned}
&\max y_t - y_s \\
&\text{s.t.} \\
\forall (u,v) \in E : &y_v - y_u \le w_{uv}.
\end{aligned}
$$

b. The complementary slackness conditions say that a feasible flow f and a feasible dual solution y are optimal if and only if for every edge $(u,v) \in E$ we have $(w_{uv} - y_v + y_u) f_{uv} = 0$. I.e. complementary slackness is satisfied if it is possible to send one unit of flow from s to t using only tight edges, i.e. edges $(u,v) \in E$ such that $y_v - y_u = w_{uv}$.

c. The algorithm starts with $y_u = 0$ for all vertices u. Initially we set $S = \{s\}$. Let

$$\delta = \min\{w_{uv} + y_u - y_v : u \in S, v \notin S\}.$$

We modify y_u to $y_u - \delta$ for every $u \in S$ and leave the other dual variables unchanged. Then we reset S to all vertices which are reachable from s along edges (u,v) for which $w_{uv} + y_u - y_v = 0$ (we call such edges "tight"). I.e. we add to S any vertex v which was not previously in S and for which after we updated y we have $w_{uv} + y_u - y_v = 0$. We stop when S contains t. Then we can take any path P from s to t which uses only tight edges and output the flow that has value $f_e = 1$ for every edge e of P, and value $f_e = 0$ for all other edges. By complementary slackness, this pair of flow f and feasible solution y are optimal.

Because all edge weights are positive, the initial dual solution $y = 0$ is feasible. To see that the solution remains feasible after every update to y, notice that the left hand side of the constraint $y_v - y_u \le w_{uv}$ increases only for edges (u,v) for which $u \in S$ and $v \notin S$, and for such edges it increases exactly by δ. We chose δ exactly so that for every such edge the constraint remains satisfied.

Finally, we need to argue that the algorithm terminates in a polynomial number of steps, and that each step can be executed in polynomial time. Since we assumed that the problem was feasible, we know that there is a path from s to t in G. Recall that S is the set of vertices reachable from s along tight edges. If S does not contain t, then there must be some edge (u,v) of G with $u \in S$ and $v \notin S$. Moreover, any edge going out of S can not be tight, i.e. for every edge $(u,v) \in E$ for which $u \in S$, $v \notin S$ we have $w_{uv} + y_u - y_v > 0$. By our choice of δ, for at least one edge (u,v) going out of S we have $w_{uv} + y_u - y_v = \delta$; so, after we update y, this edge becomes tight, and v becomes an element of S. Therefore, the size of S grows by at least one vertex for every update of y, and after at most

n updates S must contain t and the algorithm terminates. It is easy to see that each update can be implemented in time $O(n+m)$: it is enough to iterate over all edges to compute δ and then we can update y in time $O(n)$. Therefore, the algorithm can be implemented to run in time $O(mn)$.

A slightly more sophisticated implementation which uses the adjacency matrix representation of G runs in time $O(n^2)$. This implementation is left as an exercise. More interestingly, this algorithm can be seen as a variant of Dijkstra's algorithm. In particular, the vertices reachable from s along tight edges correspond to the black (visited) vertices in Dijkstra's algorithm. It can be shown that the value of the minimum weight flow equals the smallest weight of a path from s to t in G.

Question 3. (9 marks) The algorithm independently assigns each vertex to a random part in the partition, where each part is equally likely. Let us say that an edge (u, v) is *cut* if there exist some $i < j$ such that $u \in V_i$ and $v \in V_j$. Then the value of a solution is equal to the total weight of cut edges. Let us analyze the probability that an edge is cut. We have

$$\begin{aligned}
\mathbb{P}((u,v) \text{ is cut}) &= \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \mathbb{P}(u \in V_i \text{ and } v \in V_j) \\
&= \sum_{i=1}^{k-1} \sum_{j=i+1}^{k} \frac{1}{k^2} \\
&= \binom{k}{2} \frac{1}{k^2} = \frac{k-1}{2k}.
\end{aligned}$$

Let X_e be the indicator random variable equal to 1 if edge e is cut and to 0 otherwise; we have $\mathbb{E}[X_e] = \mathbb{P}(e \text{ is cut}) = \frac{k-1}{2k}$ for any edge e. Then the expected value of the solution output by the algorithm is

$$\mathbb{E}[\sum_{e \in E} w_e X_e = \sum_{e \in E} w_e \mathbb{E}[X_e] = \frac{k-1}{2k} \sum_{e \in E} w_e.$$

Since any solution to the problem cuts every edge in the best possible case, the right hand side above is at least $\frac{k-1}{2k}$ times the value of the optimal solution.