CS532 Homework 1

Qilin Ye

September 18, 2024

Solution to problem 1. (i) For each vertex v we assign a real valued variable x_v such that $x_v \geqslant 0$ indicates v is selected, and 0 otherwise. By the definition of a vertex cover, this means for each edge (u, v), $x_u + x_v \geqslant 1$. Therefore, an integer LP formulation can be given as follows:

$$\min \sum_{v \in V} x_v \qquad \text{subject to} \qquad \begin{cases} x_u + x_v \geqslant 1 & \text{for all } (u,v) \in E \\ x_v \in \mathbb{Z}^{\geqslant 0} & \text{for all } v \in V. \end{cases}$$

Observe immediately that we can further assume $x_v \in \{0,1\}$ for otherwise setting $x_v = \min(x_v, 1)$ yields a feasible solution that is at least as good as the original objective. Hence a modified integer LP is:

$$\min \sum_{v \in V} x_v \qquad \text{subject to} \qquad \begin{cases} x_u + x_v \geqslant 1 & \text{for all } (u,v) \in \mathbb{E} \\ x_v \in \{0,1\} & \text{for all } v \in V. \end{cases}$$

For the LP relaxation, replace $x_v \in \{0,1\}$ with $x_v \geqslant 0$.

(ii) Consider a triangle graph with vertices $\{u, v, w\}$ and three edges. The integer LP by pigeonhole at least needs assign 1 to at least two vertices, so $\min_v x_v$ in integer LP is 2. On the other hand, the fractional LP can assign $1/2$ to all three vertices, giving an objective of 1.5.

(iii) Let us first write the fractional LP in linear algebraic terms. Let x be the column vector length $|V|$ encoding x_v's. Let $\mathbf{1}_V$ denote the $|V| \times 1$ vector of ones and $\mathbf{1}_E$ likewise. Then the fractional LP primal can be formulated as

$$\min\{(\mathbf{1}_V)^T x \mid Ax \geqslant \mathbf{1}_E, x \geqslant 0\}$$

where A is the $|E| \times |V|$ matrix where $A_{e,v} = 1$ if and only if v is one of the two endpoints of edge e. The dual can therefore be formulated as

$$\max\{(\mathbf{1}_E)^T y \mid A^T y \leqslant \mathbf{1}_V, y \geqslant 0\}$$

where y is a column vector of shape $|E| \times 1$, i.e., one entry for each edge. In other words let us assign a variable y_e to each $e \in E$. The objective is $\max \sum_e y_e$. For each $v \in V$, let A_v be the corresponding row in A. Then the constraint is equivalent to

$$(A_v)^T y = \sum_{e \in E} A_{v,e} y_e = \sum_{e \text{ incident on } v} y_e \leqslant 1.$$

Putting it together, the dual (fractional) LP is:

$$\max \sum_{e \in E} y_e \qquad \text{subject to} \qquad \begin{cases} \sum_{e \text{ incident on } v} y_e \leqslant 1 & \text{for all } v \in V \\ y_e \geqslant 0 & \text{for all } e \in E. \end{cases}$$

Solution to problem 2. By duality theorems, since P is feasible and bounded, the optimal solutions x^* and y^* therefore satisfy $c^T x^* = b^T y^*$. Therefore, the following chain of inequalities

$$c^T x^* \leqslant (A^T y^*)^T x^* = (y^*)^T A x^* \leqslant (y^*)^T b = b^T y^*$$

imply that

$$c^T x^* = (y^*)^T A x^* = b^T y^*.$$

Therefore,

$$(y^*)^T (Ax^* - b) = (y^*)^T A x^* - (y^*)^T b = 0$$

and

$$(x^*)^T (Ay^* - c) = (x^*)^T A y^* - (x^*)^T c = 0,$$

from which we recover the linear algebraic equivalent expressions of (1) and (2).

Solution to problem 3. The challenge here lies in the fact that we have too many constraints: we have to satisfy both flow conservation (flow in = flow out) and edge capacity constraint. This complicates the dual since each variable in dual corresponds to a constraint in the primal. To this end, we transform the primal LP via *flow decomposition*, in which the first constraint (flow conservation) is naturally satisfied. In this perspective, a *path* is a valid $s-t$ walk, and the entire network flow can be seen as choosing k paths $p_1, \cdots, p_k$ and sending amount x_i to each path p_i. Observe that under this decomposition,

$$\sum_{(s,v)} f(s,w) = \sum_{i=1}^{k} x_i,$$

since the total amount of flow leaving f (LHS) is precisely the total amount of flow we send along each path (RHS). Let $\mathcal{P}$ be the collection of all valid $s-t$ walks. The primal can now be written as

$$\max \sum_{p\in\mathcal{P}} x_p \qquad \text{subject to} \qquad \begin{cases} \sum_{p\in\mathcal{P}} x_p \mathbf{1}[(u,v)\in p] \leqslant c(u,v) & \text{for all } (u,v)\in E \\ x_p \geqslant 0 & \text{for all } p \in \mathcal{P}. \end{cases}$$

($\mathbf{1}[A]$ means the indicator on A, i.e., $\mathbf{1}[A](\omega) = 1$ iff $\omega \in A$.) In linear algebraic notations, this is

$$\max\{(\mathbf{1}_{\mathcal{P}})^T x \mid Ax \leqslant \text{capacity}, x \geqslant 0\}$$

where

- $x \in \mathbb{R}^{(|\mathcal{P}|\times 1)}$ is the amount of flow we send along each path,
- the "capacity vector" is the $\mathbb{R}^{(|E|\times 1)}$ vector of edge capacities, and
- $A \in \mathbb{R}^{|E|\times|\mathcal{P}|}$ is defined as $A_{(u,v),p} = \mathbf{1}[(u,v)\in p]$, i.e., the entry corresponding to edge (u,v) and path p is 1 if and only if (u,v) belongs to path p.

Now what does the dual look like? $\min\{\text{capacity}^T y \mid A^T y \geqslant \mathbf{1}_{\mathcal{P}}, y \geqslant 0\}$, or

$$\min \sum_{(u,v)\in E} c(u,v) y_{(u,v)} \qquad \text{subject to} \qquad \begin{cases} \sum_{(u,v)\in p} y_{(u,v)} \geqslant 1 & \text{for all } p \in \mathcal{P} \\ y_{(u,v)} \geqslant 0 & \text{for all } (u,v) \in E. \end{cases}$$

It is well known that the dual of max flow is min cut, and this is precisely the case! Restricting edge values $y_{(u,v)}$ to integers, we see that we may WLOG assume $y_{(u,v)} \in \{0,1\}$ for otherwise decreasing any excess value to 1 further reduces the objective function. Then, the dual constraint $\sum_{(u,v)\in p} y(u,v) \geqslant 1$ essentially says each path must have (at least) an edge assigned value 1. This precisely formulates an $s-t$ cut by grouping the s-side

endpoints of all edges with value 1 into one set, and the rest into the other. And the objective is the capacity of the cut. This concludes our reasoning.
(*Of course, note that if we were to compute the optimal value, this approach is absolutely abysmal, for the number of paths grows exponentially w.r.t the size of the graph. But it is nevertheless valid for showing the duality between max flow and min cut, which is our goal.*)

Solution to problem 4. (i) We first write the primal LP in standard linear algebraic notations. To do so we will rewrite the constraints as

$$x_i - \sum_{j=1}^{n-1} D_{a,i}(j)x_j \leqslant 1,$$

or

$$\sum_{j=1}^{n-1} \delta_{a,i}x_j \leqslant 1, \qquad \text{where} \qquad \delta_{a,i} = \begin{cases} D_{a,i}(j) & \text{if } i = j \\ 1 - D_{a,i}(j) & \text{otherwise.} \end{cases}$$

This tells us we want to design massive constraint matrix of dimension $(k(n-1)) \times (n-1)$ as follows:

$$A = \begin{bmatrix} 1 - D_{1,1}(1) & -D_{1,1}(2) & \cdots & -D_{1,1}(n-1) \\ -D_{1,2}(1) & 1 - D_{1,2}(2) & \cdots & -D_{1,2}(n-1) \\ \vdots & \vdots & \cdots & \vdots \\ 1 - D_{2,1}(1) & -D_{2,1}(2) & \cdots & -D_{2,1}(n-1) \\ \vdots & \vdots & \ddots & \vdots \\ -D_{k,n-1}(1) & -D_{k,n-1}(2) & \cdots & 1 - D_{k,n-1}(n-1) \end{bmatrix}.$$

Dualizing the LP, we obtain

$$\min(1, \cdots, 1)^T (y_{1,1}, y_{1,2}, \cdots, y_{1,n-1}, y_{2,1}, \cdots, y_{k,n-1})$$

subject to

$$A^T y = (1, 0, \cdots, 0) \qquad \text{and} \qquad y \geqslant 0$$

for matching dimensions (equation above because x_i is assumed to be real valued). The dual objective is $\max \sum_{a,i} y_{a,i}$. The first coordinate of the constraint suggests that the dot product of the first column of A (or the first row of A^T) with y is $\leqslant 1$:

$$\begin{aligned} 1 &= (1 - D_{1,1}(1))y_{1,1} - D_{1,2}(1) - \cdots - (1 - D_{2,1}(1))y_{2,1} - \cdots - D_{k,n-1}(1)y_{k,n-1} \\ &= \sum_{a \in [k]} y_{a,1} - \sum_{a,i} D_{a,i}(1)y_{a,i}. \end{aligned}$$

The other coordinates of the constraint are all zeros, so

$$\sum_{a \in [k]} y_{a,j} - \sum_{a,i} D_{a,i}(j)y_{a,i} = 0 \qquad \text{for } j \in \{2, \cdots, n-1\}.$$

and similarly

$$\min \sum_{a,i} y_{a,i} \qquad \text{subject to} \qquad \begin{cases} \sum_a y_{a,j} - \sum_{a,i} D_{a,i}(j)y_{a,i} = 1 & \text{for } j = 1 \\ \sum_a y_{a,j} - \sum_{a,i} D_{a,i}(j)y_{a,i} = 0 & \text{for } j \geqslant 2. \end{cases}$$

(ii) To prove this, we first note that by setting $x_i := \mu_i$ we obtain a feasible primal solution where x_i denotes the expected number of steps to reach state n from state i. This is because a Bellman-like argument states that

$$x_i \leqslant \min_j 1 + \sum_{j=1}^{n-1} D_{a,i}(j)x_j,$$

namely, the expected number of steps to get from state i to n is 1 (from current state) plus the weighted expected values from any other state to state n.

So it remains to prove that our dual can also attain the objective value of μ_1. If we use $y_{a,i}$ to represent the expected number of actions (landing at state i, using action a), then $\sum_{a,i} y_{a,i} = \mu_1$, the total expected number of steps to reach state n from 1.

For $j \geqslant 2$, $\sum_a y_{a,j}$ is the expected number of times we perform any action starting from state j, whereas $\sum_{a,i} D_{a,i}(j) y_{a,i}$ is the expected number of times we perform an action that lands in state j. For $j \geqslant 2$, these two quantities are equal, since there exists a one-to-one correspondence between states landing on j and acting on j. However, when $j = 1$, the difference is 1, since we initially start the game on state 1 — this technically is an "action" entering state 1 but is not reflected in the second sum. Therefore, the dual constraints are satisfied, and we have found a primal solution and a dual solution whose objective values match. By strong duality, $x_i = \mu_i$ is optimal.

Solution to problem 5. $\log\Delta$ suggests a binary search w.r.t. $[1,\Delta]$. That means with a given bottleneck, we just need an algorithm that runs in $O(n^3)$ to find *any* perfect matching. $\mathcal{O}(n)$ iterations of BFS will suffice, since the graph has $\mathcal{O}(n^2)$ edges.

Algorithm 1: Bottleneck Matching Problem in $\mathcal{O}(n^3 \log\Delta)$

```
1  Inputs: $G = (X \cup Y, X \times Y)$ with edge costs $w : X \times Y \to \mathbb{N} \cap [1,\Delta]$; $\Delta \in \mathbb{N}$
2  Initialization: Set low = 1, high = $\Delta$        # binary search with threshold
3  while low ⩽ high do
4      mid ← ⌊(low + high)/2⌋
5      define directed graph $G' = (V, E')$ where
           $E' = \{(u \to v) : (u,v) \in X \times Y, w(u,v) \leqslant \text{mid}\}$    # low weight edges, made directed
                 $\cup \{(v \to u) : (u,v) \in X \times Y\}$                           # augmented reverse edges
       initialize all vertices in $X$ and $Y$ as unmatched
6      for each vertex $v \in X$ do
7          run BFS/DFS on $G'$
8          if a path of length $2n$ is found then
9              perfect matching in $G'$ exists    # this path must alternate between $X$ and $Y$ so it can be
                                                    viewed as a perfect matching
10             break
11 # updating binary search range
12 if a perfect matching in G' is found then
13     high ← mid − 1
14 else
15     low ← mid + 1
16 return the most recently found perfect matching
```

Note that based on how we construct each $G' = (V, E')$, if a path of length $2n$ is found, then it means that edges along this path must be alternating between forward and backward (augmented) directions. This ensures that all n forward edges all have different starting vertices (in X) and ending vertices (in Y), so this effectively provides a perfect matching whose maximum edge cost is bounded by `mid`. The binary search ensures that by

the end of the algorithm we obtain an exact, tight threshold.
Runtime? Binary runs a total of $\mathcal{O}(\log \Delta)$ iterations. In each iteration constructing the graph takes $\mathcal{O}(n^2)$, and we run $\mathcal{O}(n)$ iterations of BFS on this graph, each of which takes $\mathcal{O}(n^2)$. So $\mathcal{O}(n^3)$ for each outer loop iteration. Clearly not optimal, but it gets the job done.