Homework 5

Haoyu Zhen

May 27, 2022

1 Konig-Egervay Theorem

(a)

For any given matching $\mathcal{M}\colon E \to \{0,1\},\ e \mapsto x_e$. Simply, x_e be a boolean value which denote whether the edge is used in the matching. Thus the IP problem reads

$$\begin{aligned} \max_{\mathcal{M}} \quad & \sum_{e\in E} x_e \\ \text{subject to} \quad & \sum_{e=(u,v)} x_e \le 1, \text{ for any } v \in V \\ & x_e \in \{0,1\} \text{ , for any } e \in E \end{aligned}$$

where the second line represents that every vertex will be matched at most once. Relaxing the IP, the third line convert to $0 \le x_e \le 1$. And $x_e \le 1$ holds naturally by $\sum_{e=(u,v)} x_e \le 1$. Finally we get the identical LP-relaxation.

(b)

The dual of the above linear program reads:

$$\begin{aligned} \min_{\mathcal{M}} \quad & \sum_{v\in V} y_v \\ \text{subject to} \quad & \sum_{e=(u,v)} y_v \ge 1, \text{ for any } e=(u,v) \in E \\ & y_v \ge 0 \qquad \text{, for any } v \in V \end{aligned}$$

Since y_v denote the weight/"probability" under which we choose vertex v into the cover set, the function above represents the fractional version of the minimum vertex cover problem.

(c)

Proof by mathematical induction:

- Suppose that every $k \times k$ submatrix of A 's determinant is 0 and ± 1. Here we consider
- If a column of A_{k+1} is all-zero, then $|A_{k+1}| = 0$.
- If a column of A_{k+1} contains only one non-zero entry, then $|A_{k+1}| = A_k$.

- IF every column of A_{k+1} has 2 non-zero entry: exists a partition (V_1, V_2) of A_{k+1},

$$\sum_j \sum_{v \in V_1} a_{vj} - \sum_j \sum_{v \in V_2} a_{vj} = 0$$

  Thus A_{k+1} is linear dependant.

(d)

- By (a),(b) and (c), we have OPT(match) = OPT(primal) $\leq$ OPT(dual) = OPT(cover).
- Since the problem is linear, the strong duality holds.
- For the integral solution of those relaxation problem, $y_v \in \{0, 1\}$ and $x_e \in \{0, 1\}$. (If not, $\sum_v y_v$ is not minimum and $\sum_e x_e \geq 1$.)

Then we have OPT(primal) = OPT(dual) which we prove the K-E Thm.

(e)

Let G be a fully connected graph with 4 vertices. Then the minimum cover's size is 1 while the size of the maximum matching is 2.

2 Money

Intutively, a tree has $|V| - 1$ edges. I will construct the tree by algorithm 1.

Algorithm 1 Construct the payment tree

Input: A directed graph $G = (V, E, w)$

Convrt G into a undirected one (with the same weight and topology), denoted by $T = (V, E', w')$
while T has a cicle c **do**
  Find the edge $\{u, v\} \in E'$ which has the minimum weight w'_0 in c.
  Suppose $(u, v) \in E$ and $p = c - \{\{u, v\}\}$ $\#\{u, v\} = \{v, u\}$ *while* $(u, v) \neq (v, u)$
  Remove $\{u, v\}$
  for edge $\{a, b\}$ in p (get path from u to v) **do**
    if $(a, b) \in E$ **then**
      $w'_{ab} \leftarrow w'_{ab} + w'_0$
    else
      $w'_{ab} \leftarrow w'_{ab} - w'_0$ (≥ 0) $\#(b, a) \in E$
    end if
  end for
end while
return T

The process of generating T is sound by following reasons:

- The algorithm could stop: every step there will be at least one edge to be removed.
- That A owes B is equivalent to that A owes C, C owes D ... and E owes B.
- Every weight is non-negative.

3 Bounded Network Flow

Reference: https://courses.engr.illinois.edu/cs498dl1/sp2015/notes/25-maxflowext.pdf.

(a)

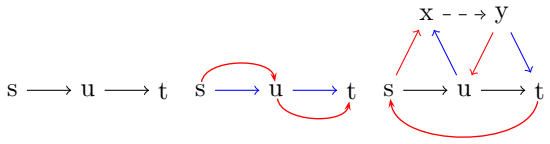
I design algorithm 2:

Algorithm 2 The existence of feasible flow

Input: A weighted graph $G = (V, E, w)$, d_e, f_e, c_e $(\forall e \in E)$ and vertices s, t (source and sink)

1: $V \leftarrow V \cup \{x, y\}$
2: $E \leftarrow E \cup \{(t, s)\}$ and $w_{(t,s)} = +\infty$
3: **for** $e = (u, v) \in E$ **do**
4: $\quad w_e \leftarrow c_e - d_e$
5: $\quad E \leftarrow E \cup \{(u, x)\}$ and $w_{(u,x)} \leftarrow d_e$
6: $\quad E \leftarrow E \cup \{(y, v)\}$ and $w_{(y,v)} \leftarrow d_e$
7: **end for**
8: **return** $\mathbb{1}\left[\text{MaxFlow}(y, x, G) = \sum_{e \in E} d_e\right]$ *#where $\mathbb{1}$ is an indicator.*

The correctness of algorithm 2 holds because:



Firsly, we could split an edge into a blue one and a red one where $w_r = d_e$ and $w_b = c_e - d_e$. To get a feasible flow, red edges in the new graph should be full.
Since the constraint should be satisfied, we could desgine the third graph above via introducing 2 vertices x, y and an edge (t, s) with weight $+\infty$. Now we just need to run a max flow algorithm from y to x. Naturally, if the max flow equals $\sum_{e \in E} d_e$, then the constraint is satisfied.

(b)

Firstly, we need check whether graph G is feasible by algorithm 2. This step generate a new graph H. Then we could run Ford-Fulkerson algorithm on G with some refinement:

- Source: s, sink: t.
- Initialization:

  $$f_{(u,v)} = g_{(u,v)} + d_{(u,v)}$$

  where $g_{(u,v)}$ is the flow at edge (u, v) in graph H.
- The updating rule becomes:

  $$c_f(u, v) = \begin{cases} c_e - f_e & \text{if } (u, v) \in E \\ f_e - d_e & \text{if } (v, u) \in E \end{cases}$$

  where $(u, v) = e$.

The initialization means that we begin with a feasible way. And the updating rule ensures every flow is feasible because every edge is positive. Then like the proof of FF Algo., we could prove that this new algorithm could find a maximum feasible flow with lower bounds.

4 Misc

About a day. Dfficulty 3. No collaborator.