#4 Assignment - CMPT 405

Luiz Fernando Peres de Oliveira - 301288301 - lperesde@sfu.ca

November 16, 2018

#1

Let w_{ij} be the weight of every $(i, j) \in E$ and x_{ij} be variables such that $x_{ij} = 1$ if the shortest path contains $i \to j$ and $x_{ij} = 0$, otherwise. The shortest path from a source $s \in V$ to a target $t \in V$ in a weighted graph $G = (V, E, w)$ can be found by minimizing the summation of $w_{ij}x_{ij}$ for every (i, j). See below:

$$x_{ij} = \begin{cases} 1 & \text{if the shortest path contains } i \to j \\ 0 & \text{otherwise} \end{cases}$$

By the principle of network flow, we have that for each single node i, the amount of a flow f_i is equal the difference between the amount of outgoing flow from i and the amount of incoming flow to i:

$$f_i = \sum_j x_{ij} - \sum_k x_{ki}$$

As we are looking for the shortest path from s to t, we know that our network will "travel" from the source to the target, cancelling any flow f_u for single vertices u between s and t in our network, where $u \neq s$ and $u \neq t$. Because there is no incoming flow in s, $f_s = 1$. Likewise, because there is no outgoing flow in t, $f_t = -1$. Thus:

$$f_i = \begin{cases} 1 & \text{if } i = s \\ -1 & \text{if } i = t \\ 0 & \text{otherwise} \end{cases}$$

Assuming that $x_{ij} \geq 0$, the linear program for the shortest problem is:

$$min \sum_{(i,j) \in E} w_{ij}x_{ij}$$

The resulting dual will have one variable y_u for each vertex u in the graph. The values of y have the constraint that $y_j - y_i \leq w_{ij}$ and the objective function is the maximization of $y_s - y_t$:

$$max\ y_s - y_t$$

$$y_j - y_i \leq w_{ij}\ ,\ \forall (i,j) \in E$$

Dual Encoding: The dual can be interpreted as the encoding of Bellman-Ford, because when BF terminates, it has computed for each vertex j a value y_j, such that for each edge $(i,j) \in E$, we have the same constraints as the dual: $y_j \leq y_i + w_{ij}$. The objective function is also the maximization of $y_s - y_t$.

#2

In a similar way of question #1, let w_e be the weight of every $e \in E$ and x_e be $0-1$ variables such that $x_e = 1$ if the edge e is in the matching and $x_e = 0$, otherwise.

$$x_e = \begin{cases} 1 & \text{inclusion of edge } e \text{ in the matching} \\ 0 & \text{otherwise} \end{cases}$$

We need to choose at each step an augmenting path p that produces the largest possible increase in weight so that the matching obtained by flipping the edges has maximum weight.

The objective function maximizes the weight of all edges e in the matching and, because we are augmenting paths in our matchings, we use constraints to limit one edge per vertex so that the path is created in the form $x_e \leq 1$, for all vertices u, such that $e = (u,v)$. The linear program is then:

$$max\ \sum_e w_e x_e$$

$$\sum_{e=(u,v)} x_e \leq 1\ ,\ \forall u \in V$$

#3

To prove that the constraint matrix in previous example is totally unimodular for bipartite graphs, we must prove that every entry is equal to $1, 0$ or -1.

Let x_{ij} be an alias for x_e in the previous example, such that $e = (i,j)$. Each row i in x represent a vertex v_i and each column j represents an edge $e = (i,j)$. $x_{ij} = 1$ only if our previous algorithm included (i,j) in the matching.

Proof. Consider an arbitrary square submatrix x' of x. The goal here is to show that the determinant of x' is in $\{1, 0, -1\}$.

Case 1: x' has a column with only 0. Then the determinant of $x' = 0$.
Case 2: x' has a column with only 1. By induction, x'' has determinant equal $1, 0$ or -1 and so does x'.

$$x' = \begin{bmatrix} 1 & ... \\ 0 & x'' \end{bmatrix}$$

Case 3: Each column of x' has exactly two 1. Because we are dealing with a bipartite graph, we have two distinct sets $x^{[+1]}, x^{[-1]}$, $x^{[+1]} \cap x^{[-1]} = \emptyset$, such that each edge $e = (i, j) \in x$ will have i in one of this sets and j in the other set (otherwise it would not be bipartite). Because the rows are linearly dependent, by multiplying $+1$ in the rows in $x^{[+1]}$ and -1 on the columns in $x^{[-1]}$, the determinant of $x' = 0$.

$$x' = \begin{bmatrix} x^{[+1]} \\ x^{[-1]} \end{bmatrix}$$

$\square$

Counterexample for non-bipartite graphs:
In spite of working for bipartite graphs, the same does not hold for non-bipartite graphs. It can be easily proved by a simple counterexample (use the same linear program as described on #2): $G = (V, E)$, $V = \{a, b, c\}$, $E = \{(a, b) = 1/3, (b, c) = 1/3, (a, c) = 1/3\}$.

#4
#5