CS170–Spring 2019 — Homework 11 Solutions

Ran Liao, SID 3034504227

April 14, 2019

Collaborators:Jingyi Xu, Renee Pu

1 Study Group

Name	SID
Ran Liao	3034504227
Jingyi Xu	3032003885
Renee Pu	3032083302

2 Bipartite Vertex Cover

(a) **Main Idea**

Since G is a bipartite graph, we can split V into two disjoint set L and R such that there's no edge linking two vertices in the same set.

Then, run max-flow algorithm on following graph G'. Add nodes s and t into the graph. Add a directed edge from s to every node in L and from every node in R to t. Convert the undirected edges between L and R to directed edges. Set the capacity of every edge to 1. As textbook mentioned, this max-flow can be converted into a max-matching.

A minimum vertex cover can be constructed as follows. Let U be the set of unmatched vertices in L, and Z be the set of vertices that are either in U or are connected to U by alternating paths(a path of odd length that starts and ends with a non-covered vertex, and whose edges alternate between matched edges and unmatched edges). Let $K = (L \setminus Z) \cup (R \cap Z)$. Z is the minimum vertex cover we're looking for.

(b) **Proof of Correctness**

Suppose that M is a maximum matching. No vertex in a vertex cover can cover more than one edge of M. So $|M|$ is a lower bound for vertex cover and if we can construct a vertex cover with $|M|$ vertices, it must be a minimum cover.

Every edge e in E either belongs to an alternating path, or it has a left endpoint in K. If e is matched but not in an alternating path, then its left endpoint cannot be in an alternating path (because two matched edges can not share a vertex) and thus belongs to $(L \setminus Z)$. Alternatively, if e is unmatched but not in an alternating path, then its left endpoint cannot be in an alternating path, for such a path could be extended by adding e to it. Thus, K forms a vertex cover with size $|M|$.

3 Direct Bipartite Matching

(a) **Main Idea**

To prove M is a maximum matching if and only if there does not exist an alternating path, it's equivalent to prove M is **not** a maximum matching if and only if there does **exist** an alternating path.

Suppose M is not a maximum matching, run the max-flow algorithm on this graph. At least one augment path can be found. Denote it as $s \to v_1 \to v_2 \to \cdots \to v_{2n} \to t$. When i is odd, $v_i \in L$ and $v_i \to v_{i+1} \in E \setminus M$, otherwise $v_i \in R$ and $v_i \to v_{i+1} \in M$. The number of edges in this path is odd. Therefore, it is an alternating path.

Suppose there does exist an alternating path. Denote it as $v_1 \to v_2 \to \cdots \to v_{2n}$. By definition, $(v_1, v_2), (v_3, v_4) \cdots (v_{2n-1}, v_{2n}) \in E \setminus M$ and denote it as E_1. Denote the remaining edges as E_2. We can construct a larger flow M' as follows.

$$M' = (M \cup E_1) \setminus E_2$$

Therefore, M cannot be a maximum matching.

(b) (i) **Main Idea**

Create a dumb node s and add edge (s, v) if v is an unmatched vertex. Use a global variable *depth* to record the depth of the search tree. If *depth* is odd, only unmatched edges can be explored. Otherwise only matched edges can be explored. Run BFS starting from s.

(ii) **Proof of Correctness**

The alternating path will start from an unmatched point and alternating between matched and unmatched edges. The variable *depth* will be helpful to determine whether to explore matched edges or unmatched edges.

(iii) **Runtime Analysis**

The modified BFS will not change overall runtime. Therefore it is $O(|V| + |E|)$.

(c) (i) **Main Idea**

Find an alternating path use the algorithm in part b and construct a larger matching use method mentioned in part a. Keep doing this until there's no alternating path exists.

(ii) **Proof of Correctness**

As proved in part a, M is a maximum matching if and only if there does not exist an alternating path. And if there's an alternating path, part b algorithm will find it.

(iii) **Runtime Analysis**

The algorithm in part b will be run at most $O(|V|)$ times, and each run will cost $O(|V| + |E|)$. Therefore, the overall runtime is $O((|V| + |E|)|V|) = O(|V||E|)$.

4 Zero-Sum Battle

(a)

$$\max p$$

$$p \leq -10x_1 + 4x_2 + 6x_3 \text{ (payoff when trainer B chooses the ice Pokemon)}$$

$$p \leq 3x_1 - 1x_2 - 9x_3 \text{ (payoff when trainer B chooses the water Pokemon)}$$

$$p \leq 3x_1 - 3x_2 + 2x_3 \text{ (payoff when trainer B chooses the fire Pokemon)}$$

$$x_1 + x_2 + x_3 = 1$$

$$x_1 \geq 0$$

$$x_2 \geq 0$$

$$x_3 \geq 0$$

The optimal strategy is $(0.335, 0.563, 0.102)$ and the payoff is -0.48.

(b)

$$\min p$$

$$p \geq -10y_1 + 3y_2 + 3y_3 \text{ (payoff when trainer A chooses the dragon Pokemon)}$$

$$p \geq 4y_1 - 1y_2 - 3y_3 \text{ (payoff when trainer A chooses the steel Pokemon)}$$

$$p \geq 6y_1 - 9y_2 + 2y_3 \text{ (payoff when trainer A chooses the rock Pokemon)}$$

$$y_1 + y_2 + y_3 = 1$$

$$y_1 \geq 0$$

$$y_2 \geq 0$$

$$y_3 \geq 0$$

The optimal strategy is $(0.268, 0.323, 0.409)$ and the payoff is -0.48.

5 Domination

(a) It should be 0 since choosing E instead will always give a better payoff.

(b) It should also be 0 since choosing B instead will always give a better payoff(column player wants to minimize the payoff).

(c) Both of them should be $(0.5, 0.5)$, since they are completely symmetric