Assignment 6
201318013229054

Jun Zhang

January 8, 2014

1 Problem 1

For the undirected bipartite graph $G = (V, E)$, partition V into two sets L, R. Then construct a network $(G' = (V', E'), s, t, c)$ as follows:

(1) The vertex set is $V' = V \cup \{s, t\}$, where s and t are two new vertices.

(2) E' contains a directed edge (s, u) for every $u \in L$; a directed edge (u, v) for every edge $(u, v) \in E$, where $u \in L$ and $v \in R$; and a directed edge (v, t) for every $v \in R$.

(3) Each edge (s, u) for every $u \in L$ has a capacity of the positive weight of vertex u; each edge (u, v) for every edge $(u, v) \in E$, where $u \in L$ and $v \in R$ has a capacity of $+\infty$; each edge (v, t) for every $v \in R$ has a capacity of the positive weight of vertex v.

The cut $(S, \bar{S})$ is defined as follows: A cut partition V into two sets $S, \bar{S}$, where the source $s \in S$, and the sink $t \in \bar{S}$. Find a minimum-capacity cut $(S, \bar{S})$ in the network. Define $L_1 = L \cap S, L_2 = L \cap \bar{S}, R_1 = R \cap S, R_2 = R \cap \bar{S}$. Given such a cut, the corresponding (supposed) vertex cover will be $X = L_2 \cup R_1$. It is clear that $capacity(S, \bar{S}) = |X|$. Thus the optimal cover problem can be turned into the minimum cut problem.

Theorem 1.1. *For every cut $(S, \bar{S})$ of finite capacity, define the set $X = L_2 \cup R_1$, the X is a valid vertex cover and $|X| = capacity(S, \bar{S})$.*

Proof. The cut is finite capacity would mean that each of the infinity edges either starts inside the L_2 or ends in the R_1 (or both). So all the edges of the original graph at least either have one endpoint in L_2 or R_1. It is clear that $capacity(S, \bar{S}) = |X|$.

Theorem 1.2. *For every valid vertex cover X, define the cut $(S, \bar{S})$ as,*

$$S = \{u \in L : u \notin X\} \cup \{u \in R : u \in X\},\ \bar{S} = \{u \in L : u \in X\} \cup \{u \in R : u \notin X\}$$

The $(S, \bar{S})$ has finite capacity and $capacity(S, \bar{S}) = |X|$.

Proof. Assume the contrary that an edge going from u to v is cut by this cut and has infinite capacity. Then u must be in $L \quad S$ and v must be in $R \cap \bar{S}$. So neither u nor v are in X in the original graph but there is an edge between these two vertices. That means that X is not a vertex cover which is a contradiction. It is clear from the way the cut is defined that its capacity is $|X|$.

2 Problem 2

2.1 Question a

The following matrix is not re-arrangeable:

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

2.2 Question b

For the $n \times n$ matrix M, construct a bipartite graph $G = (V, E)$ as follows:

(1) The vertex set V is partitioned into two sets R, C. The set R has n vertices, each $r_i \in R$ represents Row i; The set C has n vertices, each $c_j \in C$ represents Column j.

(2) E contains a undirected edge (r_i, c_j) for every $r_i \in R$ and $c_j \in C$, where $m_{i,j} = 1$.

Swapping rows can reorder the sequence of rows, swapping columns can reorder the sequence of columns. After several swapping actions, all the diagonal entries of M may be equal to 1. That means there exist a perfect matching between rows and columns.

Theorem 2.1. *If there exist a perfect matching of the bipartite graph, the matrix is re-arrangeable.*

Proof. There are n edges in the perfect matching of the bipartite graph. For the k^{th} edge (r_i, c_j), swap original Row i with Row k and original Column j with Column k of matrix m. Because (r_i, c_j) is an edge, the original $m_{i,j} = 1$. After swapping, $m_{k,k} = 1$, then Row k and Column k is fixed. Thus, after n pairs of swapping, the matrix is re-arranged. (Original Row i means the original sequence number of the row is i, Row i means the i^{th} row now. The column number is defined analogously. These definitions are used in the rest of the discussion for this problem.)

Theorem 2.2. *If the matrix is re-arrangeable, there exist a perfect matching of the bipartite graph.*

Proof. The matrix is re-arrangeable, so after several swapping actions, all the diagonal entries of M may be equal to 1. For the k^{th} diagonal entry $m_{k,k} = 1$, suppose its original row number is i, and original column number is j. The original $m_{i,j} = 1$, so match r_i with c_j in the bipartite graph. After n matches, it becomes a perfect matching of the bipartite graph.

Now the problem becomes the perfect matching problem in a bipartite graph. We can just apply *Hungarian Algorithm*[1] to solve the perfect matching problem, whose complexity is $O(n^3)$. Because the corresponding lecture is about network flow, we use network flow to solve the perfect matching problem.

For the undirected bipartite graph $G = (V, E)$, V is partitioned into two sets R, C. Then construct a network $(G' = (V', E'), s, t, c)$ as follows:

(1) The vertex set is $V' = V \cup \{s, t\}$, where s and t are two new vertices.

(2) E' contains a directed edge (s, u) for every $u \in R$; a directed edge (u, v) for every edge $(u, v) \in E$, where $u \in R$ and $v \in C$; and a directed edge (v, t) for every $v \in C$.

(3) Each edge has capacity 1.

Then try to find a maximum flow whose flow value is equal to n. Using network flow to solve the perfect matching problem has been discussed in the lecture. The complexity is $O(n^3)$.

3 Problem 3

Getting the max-flow or min-cut in the G, we have the residual graph G_f. Find all the vertices reachable from s in the residual graph G_f and we've found a min-cut $(S, \bar{S})$ in G. Look at the same residual graph, starting at t. Find the group of vertices reachable from t in the reverse direction of the directed edges (meaning all the vertices which can reach t). This group is also a min-cut $(\bar{T}, T)$. If that cut is identical to the original cut, which is $(T = \bar{S})$, then there is only one min-cut. Otherwise, the min-cut is not unique. Searching the vertices reachable from s and the vertices which can reach t by using BFS or DFS takes $O(n^2)$ (Adjacency Matrix).

Proof. If $S \cup T \neq V$, then there exists vertex $x \notin S \cup T$. In G_f, s can not reach x and x can not reach t. In G, $f_{in}(x) = c_{in}(x)$, $f_{out}(x) = c_{out}(x)$ and $f_{in}(x) = f_{out}(x)$, so we have $c_{in}(x) = c_{out}(x)$. The min-cut $(S, \bar{S})$ has two cases that $x \in S$ or $x \in \bar{S}$. The min-cut is not unique.

[1] Kuhn H W. The Hungarian method for the assignment problem[J]. Naval research logistics quarterly, 1955, 2(1-2): 83-97.

4 Problem 4

For n weighted open intervals. The i^{th} interval covers (a_i, b_i) and weighs w_i. Then construct a weighted network $(G = (V, E), s, t, c, w)$ as follows:

(1) The vertex set is $V = \left(\bigcup_{i=1}^{n}\{a_i, b_i\}\right) \cup \{s, t\}$.

(2) The edge set is $E = E_0 \cup E_1 \cup E_2 \cup E_3$. E_0 contains a directed edge (a_i, b_i) for every interval. E_1 contains directed edges (b_i, a_j) where $(b_i \leq a_j)$. E_2 contains a directed edge (s, a_i) for every a_i and E_3 contains a directed edge (b_i, t) for every b_j.

(3) Each edge $(a_i, b_j) \in E_0$ for every interval has a capacity of 1 and a weight of $-w_i$ (Use the opposite value, so that we can apply the minimum cost flow algorithm, which has been discussed in the lecture, to solve the problem). Each edge in $E_1 \cup E_2 \cup E_3$ has a capacity of 1 (or $+\infty$, any value not less than 1 is OK) and a weight of 0.

Then apply the minimum cost flow algorithm to find a flow with flow value k and the cost is minimized on the network. The opposite value of the minimum cost result is the maximum weight of feasible intervals.

Proof. In the flow, every s-t path goes through one or more edges in E_0. Because it should go through an edge in E_1 every time before it goes through an extra edge in E_0, the path represent a feasible interval arrangement which covers any point in the real axis no more than one time. The flow value k constrain the number of such feasible interval arrangements to be no more than k. Finally, the opposite value of the cost result is the weight of feasible intervals.

The complexity of the algorithm is $O(n^3 \log n \min\{\log nC, n^2 \log n\})$ [2] (suppose $|E| = |V|^2$).

5 Problem 5

For the n little dogs and n kennels, construct a weighted bipartite graph $G = (V, E)$ as follows:

(1) The vertex set V is partitioned into two sets D, K. The set D has n vertices, each $d_i \in D$ represents Dog i; The set K has n vertices, each $k_j \in K$ represents Kenel j.

(2) E contains a undirected edge (d_i, k_j) for every $d_i \in D$ and $k_j \in K$.

(3) Each edge (d_i, k_j) weights the travel fee for d_i to arrive at k_j.(The travel fee is fixed for a given dog (x_i, y_i) to arrive at a given kennel (x_j, y_j), that is $|x_i - x_j| + |y_i - y_j|$.)

Then the problem becomes the minimum weighted perfect matching problem in a bipartite graph. We can apply *Kuhn-Munkres Algorithm*[3] to solve the minimum weighted perfect matching problem, whose complexity is $O(n^4)$ (or $O(n^3)$ by using a slack value s_j for every $k_j \in K$). Because the corresponding lecture is about network flow, we use network flow to solve the minimum weighted perfect matching problem.

For the undirected bipartite graph $G = (V, E)$, V is partitioned into two sets D, K. Then construct a weighted network $(G' = (V', E'), s, t, c, w)$ as follows:

(1) The vertex set is $V' = V \cup \{s, t\}$, where s and t are two new vertices.

(2) E' contains a directed edge (s, d_i) for every $d_i \in D$; a directed edge (d_i, k_j) for every edge $(d_i, k_j) \in E$, where $d_i \in D$ and $k_j \in K$; and a directed edge (k_j, t) for every $k_j \in K$.

[2] Goldberg A V, Tarjan R E. Finding minimum-cost circulations by canceling negative cycles[J]. Journal of the ACM (JACM), 1989, 36(4): 873-886.

[3] Munkres J. Algorithms for the assignment and transportation problems[J]. Journal of the Society for Industrial and Applied Mathematics, 1957, 5(1): 32-38.

(3) Each edge (s, d_i) and (k_j, t) has a capacity of 1 and a weight of 0. Each edge (d_i, k_j) has a capacity of 1 and a weight of the travel fee for d_i to arrive at k_j.

Then apply the minimum cost flow algorithm to find a flow with flow value n and the cost is minimized on the network. The complexity of the algorithm is $O(n^3 \log n \min\{\log nC, n^2 \log n\})$.

6 Problem 6

For the weighted undirected graph $G = (V, E)$, we construct a network $(G' = (V', E'), s, t, c)$ as follows:

(1) The vertex set is $V' = V \cup V_e \cup \{s, t\}$, where s and t are two new vertices, and V_e contains a vertex v_{ij} for every edge $(i, j) \in E$.

(2) E' contains a directed edge (s, v_{ij}) for every $v_{ij} \in V_e$; a pair of directed edges (v_{ij}, v_i) and (v_{ij}, v_j) for every $v_{ij} \in V_e$; and a directed edge (v_i, t) for every $v_i \in V$.

(3) Each edge (s, v_{ij}) has a capacity of the weight of edge $(i, j) \in E$. Each pair of edges (v_{ij}, v_i) and (v_{ij}, v_j) has a capacity of $+\infty$. Each edge (v_i, t) has a capacity of α.

Then find a minimum-capacity cut $(C, \bar{C})$ of finite capacity. Let $S = C \cap V$.

Theorem 6.1. *$v_{ij} \in C$ if and only if both $v_i \in C$ and $v_j \in C$.*

Proof. The cut is finite capacity would mean that each of the infinity edges either starts inside the $V_e \cap \bar{C}$ or ends in the $V \cap C$ (or both). If $v_{ij} \in V_e \cap C$, then both $v_i \in V \cap C$ and $v_j \in V \cap C$. If both $v_i \in V \cap C$ and $v_j \in V \cap C$, assume $v_{ij} \in V_e \cap \bar{C}$, we can get a new cut $(C', \bar{C}')$ that $C' = C \cup \{v_{ij}\}$. Because v_{ij} only has edges (s, v_{ij}), (v_{ij}, v_i) and (v_{ij}, v_j), the new cut $(C', \bar{C}')$ has *capacity*(s, v_{ij}) less capacity than the cut $(C, \bar{C})$, which is a contradiction.

Theorem 6.2. *capacity*$(C, \bar{C}) = e(V) - e(S) + \alpha|S|$

Proof. The cut $(C, \bar{C})$ cuts each edge (v_i, t) for every $v_i \in V \cap C$, each edge (s, v_{ij}) for every $v_{ij} \in V_e \cap \bar{C}$ and infinity edges either starts inside the $V_e \cap \bar{C}$ or ends in the $V \cap C$. Thus,

$$\mathit{capacity}(C, \bar{C}) = \sum_{v_{ij} \in V_e \cap \bar{C}} \mathit{capacity}(s, v_{ij}) + \sum_{v_i \in V \cap C} \mathit{capacity}(v_i, t) = e(V) - e(S) + \alpha|S|$$

Theorem 6.3. *There is a subset S with cohesiveness larger than α if and only if the min-cut $(C, \bar{C})$ in G has capacity less than $e(V)$.*

Proof. The min-cut in G has capacity less than $e(V)$ then $C \neq \{s\}$, then $|S| > 0$.

$$\begin{aligned} \mathit{capacity}(C, \bar{C}) = e(V) - e(S) + \alpha|S| &< e(V) \\ \Leftrightarrow \quad e(S) &> \alpha|S| \\ \Leftrightarrow \quad e(S)/|S| &> \alpha \end{aligned}$$

Theorem 6.4. *There is a subset S with cohesiveness $= \alpha$ if and only if the min-cut $(C, \bar{C})$ in G has capacity $= e(V)$ and $C \neq \{s\}$.*

Proof. If $C \neq \{s\}$ then $|S| > 0$.

$$\begin{aligned} \mathit{capacity}(C, \bar{C}) = e(V) - e(S) + \alpha|S| &= e(V) \\ \Leftrightarrow \quad e(S) &= \alpha|S| \\ \Leftrightarrow \quad e(S)/|S| &= \alpha \end{aligned}$$

Suppose there are n vertices and m edges in G, then there are $|V'| = m+n+2$ vertices and $|E'| = 3m+n$ edges in G'. The complexity of the algorithm is $O(m^3)$.