Problem 1: Graph cohesiveness (KT 7.46)

In sociology, one often studies a graph G in which nodes represent people and edges represent those who are friends with each other. Let's assume for purposes of this question that friendship is symmetric, so we can consider an undirected graph.

Now suppose we want to study this graph G, looking for a "close-knit" group of people. One way to formalize this notion would be as follows. For a non-empty subset S of nodes, let $e(S)$ denote the number of edges in S-that is, the number of edges that have both ends in S. We define the cohesiveness of S as $e(S)/|S|$. A natural thing to search for would be a set S of people achieving the maximum cohesiveness.

1.1: Give a polynomial-time algorithm that takes a rational number α and determines whether there exists a set S with cohesiveness greater than α.

Algorithm:

Let the original graph $G = (V, E)$, where $V = \{v_1, ..., v_i, ..., v_n\}$ denotes people. Let $U = \{u_{ij} \mid (v_i, v_j) \in E\}$ denote friendship between v_i and v_j. Let the flow graph $G' = (V', E', c)$, where

$$V' = \{s, t\} \cup U \cup V$$
$$E' = \{s\} \times U \cup \{(u_{ij}, v_i), (u_{ij}, v_j) \mid u_{ij} \in U\} \cup V \times \{t\}$$

and the edge capacities are

$$c(s, u_{ij}) = |V|$$
$$c(u_{ij}, v_i) = c(u_{ij}, v_j) = \infty$$
$$c(v_i, t) = \alpha|V|.$$

All finite edge capacities are integers. We run the Ford-Fulkerson algorithm on G'. Let (A, B) be the min-cut. If $c(A, B) < |V||E|$, output True, otherwise output False.

Proof of correctness:

Let $E(S)$ denote edges that have both ends in S. Define a cut (A, B) to be normal if $U \cap A = E(V \cap A)$.

Claim 1. *Let (A, B) be a normal cut, then $c(A, B) = |V|(|E| - e(V \cap A) + \alpha|V \cap A|)$.*

Proof. Because (A, B) is a normal cut, only two types of edges cross from A to B: $\{(s, u_{ij}) \mid u_{ij} \in U \cap B\}$ and $\{(v_i, t) \mid v_i \in V \cap A\}$. So

$$\begin{aligned} c(A, B) &= \sum_{u_{ij} \in U \cap B} c(s, u_{ij}) + \sum_{v_i \in V \cap A} c(v_i, t) \\ &= \sum_{u_{ij} \in U} c(s, u_{ij}) - \sum_{u_{ij} \in U \cap A} c(s, u_{ij}) + \sum_{v_i \in V \cap A} c(v_i, t) \\ &= |V|(|E| - e(V \cap A) + \alpha|V \cap A|) \end{aligned}$$

□

Claim 2. *For every set S, there is a normal cut (A, B) s.t. $c(A, B) = |V|(|E| - e(S) + \alpha|S|)$.*

Proof. Let $A = \{s\} \cup S \cup \{u_{ij} \mid (v_i, v_j) \in E(S)\}$, $B = V' \setminus A$, then (A, B) is a normal cut. We know from Claim 1 that $c(A, B) = |V|(|E| - e(S) + \alpha|S|)$. □

Claim 3. *The min-cut of G is a normal cut.*

Proof. The max-flow is no greater than $|V||E|$, so the min-cut is finite. Suppose there is $u_{ij} \in B$ such that $\{v_i, v_j\} \subset A$, we can include u_{ij} into A and delete it from B. In such way the cut capacity is reduced by $|V|$, which is contradictory to the fact that we already have a min-cut. Hence the min-cut is a normal cut. □

Claim 4. *Let (A, B) be the min-cut. $c(A, B) < |V||E|$ iff there is a set S with $e(S)/|S| > \alpha$.*

Proof. $\Rightarrow$: Per Claim 1 and Claim 3, we have $c(A, B) = |V|(|E| - e(V \cap A) + \alpha|V \cap A|) < |V||E|$, so $e(V \cap A)/|V \cap A| > \alpha$.

$\Leftarrow$: For the purpose of contradiction, assume $c(A, B) = |V||E|$. Per Claim 2, we have a cut (A^*, B^*) with $c(A^*, B^*) = |V|(|E| - e(S) + \alpha|S|) < |V||E|$, which is contradictory to the fact that (A, B) is the min-cut. □

Time complexity:

Constructing G' takes $O(|V'| + |E'|) = O(2|V| + 4|E|)$ time. We run Ford-Fulkerson algorithm to find the min-cut, which takes $O((|V'| + |E'|)C)$ time where C is the max-flow. Since the max-flow can not be greater than $|V||E|$, the overall time complexity is $O(2|V|^2|E| + 4|V||E|^2)$.

1.2: Give a polynomial-time algorithm to find a set S of nodes with maximum cohesiveness.

Algorithm:

Let the algorithm for previous question be Algorithm 1. Algorithm 1 takes α and determines whether there is a set with cohesiveness greater than α. It's easy to modify it so that when it returns True, it also returns the exact cohesiveness $\beta = e(V \cap A)/|V \cap A| > \alpha$ and the set $V \cap A$.

Observe that $0 \leq e(S) \leq |E|$ and $0 < |S| \leq |V|$, so there are at most $|E||V| + 1$ possible cohesiveness value (if $|E| = 0$, the maximum cohesiveness is 0, so we assume $|E| > 0$). We can pre-process these values by sorting them, then simply apply binary search. For each α, we use Algorithm 1 to determine whether there is a set with cohesiveness greater than it. If the answer is True, we record β and the corresponding set. Finally, we return the set that corresponds to maximum β.

Proof of correctness:

Claim 5. *The above algorithm returns the set that corresponds to maximum cohesiveness.*

Proof. Since $|E| > 0$, the algorithm is guaranteed to find a set when checking $\alpha = 0$.

Suppose the algorithm returns a set S and there is a set S^* such that $e(S^*)/|S^*| > e(S)/|S|$. Then the algorithm should have discovered a better set when checking $\alpha = e(S)/|S|$, which causes contradiction. □

Time complexity:

Taking from the previous question, we have $\log(|E||V| + 1)$ checks, so the overall time complexity is $O((2|V|^2|E| + 4|V||E|^2)\log(|E||V|))$.

Problem 2: Remote sensors

Devise as efficient as possible algorithm for the following problem. You have n remote sensors s_i and $m < n$ base stations B_j. For $1 \leq j \leq m$, base station B_j is located at (x_j, y_j) in the two-dimensional plane. You are given that no two base-stations are less than 1 km apart (in standard Euclidean distance, $\sqrt{(x_j - x_k)^2 + (y_j - y_k)^2}$). All base stations have the same integer bandwidth capacity C. For $1 \leq i \leq n$, sensor s_i is located at (x_i, y_i) in the two-dimensional plane and has an integer bandwidth requirement of r_i, which can be met by assigning bandwidth on multiple base stations. Let $b_{i,j}$ be the amount of bandwidth assigned to sensor s_i on base station B_j. The assignment must meet the following constraints:

- No sensor may be assigned any bandwidth on a base station more than 2 km distance from it, i.e., if the distance from s_i to B_j is greater than 2, $b_{i,j} = 0$.
- The sum of all the bandwidth assigned to any remote sensor s_i must be at least r_i: for each $1 \leq i \leq n$, $\sum_j b_{i,j} \geq r_i$.
- The sum of all bandwidth assigned on base station B_j must be at most C: for each $1 \leq j \leq m$, $\sum_i b_{i,j} \leq C$.

Your algorithm should find a solution meeting the above constraints if possible, and otherwise output a message saying "No solution exists". Prove the correctness of your algorithm and discuss its time complexity.

Algorithm:

Let node set $B = \{B_1, ..., B_j, ..., B_m\}$ denote base stations and node set $S = \{s_1, ..., s_i, ..., s_n\}$ denote sensors. Let the flow graph $G = (V, E, c)$, where

$$V = \{s, t\} \cup B \cup S$$
$$E = \{s\} \times B \cup \{(B_j, s_i) \mid \sqrt{(x_j - x_i)^2 + (y_j - y_i)^2} \leq 2\} \cup S \times \{t\}$$

and the edge capacities are

$$c(s, B_j) = C$$
$$c(B_j, s_i) = \infty$$
$$c(s_i, t) = r_i.$$

We then run the Preflow-Push algorithm on G. If the max-flow is not equal to $\sum_i r_i$, output "No solution exists", otherwise, output $b_{i,j}$ as $f(B_j, s_i)$.

Proof of correctness:

Claim 1. *If there is a solution meeting the constraints, then the max-flow is* $\sum_i r_i$.

Proof. Let $b_{i,j}$ be the amount of bandwidth assigned to sensor s_i on base station B_j in the solution. Since for each $1 \leq i \leq n$, $\sum_j b_{i,j} \geq r_i$, we can set $f(B_j, s_i) \leq b_{i,j}$ such that $\sum_j f(B_j, s_i) = r_i$. Also, for each $1 \leq j \leq m$, $\sum_i b_{i,j} \leq C$, so $\sum_i f(B_j, s_i) \leq C$, i.e., capacity property holds for all (s, B_j) edges. Thus we find a flow equal to $\sum_i r_i$, and it is the maximum possible flow. $\square$

Claim 2. *If the max-flow is $\sum_i r_i$, then the algorithm outputs a solution meeting the constraints.*

Proof. Let $b_{i,j} = f(B_j, s_i)$, then for each $1 \leq i \leq n$, $\sum_j b_{i,j} = r_i$. Per flow properties, for each $1 \leq j \leq m$, we also have $f(s, B_j) = \sum_i b_{i,j} \leq C$. Since the distance constraint is satisfied when constructing G, our solution meets all constraints. $\square$

Time Complexity:

In constructing the graph, we iterate through n sensors for m base stations, which takes $O(mn)$ time. We run Preflow-Push algorithm to find the max-flow, which takes $O(|V|^3)$ where $|V|$ is the number of nodes. Since we have $m + n$ nodes and $m < n$, the overall time complexity is $O(n^3)$.

Problem 3: Scheduling in a medical consulting firm (KT 7.19)

For each of the next n days, the hospital has determined the number of doctors they want on hand; thus, on day i, they have a requirement that exactly p_i doctors be present.

There are k doctors, and each is asked to provide a list of days on which he or she is willing to work. Thus doctor j provides a set L_j of days on which he or she is willing to work.

The system produced by the consulting firm should take these lists and try to return to each doctor j a list L'_j with the following properties.

(A) L'_j is a subset of L_j, so that doctor j only works on days he or she finds acceptable.

(B) If we consider the whole set of lists $L'_1, ..., L'_k$, it causes exactly p_i doctors to be present on day i, for $i = 1, 2, ..., n$.

3.1: Describe a polynomial-time algorithm that implements this system.

Algorithm:

Let node set $D = \{D_1, ..., D_j, ..., D_k\}$ denote doctors and node set $A = \{A_1, ..., A_i, ..., A_n\}$ denote days. Let the flow graph $G = (V, E, c)$, where

$$\begin{aligned} V &= \{s, t\} \cup D \cup A \\ E &= \{s\} \times D \cup \{(D_j, A_i) \mid i \in L_j\} \cup A \times \{t\} \end{aligned}$$

and the edge capacities are

$$\begin{aligned} c(s, D_j) &= \infty \\ c(D_j, A_i) &= 1 \\ c(A_i, t) &= p_i. \end{aligned}$$

We then run the Preflow-Push algorithm on G. If the max-flow is not equal to $\sum_i p_i$, output "No solution exists", otherwise, output $L'_j = \{i \mid f(D_j, A_i) = 1\}$.

Proof of correctness:

Claim 1. *If there is a solution meeting the constraints, then the max-flow is $\sum_i p_i$.*

Proof. Let L'_j be the list assigned to doctor j in the solution. Since L'_j is a subset of L_j, we set $f(D_j, A_i) = 1$ if i is in L'_j and 0 otherwise. According to property (B), for $1 \leq i \leq n$, $\sum_j f(D_j, A_i) = p_i$. Thus the flow is equal to $\sum_i p_i$, and is the maximum possible flow. □

Claim 2. *If the max-flow is $\sum_i p_i$, then the algorithm outputs a solution meeting the constraints.*

Proof. Since the max-flow is $\sum_i p_i$, we have $\sum_j f(D_j, A_i) = p_i$ for $1 \leq i \leq n$. Now $L'_j = \{i \mid f(D_j, A_i) = 1\}$ is a subset of L_j for $1 \leq j \leq k$, and our solution meets all constraints. □

Time complexity:

Constructing the graph takes $O(kn)$ time. We run Preflow-Push algorithm to find the max-flow, which takes $O((k+n)^3)$ time. The overall time complexity is $O((k+n)^3)$.

3.2: Describe a polynomial-time algorithm that implements the revised system, where L'_j contains at most c days that do not appear on the list L_j.

Algorithm:

Let a new node set $D' = \{D'_1, ..., D'_j, ..., D'_k\}$ denote doctors. Reusing the same node sets from previous question, let the flow graph $G' = (V', E', c')$, where

$$\begin{aligned} V' &= D' \cup V \\ E' &= \{s\} \times D' \cup \{(D'_j, A_i) \mid i \notin L_j\} \cup E \end{aligned}$$

and capacities for E are the same as in G, capacities for new edges are

$$\begin{aligned} c'(s, D'_j) &= c \\ c'(D'_j, A_i) &= 1. \end{aligned}$$

We then run the Preflow-Push algorithm on G'. If the max-flow is not equal to $\sum_i p_i$, output "No solution exists", otherwise, output $L'_j = \{i \mid f(D_j, A_i) = 1\} \cup \{i \mid f(D'_j, A_i) = 1\}$.

Proof of correctness:

Claim 3. *If there is a solution meeting the new constraints, then the max-flow is $\sum_i p_i$.*

Proof. Let L'_j be the list assigned to doctor j in the solution. We set $f(D_j, A_i) = 1$ if i is in L'_j and also in L_j. We set $f(D'_j, A_i) = 1$ if i is in L'_j but not in L_j. Since the solution meets the new constrains, we have $\sum_i f(D'_j, A_i) = f(s, D'_j) \leq c$ for $1 \leq j \leq k$, and $\sum_j \big(f(D'_j, A_i) + f(D_j, A_i)\big) = f(A_i, t) = p_i$ for $1 \leq i \leq n$. Thus the flow is equal to $\sum_i p_i$, and is the maximum possible flow. □

Claim 4. *If the max-flow is $\sum_i p_i$, then the output solution meets the new constraints.*

Proof. Since the max-flow is $\sum_i p_i$, we have $\sum_j \big(f(D'_j, A_i) + f(D_j, A_i)\big) = f(A_i, t) = p_i$ for $1 \leq i \leq n$. Now $\{i \mid f(D_j, A_i) = 1\}$ is a subset of L_j. Per flow properties, we have $\sum_i f(D'_j, A_i) = f(s, D'_j) \leq c$ for $1 \leq j \leq k$, so the size of $\{i \mid f(D'_j, A_i) = 1\}$ is no greater than c, and our solution meets all constraints. □

Time complexity:

Same as in the previous question, except that we now have an extra set of k nodes. The overall time complexity is $O((2k+n)^3)$.

Problem 4: Cellular network

Consider the problem of selecting nodes for a cellular network. Any number of nodes can be chosen from a finite set of potential locations. We know the cost $c_i \geq 0$ of establishing site i. If sites i and j are selected as nodes, then we derive the benefit b_{ij}, which is the revenue generated by the traffic between the two nodes. Both the benefits and costs are non-negative integers. Find an efficient algorithm to determine the subset of sites as the nodes for the cellular network such that the sum of the benefits provided by the edges between the selected nodes less the selected node costs is as large as possible.

Design an efficient polynomial-time algorithm.

Algorithm:

Let node set $S = \{s_1, ..., s_i, ..., s_n\}$ denote sites and node set $U = \{u_{ij} \mid 1 \leq i, j \leq n, i \neq j\}$ denote traffic between site s_i and s_j. Let the flow graph $G = (V, E, c)$, where

$$V = \{s, t\} \cup U \cup S$$
$$E = \{s\} \times U \cup \{(u_{ij}, s_i), (u_{ij}, s_j) \mid u_{ij} \in U\} \cup S \times \{t\}$$

and the edge capacities are

$$c(s, u_{ij}) = b_{ij}$$
$$c(u_{ij}, s_i) = c(u_{ij}, s_j) = \infty$$
$$c(s_i, t) = c_i.$$

Define benefits of a subset of sites $R(\hat{S}) = \sum_{\{s_i,s_j\} \subset \hat{S}, i \neq j} b_{ij}$, cost $C(\hat{S}) = \sum_{s_i \in \hat{S}} c_i$, profit $P(\hat{S}) = R(\hat{S}) - C(\hat{S})$. We run the Preflow-Push algorithm on G. Let (A, B) be the min-cut. Output $S \cap A$ as the subset of sites with the largest profit.

Proof of correctness:

We define a cut (A, B) to be a normal cut if $c(A, B)$ is finite and $\sum_{u_{ij} \in U \cap A} c(s, u_{ij}) = R(S \cap A)$.

Claim 1. *Let (A, B) be a normal cut, then $P(S \cap A) = R(S) - c(A, B)$.*

Proof. Because (A, B) is a finite cut, only two types of edges cross from A to B: $\{(s, u_{ij}) \mid u_{ij} \in U \cap B\}$ and $\{(s_i, t) \mid s_i \in S \cap A\}$. So

$$\begin{aligned} c(A, B) &= \sum_{u_{ij} \in U \cap B} c(s, u_{ij}) + \sum_{s_i \in S \cap A} c(s_i, t) \\ &= \sum_{u_{ij} \in U} c(s, u_{ij}) - \sum_{u_{ij} \in U \cap A} c(s, u_{ij}) + \sum_{s_i \in S \cap A} c(s_i, t) \\ &= R(S) - R(S \cap A) + C(S \cap A) \\ &= R(S) - P(S \cap A) \end{aligned}$$

and $P(S \cap A) = R(S) - c(A, B)$. □

Claim 2. *For every subset of sites $\hat{S}$, there is a normal cut (A, B) s.t. $c(A, B) = R(S) - P(\hat{S})$.*

Proof. Let $A = \{s\} \cup \hat{S} \cup \{u_{ij} \mid \{s_i, s_j\} \subset \hat{S}\}$, $B = V \setminus A$, then (A, B) is a normal cut. We know from Claim 1 that $c(A, B) = R(S) - P(\hat{S})$. □

Claim 3. *The min-cut of G is a normal cut.*

Proof. The max-flow is no greater than $R(S)$, so the min-cut is finite. Suppose there is $u_{ij} \in B$ such that $\{s_i, s_j\} \subset A$ and $b_{ij} > 0$, we can include u_{ij} into A and delete it from B. In such way the cut capacity is reduced by b_{ij}, which is contradictory to the fact that we already have a min-cut. In the case where $b_{ij} = 0$, it has no effect on benefits and we can exclude u_{ij}. Hence the min-cut is a normal cut. □

Claim 4. *If (A, B) is the min-cut, then $S \cap A$ is the subset of sites with the largest profit.*

Proof. Per Claim 1 and Claim 3, we have $P(S \cap A) = R(S) - c(A, B)$. For the purpose of contradiction, assume there is a subset S^* with $P(S^*) > P(S \cap A)$. Per Claim 2 we know there exists a cut (A^*, B^*) s.t.

$$\begin{aligned} c(A^*, B^*) &= R(S) - P(S^*) \\ &< R(S) - P(S \cap A) \\ &= c(A, B), \end{aligned}$$

which is contradictory to the fact that (A, B) is the min-cut. □

Time complexity:

Constructing G takes $O(|V| + |E|) = O(n^2)$ time. We run Preflow-Push algorithm to find the min-cut, which takes $O(|V|^3) = O(n^6)$ time. The overall time complexity is $O(n^6)$.