CSE 202: Design and Analysis of Algorithms **(Due: 11/16/19)**

Homework #3

Instructor: Ramamohan Paturi *Name:* Shihan Ran, *Netid:* A53313589

Problem 1: Graph cohesiveness

Problem Description

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.

(Subproblem 1)

Give a polynomial-time algorithm that takes a rational number α and determines whether there exists a set S with cohesiveness at least α.

Solution

(High-level description)

We represent G as (V, E). Then we constructed a directed graph $G' = (N, E')$. It has vertex N_v for every vertex $v \in V$, vertex N_e for every edge $e \in E$, and two distinguished vertex source s and sink t.

Assume $c(N_i, N_j)$ is the capacity of edge (N_i, N_j). We let $c(s, N_e) = 1$ for every edge $e \in E$, $c(N_v, t) = \alpha$ for every vertex $v \in V$, $c(N_e, N_i) = c(N_e, N_j) = \infty$ if $e = (i, j) \in E$ and $i, j \in V$.

To determine whether there exists a set S with cohesiveness at least α is to determine if there is an s-t cut in graph G' of capacity at most $|E|$. It's a minimum cut problem. We use the maximum-flow algorithm (e.g. Ford-Fulkerson) to find an s-t cut of minimum capacity and compare it with $|E|$.

(Correctness)

We define an s-t cut as a partition (A, B) of N with $s \in A, t \in B$. The above-defined network G' has the following properties:

- The source node s has $|E|$ outgoing edges and their capacity is all equal to 1. So the capacity of the min cut should be less than or equal to $|E|$ (we can simply get this value by letting A=s).
- Since α is a rational number. We can transform our graph by multiplying all edge weights by the same constant factor β, where β is defined to be the smallest positive number such that $\alpha\beta$ is an integer. Then, all capacities are scaled to integers.
- If N_e is in A where $e = (i, j)$, then N_i and N_j should also be in A. This is because $c(N_i, N_j) = \infty$ and infinite capacity edge can not cross a min cut since our min cut is less than or equal to $|E|$. Similarly, if node N_v is in B, then all N_e where $e = (v, j)$ for $v, j \in V$ must also be in B.

We use A_e to denote the vertices N_e in A and A_v to denote the vertices N_v in A. Then $|A_e|$ should be exactly the number of edges in the original graph G that have both endpoints in A_v, and according to the problem description, there are $e(A_v)$ such edges. The edges that cross our cut are

- All edges (N_v, t) for $N_v \in A_v$, each with the capacity α. There are $|A_v|$ edges.
- All edges s, N_e for which $e = (i, j)$ and $i, j \notin A_v$, each has a capacity 1. There are $|E| - e(A_v)$ edges.

Hence, the total capacity of our cut is $c(A, B) = \alpha|A_v| + |E| - e(A_v)$. We can arrange this to get

$$|E| - c(A, B) = e(A_v) - |A_v|\alpha$$

According to the problem description, we can see that $e(A_v)/|A_v| \geq \alpha$ iff $|E| - c(A, B) \geq 0$. That is $c(A, B) \leq |E|$.

So we've prove that we have a min cut with $c(A, B) \leq |E|$ then we can find a group of vertices in our original graph with cohesiveness greater than α.

(Subproblem 2)

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

(High-level description)

There are $|V|$ choices for the size of A_v and $C_2^{|A_v|}$ choices for $e(A_v)$. Thus, it's a total of $O(|V|^3)$ possible values for α. We can find the set of maximum cohesiveness by binary search on α by checking if there exists a subset with cohesiveness greater than a given value.

(Time complexity)

By using the Ford-Fulkerson maximum-flow algorithm to determine the minimum cut for a given α, the final time complexity should be $O(|V|^3(|V| + |E|)C)$.

Problem 2: Remote Sensors

Problem Description

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 $1 \le j \le m$, 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, $\sqrt{\left((x_j - x_k)^2 + (y_j - y_k)^2\right)}$). All base stations have the same integer bandwidth capacity C.

For $1 \le i \le n$, 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 $1 \le i \le n$, $\sum_j b_{i,j} \ge r_i$.
- The sum of all bandwidth assigned on base station B_j must be at most C: for each $1 \le j \le m$, $\sum_i b_{i,j} \le C$.

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.

Solution

(High-level description)

(Correctness)

(Time complexity)

Problem 3: Scheduling in a medical consulting firm

Problem Description

You've periodically helped the medical consulting firm Doctors Without Weekends on various hospital scheduling issues, and they've just come to you with a new problem. 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 $L'_1, \ldots, L'_k$, it causes exactly p_i doctors to be present on day i, for $i = 1, 2, \ldots, n$.

(Subproblem 1)

Describe a polynomial-time algorithm that implements this system. Specifically, give a polynomial- time algorithm that takes the numbers $p_1, p_2, \ldots, p_n$, and the lists $L_1, \ldots, L_k$, and does one of the following two things.

- Return lists $L'_1, \ldots, L'_k$ satisfying properties (A) and (B); or
- Report (correctly) that there is no set of lists $L'_1, \ldots, L'_k$ that satisfies both properties (A) and (B).

(Solution 1)

(High-level description)

(Pseudo Code)

(Correctness)

(Time complexity)

Problem 4: Cellular network

Problem Description

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 benefit $b_i \geq 0$ of establishing site i. However, if sites i and j are selected as nodes, then the benefit is offset by c_{ij}, which is the cost of interference 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 node benefits less the interference costs is as large as possible.

Design an efficient polynomial-time algorithm.

Provide a high-level description of your algorithm, prove its correctness, and analyze its time complexity.

Solution

(High-level description)

(Pseudo Code)

(Correctness)

(Time complexity)