CSE 202 - Homework 3

3.1 Graph Cohesiveness

Part 1:
For the given graph G, we construct a graph $G^{'} = (V^{'}, E^{'})$ such that for every vertex in G there exists a node v_i in $G^{'}$. Let the set of all such nodes in $G^{'}$ (corresponding to vertices in G) be represented as V. Also for every edge connecting vertices v_i and v_j in the graph G, we introduce a node e_{ij} in $G^{'}$. Let the set of all such nodes in $G^{'}$ (corresponding to edges in G) be represented as E. Additionally, we introduce two nodes s, t in $G^{'}$ which represent the source and sink respectively. The vertices in $G^{'}$ therefore are:

$$V^{'} = V \cup E \cup \{s, t\}$$

Let the source node s be connected to all nodes e_{ij} in E with the edges directed from s to e_{ij}. The capacities of such edges from s to e_{ij} are set to 1. Each node e_{ij} is in turn connected to the two nodes v_i and v_j in V (the two nodes whose connecting edge in the original graph e_{ij} represents) with the edges directed from e_{ij} to v_i and v_j. The capacity of all such edges is set to infinity. All nodes v_i in V are connected to the sink t with the edges directed from v_i to t. The capacities of such edges from v_i to t are set to α. The edge capacities in the graph are then given as:

$$c(s, e_{ij}) = 1 \text{ where } e_{ij} \in E$$
$$c(e_{ij}, v_i) = c(e_{ij}, v_j) = \infty \text{ where } e_{ij} \in E \text{ and } v_i, v_j \in V$$
$$c(v_i, t) = \alpha \text{ where } v_i \in V$$

Consider an arbitrary cut in the graph $G^{'}$. Suppose that this cut divides the graph into source-side and sink-side subsections L and R respectively. Suppose E_L and E_R are the subsets of E in L and R respectively and V_L and V_R are the subsets of V in L and R respectively.

Claim 1: The minimum cut capacity in $G^{'}$ is less than or equal to number of edges in G i.e. $|E|$.
Explanation: Considering the cut where L contains only the source s, the edges across the cut are those connecting s to all nodes in E with a capacity of 1. The capacity of this cut is then number of such nodes in E i.e. $|E|$. Therefore the min cut capacity of $G^{'}$ has to be less than or equal to $|E|$.

Claim 2: For a minimum cut in $G^{'}$, if the node e_{ij} is in E_L, the vertices v_i and v_j must be in V_L. Also if v_i and v_j are in V_L, e_{ij} must be in E_L for a minimum capacity cut
Explanation: Consider a cut where e_{ij} is in E_L and either or both of v_i and v_j are in V_R. Then the capacity of such a cut must include the capacity of edges connecting E_L to v_i and/or v_j which is infinity. Hence, such a cut is not finite and can not be considered as a min cut since the min cut is finite from claim 1. Further if we consider a cut such that v_i and v_j are in V_L, e_{ij} is in E_R, it is possible to construct a cut with lesser capacity by placing e_{ij} in E_L. Hence, a cut with v_i, v_j in V_L and e_{ij} in E_R can not be a min cut.

The only edges across the cut from source-side to sink-side for a min cut then are : the edges connecting the source s to nodes in E_R and the edges connecting nodes in V_L to the sink t. The capacity of such cut

is the sum of these edges across the cut which is given by:

$$C(L,R) = \sum_{e_{ij} \in E_R} 1 + \sum_{v_k \in V_L} \alpha$$

Adding and subtracting summation of 1 over the nodes in E_L in the above equation, we have:

$$C(L,R) = \sum_{e_{ij} \in E_R} 1 + \sum_{e_{ij} \in E_L} 1 + \sum_{v_k \in V_L} \alpha - \sum_{e_{ij} \in E_L} 1$$

Since $E_L \cup E_R = E$, the first two terms are the summation of 1 over nodes in E. This summation is equal to the number of edges in the original graph and hence is a constant.

$$C(L,R) = \sum_{e_{ij} \in E} 1 + \sum_{v_k \in V_L} \alpha - \sum_{e_{ij} \in E_L} 1$$

$$C(L,R) = |E| + \sum_{v_k \in V_L} \alpha - \sum_{e_{ij} \in E_L} 1$$

$$C(L,R) = |E| + \alpha |V_L| - |E_L|$$

$$(|E_L| - \alpha |V_L|) = |E| - C(L,R)$$

Claim 3: If the maximum flow in the graph $G^{'}$ computed using preflow-push algorithm does not saturate the edges originating from the source s, only then there exists a set S with with cohesiveness greater than α. The set of nodes V_L of the minimum cut (i.e. the nodes on the source side of the cut in $G^{'}$ that represent vertices in G) then represent such set S with cohesiveness greater than α.

Explanation:
Let $minC(L,R)$ denote capacity of the min cut in the graph. Then,

$$|E_L| - \alpha |V_L| = |E| - minC(L,R)$$

Since capacity of the min cut is at most equal to $|E|$ (from Claim 1) but the maximum flow does not saturate the outgoing edges from s, there must exist a minimum cut with capacity strictly less than $|E|$. Then,

$$|E_L| - \alpha |V_L| > 0$$

Consider the set $S = V_L$. $|E_L|$ then represents the number of edges with both ends in S i.e. $e(S)$. (from Claim 2). This set has a cohesiveness strictly greater than α:

$$e(S) - \alpha |S| > 0$$

$$\frac{e(S)}{|S|} > \alpha \text{ where } S = V_L$$

Part 1 Complexity Analysis:
Suppose $|V|$ = n and $|E|$ = m. The construction of the network flow graph takes time of the order $O(n+m)$. Finding the maximum flow/ min cut for the graph $G^{'}$ with $n+m$ nodes using preflow push algorithm takes $O((n+m)^3)$ time. Therefore overall time complexity of the algorithm is $O((n+m)^3)$.

Part 2 Description:
For a graph with $|V|$ nodes, the values of $e(S)$ i.e. number of edges in any subset of the graph can vary between 0 and $|V|^2$ and the value of $|S|$ lies between 0 and $|V|$. The cohesiveness $e(s)/|S|$ can therefore assume $(|V|^2+1)|V|$ values (0 is not a valid value for denominator). By performing a binary search on these potential values of cohesiveness, assigning such values to α and finding the max flow/ min cut as outlined in part a using the preflow push algorithm, we can find the subset of nodes with maximum cohesiveness in the graph.

Part 2 Complexity Analysis:
Suppose $|V|$ = n and $|E|$ = m. The range of binary search for a graph with $|V|$ vertices is of the order n^3 ($(|V|^2+1)|V|$ possible values). For each step of the binary search the preflow push algorithm takes $O((n+m)^3)$ time. The overall time complexity is therefore $O((n+m)^3 log(n^3))$ which simplifies to $O((n+m)^3 log(n))$.

3.2 Remote Sensors

Description:
We construct a network flow graph $G=(V,E)$ which has two sets of nodes S and B corresponding to the sensors and base stations respectively. The base stations in B are denoted as B_j where $1 \leq j \leq m$ and the sensors in S are denoted by s_i where $1 \leq i \leq n$. We also introduce two nodes $source$ and $sink$ for the source and sink respectively in the graph. The vertices in G therefore are:

$$V = B \cup S \cup \{source, sink\}$$

Let the $source$ be connected to all nodes B_j in B with the edges directed from $source$ to B_j. The capacities of such edges from $source$ to B_j are set to the bandwidth C. Each node B_j is then connected to sensor nodes s_i if B_j and s_i are present within a distance of 2 km of each other. The edges are directed from B_j to s_i. The capacities of such edges are set to infinity. For each sensor s_i in S, we connect s_i to $sink$ with edges directed from s_i to $sink$. The capacities of such edges are set to their respective minimum bandwidth requirement i.e. r_i. The edges in the graph are then given as:

$$E = (source \times B) \cup (B \times S) \cup (S \times sink)$$

with edge capacities

$$c(source, B_j) = C \text{ where } B_j \in B$$

$$c(B_j, s_i) = \infty \text{ where } B_j \in B \text{ and } s_i \in S \text{ and } distance(B_j, s_i) \leq 2km$$

$$c(s_i, sink) = r_i \text{ where } s_i \in S$$

There exists a valid allocation of bandwidth if the maximum flow in this network flow graph G is equal to the sum of the required bandwidths for all the sensors. The valid allocation of bandwidth is then the flow in the edges connecting the nodes corresponding to the base stations and those corresponding to the sensors.

Let R denote such sum of the required bandwidths for all the sensors and b_{ij} denote the flow from base station B_j to sensor s_i which also indicates the bandwidth allocation to s_i from B_j.

$$R = \sum_{i=1}^{n} r_i$$

Claim 1: If the maximum flow in this network flow graph G is R, then the flow in the edges connecting the nodes corresponding to the base stations and those corresponding to the sensors is a valid allocation.
Explanation: If the maximum flow in G is R, it implies that the cut which has only $sink$ on the sink side is saturated i.e. all the edges connecting sensors s_i to $sink$ have a flow equal to their capacities r_i. By conservation of flow, the flow outgoing from each sensor is equal to the incoming flow. The net bandwidth supplied to sensor s_i is then exactly equal to r_i which satisfies the minimum required constraint for each sensor. Also since the source supplies each base station with a flow not more than C (capacity of the edge), the flow outgoing from each of the base stations is not more than C by conservation. Additionally, an edge and thus a flow between a base station and sensor exists only if the distance constraint is satisfied from the structure of the graph. Hence the flows b_{ij}'s when a maximum flow of R is achieved is a valid allocation of bandwidth.

Claim 2: If there exists a valid allocation of bandwidths to the sensors, then the maximum flow in the network graph is R.
Explanation: Suppose that the valid assignment for each sensor s_i from are denoted as b_{ij}'s where $1 \leq j \leq m$. Since it is a valid assignment the requirement for the s_i met and the sum of such b_{ij}'s is at least r_i. Then,

$$\sum_{j} b_{ij} \geq r_i$$

For cases where the equality holds true, b_{ij} can be assigned as the flow from bases stations B_j's to sensor s_i. For other cases, we iteratively pick the maximum of b_{ij}'s corresponding to that sensor and decrease such b_{ij} by 1. We do this until the sum equals r_i. When such a configuration is achieved for all sensors, these b_{ij}'s can be assigned as the flows between corresponding base stations and sensors. Since, it is ensured that the incoming sum of flow for each sensor is r_i after the adjustments, the outgoing flow from each sensor is also r_i. The maximum flow across any cut is then the sum of all r_i's i.e. R since the edges connecting sensors to sink are saturated.

Complexity Analysis:
It is given that no two base stations are less than 1 km apart and the sensors are connected to only such base stations that are less than 2 km from it. Therefore there exist finite number of base stations that each sensor can connect to. The edges in the network flow graph are therefore proportional to the number of sensors n. The number of nodes in the graph is equal to sum of number of base stations (m) and sensors (n). The construction of the graph therefore takes time of the order $O(n)$. Since $m < n$, the number of vertices is also proportional to n. Therefore we can use the preflow push algorithm which computes the maximum flow in the graph in $O(n^3)$ time. Overall time complexity is therefore $O(n^3)$.

3.3 Scheduling in a medical consulting firm

Part 1 High Level description:
The strategy to check for each day the doctors that are willing to work on that day. If on a day i, the number of doctors willing to work are at least p_i, then return the list of first p_i doctors for that day. If for any day, the number of available doctors is less than that required, output that no valid assignment exists for the scenario.

Part 1 Algorithm:

- Initialize a matrix M with n row and k columns where n is the total number of days and k is the total number of doctors. Set all the values of the matrix to 0.
- For each doctor j, iterate over the list of days they are willing to work on i.e L_j. For each work day w_i in such a list, set the value of the element at ${w_i}^{th}$ row and ${L_j}^{th}$ column to 1 i.e. $M[w_i][L_j] = 1$.
- Iterate over the n rows in M to check if for each row i, there exist at least p_i 1's in the row. If for any row, this condition is not met, output that no valid assignment exists for the scenario. Else for the day i, set the elements that have been encountered after p_i 1's to 0.
- For each column in M corresponding to a doctor j, construct a list ${L_j}^{'}$ which contains the row numbers in that column where the element is non-zero. Return the list of all such lists as the output.

Part 1 Time Complexity:
For constructing the matrix, we iterate over the list of days provided by each doctor which take time of the order $O(nk)$. To check for the availability of required number of doctors and to pick the p_i doctors for each day again takes a time of the order $O(nk)$. Construction of the output list for each doctor requires to iterate over the matrix elements which takes $O(nk)$. Therefore overall time complexity of the algorithm is $O(nk)$.

Part 2 Description:
We construct a network flow graph $G = (V, E)$ which has two sets of nodes D and W corresponding to the doctors and work days respectively. Suppose the doctors in D are denoted as d_j where $1 \leq j \leq k$ and the work days in W are denoted by w_i where $1 \leq i \leq n$. We also introduce two nodes s and t for the source and sink respectively in the graph. The vertices in G therefore are:

$$V = D \cup W \cup \{s, t\}$$

Let A_i denote the number of doctors willing to work on work day w_i which can be computed as sum of row w_i in M from step 2 of part a.

Let the source node s be connected to all nodes d_j in D with the edges directed from s to d_j. The capacities of such edges from s to d_j are set to c (maximum relaxation in number of days). All nodes d_j are then connected to such work day nodes w_i that are not present in the list L_j with edges directed from d_j to w_i. The capacities of such edges are set to 1. The source s is also connected to each workday w_i with edge directed from source to w_i. The capacities of such edges are set to A_i. For each work day w_i in W, we connect w_i to sink t with edge directed from w_i to t. The capacities of such edges are set to p_i. The edge capacities of the graph are then given as:

$$c(s, d_j) = c \text{ where } d_j \in D$$

$$c(s, w_i) = A_i \text{ where } w_i \in W$$

$$c(d_j, w_i) = 1 \text{ where } d_j \in D \text{ and } w_i \in W \text{ and } w_i \notin L_j$$

$$c(w_i, t) = p_i \text{ where } w_i \in W$$

There exists a valid assignment of doctors if the maximum flow in this network flow graph G is equal to the sum of the required doctors for each day. The valid allocation for each day is then the doctors d_j's connected to workday node w_i with a non zero flow and the first f_i doctors available on w_i (from ${w_i}^{th}$ row in M at step 2 of part a) where f_i denotes flow in the edge from source to node w_i.

Let P denote such sum of the required doctors for all work days. Then,

$$P = \sum_{i=1}^{n} p_i$$

Claim 1: If the maximum flow in this network flow graph G is P, then the edges from d_j's to w_i's with a non zero flow and the flow in edges from source s to w_i's provide a valid assignment.
Explanation: If the maximum flow in G is P, it implies that the cut which has only t on the sink side is saturated i.e. all the edges connecting sensors w_i to t have a flow equal to their capacities p_i. By conservation of flow, the flow outgoing from each workday is equal to the incoming flow. The number of doctors assigned to w_i is then exactly equal to p_i which satisfies the minimum required constraint for each day. Also since the source supplies each node d_j with a flow not more than c (capacity of the edge), the flow outgoing from each of the nodes d_j is not more than c by conservation. Since nodes d_j's are connected to only such w_i's that d_j is not willing to work, the flow takes care of the constraint that the list provided by the doctor can be relaxed by a maximum of c days. Additionally, the flow from source s to workday nodes w_i is bounded by the number of doctors willing to work on that day. Such flows ensure assignments of only those doctors available to work on that day. Therefore, when a maximum flow of P is achieved, edges from d_j's to w_i's with a non zero flow and the flow in edges from source s to w_i's provide a valid assignment.

Claim 2: If there exists a valid assignment of doctors to the work days, then the maximum flow in the network graph is P.
Explanation: If there exists a valid assignment, the number of doctors available to work on a day w_i after a relaxation of c days is at least p_i. For cases where $A_i \geq p_i$, a flow of p_i can directly be assigned on the edges connecting source s to w_i and w_i to sink t.For cases where $A_i < p_i$, a flow of A_i is assigned on the edges connecting source s to w_i. Since a valid assignment exists, there are at least $(p_i - A_i)$ doctors that can be assigned on that day after the relaxation of c days. We set flow on exactly $(p_i - A_i)$ such edges connecting d_j's to w_i's (these edges have a capacity of 1) to 1. The net flow from each w_i to sink is then p_i. The maximum flow across any cut in G is then the sum of all p_i's i.e. P since the edges connecting w_i's to sink are saturated.

Complexity Analysis:
The time taken to construct the network flow graph is of the order $O(nk)$. Number of nodes in the network flow graph excluding the source and sink are $n + k$. Using preflow push algorithm to find the maximum

flow takes time of the order $O((n+k)^3)$. After finding the maximum flow, the construction of the lists from matrix M and the graph takes a time complexity of $O(nk)$. Therefore overall complexity is $O((n+k)^3)$.

3.4 Cellular Network

High level description:

We construct a network flow graph $G = (V, E)$ which has a sets of nodes S corresponding to the available sites. Suppose the sites in S are denoted as s_i where $1 \leq i \leq n$. Also for every pair of sites s_i and s_j in the graph G, we introduce a node p_{ij} corresponding to the benefit derived from establishment of sites s_i and s_j. Let the set of all such nodes in G be represented as P. We also introduce two nodes $source$ and $sink$ for the source and sink respectively in the graph.The vertices in $G^{'}$ therefore are:

$$V = S \cup P \cup \{source, sink\}$$

Let $source$ be connected to all nodes p_{ij} in P with the edges directed from $source$ to p_{ij}. The capacities of such edges from $source$ to p_{ij} are set to b_{ij} (benefit derived from establishment of sites s_i and s_j). Each node p_{ij} is in turn connected to the two nodes s_i and s_j in S (the two nodes whose benefit b_{ij} represents) with the edges directed from p_{ij} to s_i and s_j. The capacity of all such edges is set to infinity. All nodes s_i in S are connected to $sink$ with the edges directed from s_i to $sink$. The capacities of such edges from s_i to $sink$ are set to c_i (cost of establishing s_i). The edges capacities in the graph are then given as:

$$c(source, p_{ij}) = b_{ij} \text{ where } p_{ij} \in P$$
$$c(p_{ij}, s_i) = c(p_{ij}, s_j) = \infty \text{ where } p_{ij} \in P \text{ and } s_i, s_j \in S$$
$$c(s_i, sink) = c_i \text{ where } s_i \in S$$

Consider an arbitrary cut in the graph G. Suppose that this cut divides the graph into source-side and sink-side subsections L and R respectively. Suppose P_L and P_R are the subsets of P in L and R respectively and S_L and S_R are the subsets of S in L and R respectively.

Claim 1: The minimum cut capacity in the graph G is finite.
Explanation: Considering the cut where L contains only the $source$, the edges across the cut are those connecting $source$ to all nodes in P with a capacities corresponding to benefits of each pair of sites. The capacity of this cut is then the sum of all benefits for all pairs of sites. Therefore the min cut capacity of G is finite and is less than or equal to sum of all benefits for all pairs of sites in $|S|$.

Claim 2: For a minimum cut in G, if the node p_{ij} is in P_L, the nodes s_i and s_j must be in S_L. Also if s_i and s_j are in S_L, p_{ij} must be in P_L for a minimum capacity cut.
Explanation: Consider a cut where p_{ij} is in P_L and either or both of s_i and s_j are in S_R. Then the capacity of such a cut must include the capacity of edges connecting p_{ij} to s_i and/or s_j which is infinity. Hence, such a cut is not finite and can not be considered as a min cut since the min cut is finite from claim 1. Further if we consider a cut such that s_i and s_j are in S_L, p_{ij} is in P_R, it is possible to construct a cut with lesser capacity by placing p_{ij} in P_L. Hence, a cut with s_i, s_j in S_L and p_{ij} in P_R can not be a min cut.

The only edges across the cut from source-side to sink-side for a min cut then are : the edges connecting the $source$ to nodes in P_R and the edges connecting nodes in S_L to the $sink$. The capacity of such cut is the sum of these edges across the cut which is given by:

$$C(L,R) = \sum_{p_{ij} \in P_R} b_{ij} + \sum_{s_k \in S_L} c_k$$

Adding and subtracting summation of benefits over the nodes in P_L in the above equation, we have:

$$C(L,R) = \sum_{p_{ij} \in P_R} b_{ij} + \sum_{p_{ij} \in P_L} b_{ij} + \sum_{s_k \in S_L} c_k - \sum_{p_{ij} \in P_L} b_{ij}$$

Since $P_L \cup P_R = P$, the first two terms are the summation of benefits over all nodes in P which is a constant. Let this constant be denoted as B.

$$C(L,R) = \sum_{p_{ij} \in P} b_{ij} + \sum_{s_k \in S_L} c_k - \sum_{p_{ij} \in P_L} b_{ij}$$

$$C(L,R) = B + \sum_{s_k \in S_L} c_k - \sum_{p_{ij} \in P_L} b_{ij}$$

$$\sum_{p_{ij} \in P_L} b_{ij} - \sum_{s_k \in S_L} c_k = B - C(L,R)$$

Claim 3: The minimum cut in the graph G produces a set of site nodes M that maximizes the net profit (i.e benefits - costs). The set of nodes S_L of the minimum cut (i.e. the site nodes on the source side of the cut in G) then represent such set M with maximum profit.

Explanation:
Let $minC(L,R)$ denote capacity of the min cut in the graph. Then,

$$\sum_{p_{ij} \in P_L} b_{ij} - \sum_{s_k \in S_L} c_k = B - minC(L,R)$$

Considering the set S_L, b_{ij}'s in the above equation represent the benefits corresponding to the sites s_i and s_j both of which are present in S_L(follows from Claim 2). The left hand side of the equation then represents the net profit for the sites in S_L. Since min cut capacity maximizes the RHS of the equation, the set S_L is equivalent to set M with maximum net profit. Therefore,

$$\text{Maximum Net Profit} = B - minC(L,R)$$
$$\text{Set of sites with maximum profit} = \text{Sites on source side of min-cut}$$

Complexity Analysis:
The construction of the network flow graph takes time of the order $O(n^2)$ since there are n site nodes and n^2 benefit nodes. Using the preflow push algorithm to find the maximum flow thus takes time of the order $O((n^2)^3)$ i.e. overall complexity is $O(n^6)$.