Algorithms: CSE 202 — Homework 3 Solutions

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. Give a polynomial-time algorithm that takes a rational number α and determines whether there exists a set S with cohesiveness greater than α.
2. Give a polynomial-time algorithm to find a set S of nodes with maximum cohesiveness.

Solution: Graph cohesiveness (KT 7.46)

Determining whether graph cohesiveness is strictly larger than a rational number α: Let $G = (V, E)$ be the friendship graph. For $S \subseteq V$, let $e(S)$ be the number of edges in E with both ends in S. Define the *cohesiveness* of S as $\frac{e(S)}{|S|}$.

Let α be a rational number. We design an efficient algorithm to determine whether there exists a vertex set S with cohesiveness strictly *larger* [1] than α.

Flow network: We construct a flow network $\mathbb{G} = (\mathbb{V}, \mathbb{E}, c)$ to solve the problem. $\mathbb{G}$ contains a source node s and a sink node t. For each vertex $x \in V$, $\mathbb{G}$ contains a node v_x. For each edge $(x, y) \in E$, $\mathbb{G}$ also contains a node $u_{x,y}$. $\mathbb{G}$ contains the following edges:

- source node s connects to each vertex of the form $u_{x,y}$ with an edge of capacity 1 [$s \longrightarrow u_{x,y}$ with capacity 1],
- each node of the form $u_{x,y}$ connects to nodes v_x and v_y with infinite capacity [$u_{x,y} \longrightarrow v_x$ and $u_{x,y} \longrightarrow v_y$ with capacity ∞],
- each node v_x connects to the sink node t with capacity α [$v_x \longrightarrow t$ with capacity α]

The flow network $\mathbb{G}$ is shown in Figure 1 schematically. It has the following properties

1. Since the source node s has exactly $|E|$ outgoing edges, each with capacity 1, the capacity of the min-cut($\mathbb{G}$) is less or equal to $|E|$.
2. Since α is a rational number, we can scale the capacities to integers.
3. The complexity of running the Preflow-Push maximum-flow algorithm on $\mathbb{G}$ is $O((|V| + |E|)^3)$.

[1] We will show later that cohesiveness of all subsets of G is a finite set of discrete rational numbers D. For an arbitrary rational number α, let β be the largest rational number in D such that $\beta < \alpha$. Then determining whether graph cohesiveness is *at least* α is equivalent to determining whether graph cohesiveness is strictly *larger* than β.

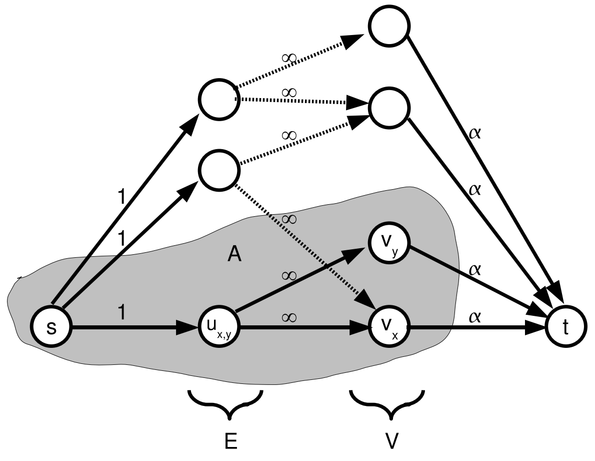

Figure 1: Flow network $\mathcal{G}$ to determine if graph cohesiveness is strictly larger than α

Theorem 0.1. *There exists a vertex set S in G with cohesiveness strictly larger than α if and only if the maximum flow in $\mathbb{G}$* cannot *saturate all the edges leaving the source s.*

Proof. Let (A, B) be a min-cut of the flow network $\mathbb{G}$ obtained by running the maximum flow algorithm. Let S^* be the set of vertices in A of the form v_x. Observe that $u_{x,y} \in A$ if and only if both $v_x, v_y \in S^*$. Indeed, if $u_{x,y} \in A$ and v_x or v_y were in B, then (A, B) will have infinite capacity, which is not possible since the cut (A, B) has finite capacity. If $v_x \in A$ and $v_y \in A$ but $u_{x,y} \notin A$, then adding $u_{x,y}$ to A decreases the cut capacity, contradicting the fact that (A, B) is a minimum cut.

The capacity of the minimum cut (A, B) is

$$\begin{aligned}\text{capacity}(A, B) &= \sum_{u_{x,y} \notin A} 1 + \sum_{v_x \in A} \alpha \\ &= (\sum_{(x,y) \in E} 1 - \sum_{u_{x,y} \in A} 1) + \sum_{x \in S^*} \alpha \\ &= |E| - e(S^*) + \alpha |S^*|\end{aligned}$$

If maximum flow cannot saturate all the edges leaving the source s, then by the max-flow min-cut theorem, capacity$(A, B) < |E|$, so $e(S^*) > \alpha|S^*|$, i.e., the set S^* has cohesiveness strictly larger than α.

Suppose there exists a set S with cohesiveness strictly larger than α and the maximum flow is $|E|$. This implies capacity$(A, B) = |E|$. We get a contradiction by constructing a cut of capacity smaller than $|E|$. Let $A' = \{s\} \cup \{v_x | x \in S\} \cup \{u_{x,y} | (x, y) \in E, x, y \in S\}$. Let B' be the remaining nodes of the flow network $\mathbb{G}$. We then have

$$\text{capacity}(A', B') = |E| - e(S) + \alpha|S| < |E|,$$

which leads to the desired contradiction since (A, B) is a minimum cut.

□

Finding a set S with maximum cohesiveness

There are $|V|$ choices for the size of $S \subseteq V$. There are $\binom{|S|}{2} = \frac{|S|(|S|-1)}{2}$ choices for $e(S)$. We thus have a total of $O(|V|^3)$ possible values for α. We apply binary search to determine the set with largest cohesiveness by using the algorithm that checks if there exists a subset with cohesiveness greater than a given value.

By applying the Preflow-Push algorithm to determine the minimum cut for a given α, we can compute the set with maximum cohesiveness in time $O((|V| + |E|)^3 \lg |V|)$.

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 compplexity.

Solution: Remote sensors

By Joseph Li

Input: n remote sensors s_i and $m < n$ base stations B_j

Output: Bandwidth allocation, that is, a list of $b_{i,j}$ indicating how much station B_j provides to sensor s_i, subject to the following constraints. If such a list does not exist, detect this fact.

Constraints

1. Each sensor must be within 2km of any base stations that supply it.
2. No station can supply more than C bandwidth.
3. Each sensor s_i must receive at least r_i bandwidth.
4. No two base stations are less than 1km apart.

Description

Nodes

1. s and t
2. $B = \{B_j | B_j \text{ is a base station}\}$
3. $S = \{s_i | s_i \text{ is a sensor}\}$

   The set of nodes for the flow graph is $\{s, t\} \cup B \cup S$.

Edges

1. $s \to B_j$ with capacity C, for each station
2. $s_i \to t$ with capacity r_i, for each sensor
3. $B_j \to s_i$ with capacity ∞, for each (i, j) pair for which station B_j is within 2km from sensor s_i

Algorithm: The edge capacities are stated to be integers, so Ford-Fulkerson will work. However, their values are not bounded, and Ford-Fulkerson has a worst-case run time that is proportional to the value of the max flow.

By comparison, the number of nodes is linear with respect to $n+m$, which reduces to $O(n)$ since we are told that $m < n$. We can naively conclude that the number of edges is at most the square of that, at $O(n^2)$ (there may in fact be a tighter bound). Therefore, we will use preflow-push.

Solution: Let $R = \sum_{i=1}^{n} r_i$. There exists a solution for the bandwidth allocation problem if and only if the max flow is R. Equivalently, the problem has a solution if and only if $(\{s\} \cup B \cup S, \{t\})$ is a minimum capacity cut. Note that the capacity of the cut is R.

If the max flow is R, we extract our configuration as follows. For each edge $s_i \to B_j$ with non-zero flow, set $b_{i,j}$ as the value of the flow.

Complexity

To build our flow graph, we require $O(n^2)$ time, as that is the previously estimated upper bound on the number of edges.

Preflow-push has a complexity of $O(|V|^2|E|)$. Again, we previously estimated the number of nodes and edges to be $O(n)$ and $O(n^2)$, respectively - this would give us a complexity of $O(n^4)$.

To determine a tight upper bound on the number of edges, we refer to constraints 1 and 4. We know that we can only fit finitely many unit circles within a larger circle with a radius of 2. Therefore, the number of edges per sensor is bounded by a constant, and therefore the total number of edges is on the same order as the number of nodes - $O(n)$.

With this, our complexity becomes $O(n^3)$.

Alternatively, by using the improved preflow-push algorithm which takes $O(|V|^3)$ time we can directly dervie an upper bound of $O(n^3)$ on the complexity of the problem.

Correctness

Claim 1. *If max flow is R, then the $b_{i,j}$ extracted from the flow satisfy the constraints of the bandwidth allocation.*

We will prove that our $b_{i,j}$ satisfy all of the original problem's constraints.

1. For each $b_{i,j}$ in our bandwidth allocation, sensor s_i is within 2 km of the base station B_j since there is an edge $B_j \to s_i$ in the flow graph and the graph is constructed with the property that an edge exists between the base station and a sensor if and only if they are less than 2 km apart.

2. For any base station B_j, $\sum_i b_{ij} \le C$ since the node B_j can receive a flow of at most C from s and conservation property ensures that the sum of the outflows is equal to the sum of the inflows.

3. Finally, we will show that the total bandwidth $\sum_j b_{i,j}$ allocated to sensor s_i is at least r_i. Since the max flow is $R = \sum_{i=1}^{n} r_i$, every edge $s_i \to t$ from a sensor to the node t carries a flow equal to its capacity which in turn implies that $\sum_j b_{i,j} = r_i$ satisfying the bandwidth requirement of the sensor s_i.

We now prove

Claim 2. *If there is a bandwidth allocation satisfying the constraints, then the maximum flow of our network is R.*

Let $b_{i,j}$ be the allocation to sensor s_i from base station B_j. We will show how to construct a flow in our network whose value is R. Since our network has a cut of capacity R, we will then conclude that the max flow is R.

Without loss of generality, we assume that the total bandwidth allocated to each sensor is exactly r_i, that is, $\sum_j b_{i,j} = r_i$. If $\sum_j b_{i,j} > r_i$, we decrease the largest $b_{i,j}$ in the sum by 1. We continue this as long as $\sum_j b_{i,j} > r_i$. At the end of this process, we get $\sum_j b_{i,j} = r_i$ for each sensor s_i. It is clear that this transformation results in a bandwidth allocation that satisfies all the constraints and achieves the additional property $\sum_j b_{i,j} = r_i$ for each sensor s_i.

Since the allocation satisfies the constraints, there is an edge from B_j to s_i in the flow graph for each $b_{i,j}$ in the bandwidth allocation. Let the flow on the edge $B_j \to s_i$ be $b_{i,j}$. Let the flow on the edge $B_j \to s_i$ be $\sum_i b_{i,j}$ and the flow on the edge $s_i \to t$ be $\sum_j b_{i,j} = r_i$. The flow on all other edges is zero.

It is clear from the construction that the flow is valid, that is, the flow satisfies the capacity and conservation constraints and the value of the flow is R.

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

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 $i = 1, 2, \ldots, n$.

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 $L'_1, L'_2, \ldots, L'_k$ satisfying properties (A) and (B); or
   - Report (correctly) that there is no set of lists $L'_1, L'_2, \ldots, L'_k$ that satisfies both properties (A) and (B).

2. The hospital finds that the doctors tend to submit lists that are much too restrictive, and so it often happens that the system reports (correctly, but unfortunately) that no acceptable set of lists $L'_1, L'_2, \ldots, L'_k$ exists.

   Thus the hospital relaxes the requirements as follows. They add a new parameter $c > 0$, and the system now should try to return to each doctor j a list L'_j with the following properties.

   (A^*) L'_j contains at most c days that do not appear on the list L_j.

   (B) (Same as before) 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 $i = 1, 2, \ldots, n$.

   Describe a polynomial-time algorithm that implements this revised system. It should take the numbers $p_1, p_2, \ldots, p_n$, the lists $L_1, \ldots, L_k$, and the parameter $c > 0$, and do one of the following two things.

   - Return lists $L'_1, L'_2, \ldots, L'_k$ satisfying properties (A^*) and (B); or
   - Report (correctly) that there is no set of lists $L'_1, L'_2, \ldots, L'_k$ that satisfies both properties (A^*) and (B).

Solution: Medical consulting

Input: k doctors each providing a list L_j of days they are willing to work and n days each such that each day i has a requirement of p_i doctors.

Output: Return lists L'_1, L'_2, ..., L'_k satisfying the constraints below. If such a lists do not exist, detect this fact.

Constraints

1. L'_j is a subset of L_j , so that doctor j only works on days he or she finds acceptable.
2. If we consider the whole set of lists L'_1, L'_2, ..., L'_k, it causes exactly p_i doctors to be present on day i, for i = 1, 2, . . . , n.

Note: There is a straightforward greedy solution to this problem. You will get full credit if you provided a correct greedy solution.

Description

Nodes

1. s and t
2. $D = \{D_j | D_j \text{ is a doctor}\}$
3. $H = \{H_i | H_i \text{ is a day}\}$

   The set of nodes for the flow graph is $\{s, t\} \cup D \cup H$.

Edges

1. $s \rightarrow D_j$ with capacity $|L_j|$, for each doctor
2. $H_i \rightarrow t$ with capacity p_i, for each day
3. $D_j \rightarrow H_i$ with capacity 1, for each (i, j) pair for which doctor D_j is willing to work on day H_i

Solution: Let $P = \sum_{i=1}^{n} p_i$. There exists a solution if and only if the max flow is P. Equivalently, the problem has a solution if and only if $(\{s\} \cup D \cup H, \{t\})$ is a minimum capacity cut. Note that the capacity of the cut is P.

If the max flow is P, we extract our configuration as follows. For each edge $D_i \rightarrow H_j$ with non-zero flow, we assign doctor i to day j, which means we include j in L'_i.

Algorithm: The max flow in this network in P. The number of edges in limited to a maximum of $n * k$, since each doctor can only be connected to each day once. Hence we can use Ford Fulkerson algorithm to solve this problem.

Complexity

To build our flow graph, we require $O(n * k)$ time, as that is the previously estimated upper bound on the number of edges.

Ford Fulkerson has a complexity of $O(|E| * C)$. This would give us a time complexity $O(n * k * P)$ where $P = \sum_{i=1}^{n} p_i$.

Correctness

Claim 3. *If max flow is P, then the L'_i extracted from the flow satisfy the constraints given in the problem.*

1. For each element j in L'_i the doctor i is willing to work on day j since there is an edge $D_i \rightarrow H_j$ in the flow graph and the graph is constructed with the property that an edge exists between the doctor and a day if and only if the doctor is willing to work on that day.
2. We will show that the total number of doctors working on a particular day i is p_i. Let $l_{i,j}$ be the allocation of doctor D_j to the day H_i. Since the max flow is $P = \sum_{i=1}^{n} p_i$, every edge $H_i \rightarrow t$ from a day i to the node t carries a flow equal to its capacity which in turn implies that $\sum_j l_{i,j} = p_i$ satisfying the requirement of the day H_i.

Claim 4. *We now prove: If there is an allocation satisfying the constraints, then there exists a flow with flow value P in our network. Further, the maxflow of our network is P.*

Let $l_{i,j}$ be the allocation of doctor D_j to the day H_i. We will show how to construct a flow in our network whose value is P. Since our network has a cut of capacity P, we will then conclude that the max flow is P.

Since the allocation of doctors to days satisfies the constraints, there is an edge from D_j to H_i in the flow graph for each $l_{i,j}$ in the allocation. Let the flow on the edge $s \rightarrow D_j$ be $\sum_i l_{i,j}$, which is the number of days the doctor j is assigned to. This has to be less than $|L_j|$ since we only have $|L_j|$ edges leaving D_j each with capacity 1. Let the flow on the edge $D_j \rightarrow H_i$ be 1 and the flow on the edge $l_i \rightarrow t$ be $\sum_j l_{i,j} = p_i$. The flow on all other edges is zero.

It is clear from the construction that the flow is valid, that is, the flow satisfies the capacity and conservation constraints and the value of the flow is P.

Part 2

Constraints

1. Each L'_j contains at most c days that are not in L_j.
2. If we consider the whole set of lists L'_1, L'_2, ..., L'_k, it causes exactly p_i doctors to be present on day i, for i = 1, 2, . . . , n.

Description

We retain the graph from above but add an extra node for each doctor to account for the extra c days as shown

Nodes

1. s and t
2. $D = \{D_j | D_j$ is used to represent when a doctor is willing to work$\}$
3. $d = \{d_j | d_j$ is used to represent when a doctor is not willing to work$\}$
4. $H = \{H_i | H_i$ is a day$\}$

   The set of nodes for the flow graph is $\{s, t\} \cup D \cup d \cup H$.

Edges

1. $s \to D_j$ with capacity $|L_j|$, for each doctor
2. $s \to d_j$ with capacity c, for each doctor
3. $H_i \to t$ with capacity p_i, for each day
4. $D_j \to H_i$ with capacity 1, for each (i, j) pair for which doctor D_j is willing to work on day H_i
5. $d_j \to H_i$ with capacity 1, for each (i, j) pair for which doctor D_j is not willing to work on day H_i

Solution: Same as before, let $P = \sum_{i=1}^{n} p_i$. There exists a solution if and only if the max flow is P. Equivalently, the problem has a solution if and only if $(\{s\} \cup D \cup d \cup H, \{t\})$ is a minimum capacity cut. Note that the capacity of the cut is P.

If the max flow is P, we extract our configuration as follows. For each edge $D_i \to H_j$ and $d_i \to H_j$ with non-zero flow, we assign doctor i to day j, which means we include j in L'_i.

Algorithm: Same as before, the max flow in this network in P. The number of edges in limited to a maximum of $n * k$, since each doctor can only be connected to each day once. Hence we can use Ford Fulkerson algorithm to solve this problem.

Complexity

To build our flow graph, we require $O(n * k)$ time, as that is the previously estimated upper bound on the number of edges.

Ford Fulkerson has a complexity of $O(|E| * C)$. This would give us a time complexity $O(n * k * P)$ where $P = \sum_{i=1}^{n} p_i$.

Correctness

Claim 5. *If max flow is P, then the L'_i extracted from the flow satisfy the constraints given in the problem.*

1. There are at most c elements in L'_i where the doctor i is not willing to work on day j. There is an edge $D_i \to H_j$ in the flow graph if and only if the doctor i is willing to work on day j. These edges do not cause the doctor to work on days when he/she is unwilling to do so. There is an edge $d_i \to H_j$ in the flow graph if and only if the doctor i is not willing to work on day j. However, the capacity of the incoming edge to d_i is c, which means at most c such days can be chosen due to the flow conservation constraint of the flow network.

2. We will show that the total number of doctors working on a particular day i is p_i. Let $l_{i,j}$ be the allocation of doctor D_j to the day H_i and $l'_{i,j}$ be the allocation of d_j to the day H_i. Since the max flow is $P = \sum_{i=1}^{n} p_i$, every edge $H_i \to t$ from a day i to the node t carries a flow equal to its capacity which in turn implies that $\sum_j l_{i,j} + \sum_j l'_{i,j} = p_i$ satisfying the bandwidth requirement of the day H_i.

Claim 6. *We now prove: If there is an allocation satisfying the constraints, then there exists a flow with flow value P in our network. Further, the maxflow of our network is P.*

Let $l_{i,j}$ be the allocation of doctor D_j to the day H_i for days i in which the doctor j is willing to work and $l'_{i,j}$ be the allocation of d_j to the day H_i for days i where the doctor is unwilling to work. We will show how to construct a flow in our network whose value is P. Since our network has a cut of capacity P, we will then conclude that the max flow is P.

Since the allocation of doctors to days satisfies the constraints, there is an edge from D_j to H_i in the flow graph for each $l_{i,j}$ and from each d_j to H_i in the flow graph for each $l'_{i,j}$in the allocation. Let the flow on the edge $s \to D_j$ be $\sum_i l_{i,j}$, which is the number of willing days the doctor j is assigned to. This has to be less than $|L_j|$ since $l_{i,j}$ represents the fact that dcotr j is allocated to day i and doctor j is willing to work on day i. Let the flow on the edge $s \to d_j$ be $\sum_i l'_{i,j}$, which is the number of unwilling days the doctor j is assigned to. This has to be less than c given the constraints of the problem. Let the flow on the edge $D_j \to H_i$ for each $l_{i,j}$ be 1, flow on the edge $d_j \to H_i$ for each $l'_{i,j}$ be 1 and the flow on the edge $H_i \to t$ be $\sum_j l_{i,j} + l'_{i,j} = p_i$. The flow on all other edges is zero.

It is clear from the construction that the flow is valid, that is, the flow satisfies the capacity and conservation constraints and the value of the flow is P.

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.

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

Solution: Cellular network

Input: A list of sites for establishing cell towers with costs $c_1, \cdots, c_n$ and benefits b_{ij} obtained when towers are established at sites i and j.

Output: A subset S of sites such that the net benefit is maximized where the net benefit $Net(S)$ of selecting a set of sites is defined as $Net(S) = \sum_{i,j \in S} b_{ij} - \sum_{i \in S} c_i$

We will construct the following flow graph and use its minimum cut to extract a set of sites to maximize the net benefit.

In addition to the source and sink nodes s and t, the flow graph includes a bipartite graph with left nodes L labeled by pairs $\{v_{ij} \mid 1 \leq i \leq n, 1 \leq j \leq n\}$ and right nodes R labeled by $\{u_1, \cdots, u_n\}$. For every i and j, the flow graph contains the edges (v_{ij}, u_i) and (v_{ij}, u_j) with infinite capacities. The flow graph also contains edges (s, v_{ij}) for every node $v_{ij} \in L$ with capacity $b_{ij} > 0$ and edges (u_j, t) with capacity c_j for every $u_j \in R$.

We will apply preflow-push algorithm to compute the maximum flow and a minimum cut in the flow graph. Let $S \subseteq R$ be the set of vertices on the source side of the minimum cut. We select the sites in S as our solution.

For correctness, we will establish a correspondence between sets of sites and *normal* finite capacity cuts. We will first develop some notation for finite capacity cuts and define normal finite capacity cuts.

Consider a finite capacity cut $C = (A, B)$ in the flow graph. Let $E_A = A \cap L$ and $V_A = A \cap R$. E_A is the set of edge nodes v_{ij} where the node v_{ij} is on the source side of the cut. Similarly V_A is the set of nodes u_i which are on the source side of the cut. Let $L - E_A$ be the rest of the edge nodes v_{ij} which are on the sink side of the cut. Let $R - V_A$ be the rest of the nodes which are on the sink side of the cut. Since C is a finite capacity cut, it must be the case that for each $v_{ij} \in E_A$, both of its outgoing edges will end up in V_A for otherwise the cut will have infinite capacity.

The capacity of the cut (A, B) is given by

$$\sum_{u_i \in V_A} c_i + \sum_{v_{ij} \in L - E_A} b_{ij}$$

We say that a finite capacity cut (A, B) is *normal* if every edge node $v_{ij} \in L - E_A$ has at least one outgoing edge to the set $R - V_A$. In other words, it is not the case that any edge node $v_{ij} \in L - E_A$ has both of its outgoing edges ending up in the set V_A.

While not all finite capacity cuts are normal. we observe that minimum cuts are normal. If a minimum cut (A, B) were not normal, it will have an edge node $v_{ij} \in L - E_A$ with both of its outgoing edges ending up in V_A. Move v_{ij} from B to A, that is, from sink side of the cut to the source side. The modified cut will have lower capacity since $b_{ij} > 0$.

The capacity of a finite capacity normal cut can be written as

$$\begin{aligned}
\sum_{u_i \in V_A} c_i + \sum_{v_{ij} \in L - E_A} b_{ij} &= \sum_{u_i \in V_A} c_i + \sum_{u_i \in R - V_A \text{ or } u_j \in R - V_A} b_{ij} \\
&= \sum_{u_i \in V_A} c_i + \sum_{u_i \in R - V_A \text{ or } u_j \in R - V_A} b_{ij} + \sum_{u_i \in V_A \text{ and } u_j \in V_A} b_{ij} - \sum_{u_i \in V_A \text{ and } u_j \in V_A} b_{ij} \\
&= m - \Big(\sum_{u_i \in V_A \text{ and } u_j \in V_A} b_{ij} - \sum_{u_i \in V_A} c_i \Big) \\
&= m - Net(V_A)
\end{aligned}$$

where $m = \sum_{1 \le i, j \le n} b_{ij}$.

Now we define cuts corresponding to sets of sites. For each set $S \subseteq R$ of sites, we define a cut $(A(S), B(S))$ where $A(S) = \{s\} \cup S \cup \{v_{ij} | i, j \in S\}$ and $B(S)$ is the set of remaining nodes in the flow graph including t. In particular, $B(S) = \{t\} \cup R - S \cup \{v_{ij} | i \notin S \text{ or } j \notin S\}$. Clearly $(A(S), B(S))$ is a finite capacity cut since none of the edges going out of $A(S)$ to $B(S)$ have infinite capacity. Moreover, $(A(S), B(S)$ is a normal cut since each $v_{ij} \in B(S)$ has at least one of its outgoing edges going to $R - S$. The capacity of the cut $(A(S), B(S))$ is

$$\sum_{u_i \in S} c_i + \sum_{u_i \in R - S \text{ or } u_j \in R - S} b_{ij} = m - Net(S)$$

We have thus established the following claims so far.

Claim 7. *1. Any minimum cut is a normal finite capacity cut.*

2. *For every set $S \subseteq R$ of sites, there is a corresponding normal finite capacity cut C such that the capacity of C is equal to $m - Net(S)$.*

3. *For every normal finite capacity cut (A, B), there exists a a set of sites V_A such that the cut capacity is equal to $m - Net(V_A)$.*

From these facts, we conclude that the set of sites extracted from a minimum cut maximizes the net benefit.

Run time It takes $O(n^2)$ time to build the bipartite graph. The run time using preflow-push algorithm would be $O(n^3)$, which is the dominating factor in the total run time.