Algorithm Analysis and Design - Assignment 4

Aryaman Kolhe
2022121002

Q1. Prove / disprove :

G : Arbitrary flow network with source s, sink t, positive integer capacity c_e $\forall e \in E(G)$

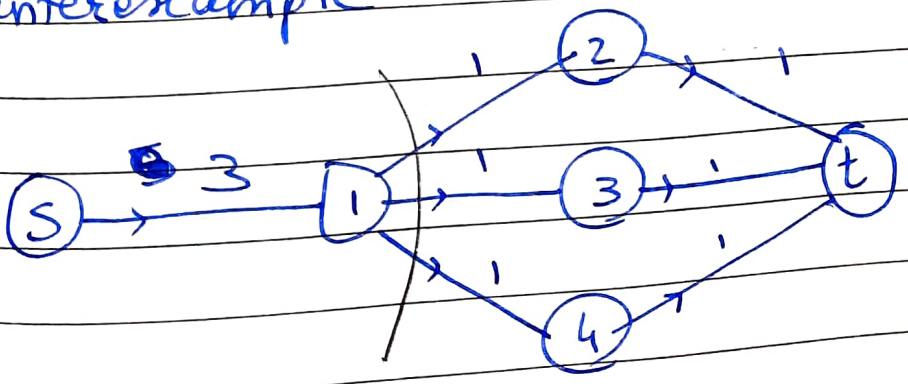
(A, B) = minimum s-t cut wrt $c_e : e \in E$. Does G with $\{1 + c_e : e \in E\}$ have the same min s-t cut (A, B) ? (Is (A, B) still a min st cut)

For min s-t cut A, B,
$A \subseteq V$, $B \subseteq V$, $A \cup B = V$, $A \cap B = \phi$.
and $c(A, B) = \sum_{(i,j) \in E} c_{ij} d_{ij}$ is minimum

$$\begin{cases} d_{ij} = 1 & \text{if } i \in A, j \in B \\ d_{ij} = 0 & \text{otherwise.} \end{cases}$$

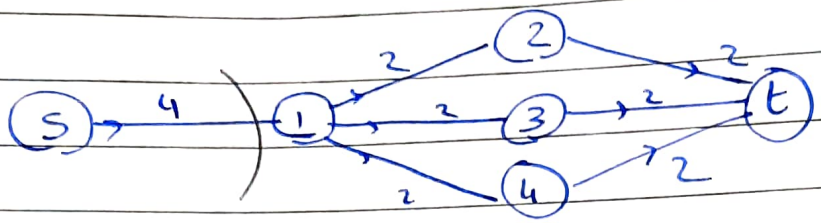
The above claim is false

Counterexample -



$A = \{S, 1\}$
$B = \{2, 3, 4, t\}$

After adding 1 to each capacity,

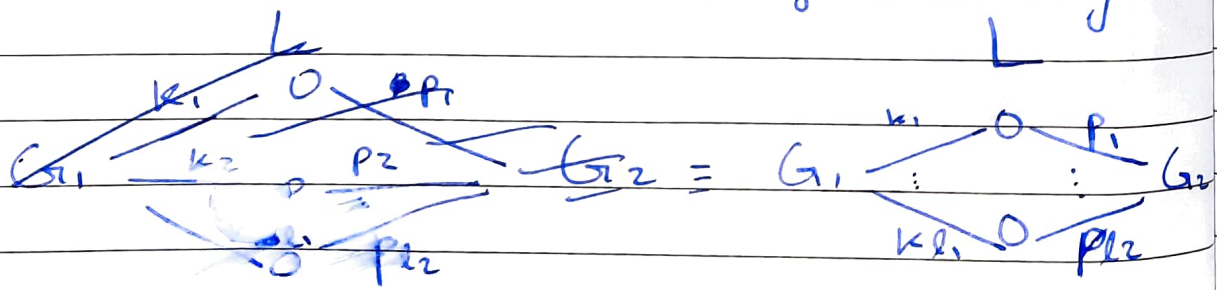


Min Cut $(C, \bar{C}) \Rightarrow C = \{s\}$

$\bar{C} = \{1, 2, 3, 4, t\}$

We can generate counter examples easily, with the following method -

- Generate an arbitrary flow network G_0.
- Find its min s-t cut : (A_0, B_0)
- If the value of $c(A_0, B_0) = x$, then split the graph G_0 into 2 subgraphs G_1 and G_2
- Create a layer between G_1 and G_2 $\Rightarrow$ sum of edges connecting G_1 to G_0 is x. and at least x for G_2 to G_0.



$\sum_{i=1}^{l} k_i = x$ Then after adding 1, min cut will change to $(x+1)$, but $(G_1, L \cup G_2)$ will not remain a min s-t cut.

Q2. n patients, k hospitals. Each hospital takes at most $\lceil \frac{n}{k} \rceil$ people.

We can model the problems as follows:

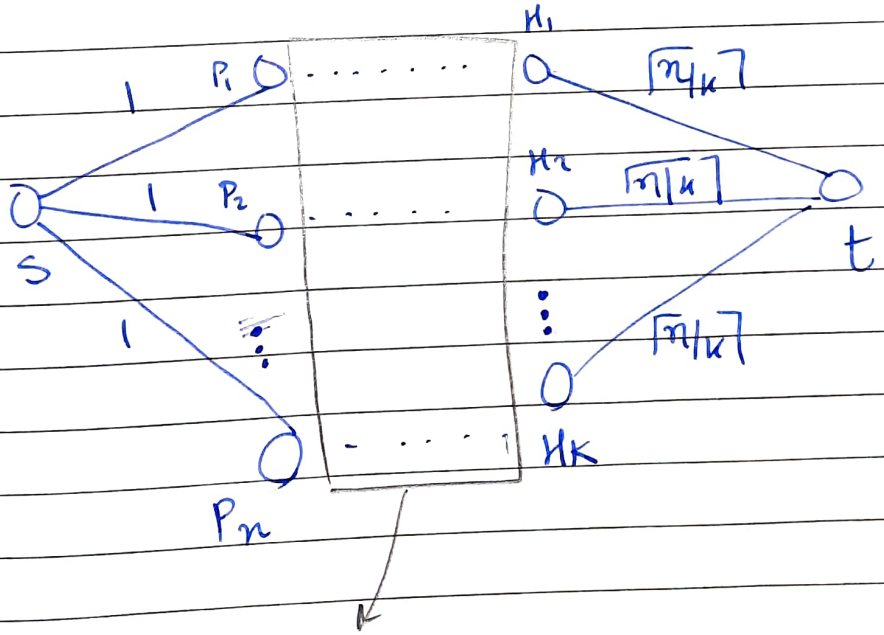
→ First create a bipartite graph using patients P_i and Hospitals H_j

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

$j = 1, 2, \ldots, k$

Connect node P_i with H_j iff patient i can reach hospital j within a 30 minute drive.

→ Now model the above graph as a flow problem by introducing a source (abstract source of n patients), and a sink (abstract sink that sums the number of patients that reach all hospitals.



$(P_i, H_j) \in E(G)$ iff H_j is reachable for P_i in 30 minutes

All P_i are connected to s with capacity 1.

All H_j are connected to t with capacity $\lceil n/k \rceil$ (not more than $\lceil n/k \rceil$ patients can "leave" the hospital).

→ Now find the maximum flow from s to t. Let this be F.

If $F < n$ then report "NO", as it is not possible to fulfill the constraints.

else report YES. ($F = n$).

Note that $F > n$ is not possible as $\alpha(\{s\}, V - \{s\})$ cut has flow n.

Apply Ford Fulkerson to find the max flow in this graph.

Start with initial flow as the min capacity $\forall\, ce \in \bar{t}$

While there is an augmenting path from 's' to 't',

→ Add the path flow to current max flow F

if $F < n$
return False
else
return True

To find the augmenting path, create the respective residual graph and update.

Time = $O(\text{max flow} \cdot (|V| + |E|))$

$= O(n((n+k+2) + nk + nk))$

which is in Polynomial Time.

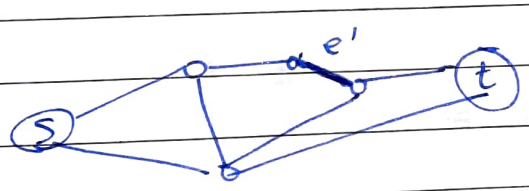
Q3. $G = (V, E)$ $\exists\, c_e \;\forall\, e \in E$

$s \in V, \quad t \in V.$

Max s-t flow in G_f is given by flow $f_e \;\forall\, e \in E$. f is acyclic.

$\nexists$ cycle in G_f on which all edges carry positive flow.

$f \in \mathbb{Z}$. If we pick a specific edge e' $\exists\, e' \in E$ and reduce its capacity by 1 unit, then show how to find max flow in the new graph G' in time $O(m+n)$ $\exists\; m = |E|, \quad n = |V|$.



CASE 1

If flow of e' is less than its capacity (strictly), then flow f is already the maxflow. ($f_{e'} < c_{e'} \rightarrow f_{e'} \leq c_{e'} - 1$)

CASE 2

$f_{e'}$ cannot be strictly greater than $c_{e'}$ (by definition of f)

CASE 3

If $f_{e'} = c_{e'}$, and if we reduce $c_{e'}$ by 1 : $c_{e'2} = c_{e'} - 1 < f_{e'}$, which violates the condition $c_{e'} \geq f_{e'}$ of f.

To fix this, we can update the flow $f_{e'2}$:

$f_{e'2} = f_{e'} - 1$.

However let $e = (u, v) \in E$

Now, the conditions,

$$\sum_{e: \text{into } u} f_e = \sum_{e: \text{out of } u} f_e$$

{conservation of flow gets violated}

and

$$\sum_{e: \text{into } v} f_e = \sum_{e: \text{out of } v} f_e$$

get violated.

So we need to adjust f in such a way that retains this property.

→ Find a path from s to u in $O(m+n)$ with DFS
Path P_{su}

→ Find a path P_{vt} from v to t in $O(m+n)$ with DFS.

→ Then reduce flow in these paths by 1
$\forall\ e \in P_{su}$ do $f_{e2} = f_e - 1$
$\forall\ e \in P_{vt}$ do $f_{e2} = f_e - 1$

Now flow is conserved.
Let this new graph be G_1

⇒ Claim:
Max flow of G_1 is either $f-1$ or f

Proof:

Max flow cannot be larger than f because f_1 is derived from f, by reducing one edge's capacity by 1.

Also, the flow of f_1 is at least $f-1$ since we found a legitimate graph f_1 that satisfies all flow constraints, and has flow $f-1$.

In one iteration of the Ford Fulkerson algorithm, the flow stays the same, or increases by at least one.

∴ Algorithm to find flow of f_1:

→ Construct augmented_graph $(f_1) = f_2$ in $O(m)$ time (can be done by considering residual capacity function which can be computed in $O(1)$ $\forall m \in E$).

→ Find s-t path in f_2. $O(m+n)$ via DFS. Find the bottleneck edge, reverse it and update edges on the path.

→ If flow increases by 1, the final flow is f.
Else it is $f-1$.

∴ Time $= 2\,O(m+n) + O(m) + O(m+n)$

- $2\,O(m+n)$: $s \to u$, $v \to t$
- $O(m)$: Create Augmented f_2
- $O(m+n)$: s-t in f_2

$= O(m+n)$ //

Q4. $G = (V, E)$ $s,t \in V$ $c_e = 1 \quad \forall e \in E$

k - parameter.

Delete k edges to reduce s-t flow in G as much as possible.

$G \xrightarrow[\text{edges optimally}]{\text{delete } k} G' \qquad F \subseteq E \quad |F| = k$

The ideal strategy will be to find a cut that splits V into A, B. We can then remove outgoing edges of A to reduce the flow of G.

To make this optimal, choose a min cut of G. (choosing any other cut will result in deleting an edge that does not reduce the flow optimally).

To find min cut in Polynomial time.

→ Run Ford Fulkerson algo in polynomial time $O(|f^*| \cdot E)$, and consider the final residual graph G_R.

→ Use DFS to find the set of vertices that are reachable from the source in G_R.
T. = $m \cdot O(m+n) \Rightarrow O(m(m+n))$

→ Let this set of vertices be A.
Let $B = V - A$

Then, (A, B) is a min cut of G.

Removing an outgoing edge between A and B (outgoing from A), will not change the min cut. (A, B) will be the new min cut again (this removes the problem of non-unique min cuts). This can be done k times.

Let $E_{A,B}$ be the outgoing edges from A to B. (Here $F = E_{A,B}$)

If $|E_{A,B}| \leq k$, then remove all $e \in E_{A,B}$ to get max flow = 0.

Else remove any k edges $\in E_{A,B}$.
This will minimize the max s-t flow in $G' = (V, E - F)$

Time: $O(f^* |E|) + O(m(m+n)) + O(k)$
polynomial
$\Rightarrow$ polynomial.

Flow after k edges from $E_{A,B}$ have been removed is

$$f' = \begin{cases} f - k & |E_{A,B}| > k \\ 0 & |E_{A,B}| \leq k \end{cases}$$

Q5. $G = (V, E)$ $s \in V$, $t \in V$.
$c_e \geq 0$

Polynomial time to decide whether G has a unique minimum s-t cut

We know from class that the problem of finding the minimum s-t cut is the same as finding the maximum s-t flow in G.

[1] In Q4. we discussed how to find a minimum cut in polynomial time.

Idea:

→ Find flow f in G using Ford Fulkerson's algorithm in polynomial time.

→ Find a min-cut in G in polynomial time [1] ↳ (A, B).
Let $E_{A,B}$ be the outgoing edges from A to B.

→ Increase the capacity of any edge $e \in E_{A,B}$ by 1.
Now if the min cut was unique in G then the max flow of G would have changed.

→ Find the max flow of the new graph = f'.

if $f' \neq f$: the min cut is unique
else if $f' == f$: the min cut is not unique
($\exists$ another min cut which gave $f' == f$)

All steps are in polynomial time. ∴ the entire procedure is in polynomial time.

Q6. Bipartite matching $G_1 = (V_1, V_2, E)$

$(|V_1| = |V_2| = n)$

Given: G_1 has a perfect matching.

To get it, → add 1 to all $ce : \forall e \in E$

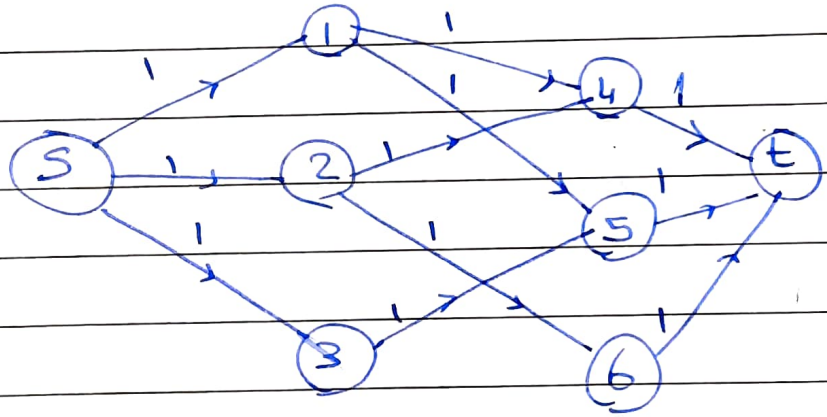
→ Add nodes s and t

such that s is connected to all $u \in V_1$ with capacity 1 and t is connected to all $v \in V_2$ with capacity 1.

To prove that the max. flow in this network gives us a perfect matching.

Proof:

Example Graph



⇒ A matching M in G_1 is a subset $M \subseteq E$
∋ each node appears in at most one edge in M.

⇒ A perfect matching M_P in G_1 is one in which every vertex of G_1 is incident to exactly one edge of the matching. (vertices not including s and t)

Suppose there is a matching in G' with k edges $(x_{i_1}, y_{i_1}), \ldots, (x_{i_k}, y_{i_k})$.

$f(e) = 1$ $\forall$ paths of the form $s \to x_{ij} \to y_{ij} \to t$.

- All capacities are integral. So the flow will be integral.
- All capacities are 1 which implies
  - → We can use an edge to send 1 unit of flow
  - → Not use an edge.
- Let M be be the set of edges outgoing from A that we "use".

It is enough to show that M is a matching (1), and that M is the largest possible matching (2).

Proof that M is a matching:
→ We can choose at most one edge leaving any node in A.
→ We can choose at most one edge entering any node in B.

(Choosing more than 1 will violate conservation of flow)

The corresponding correspondence b/w flows and matchings -

→ If ∃ matching of k edges, then ∃ flow of value k

→ If ∃ flow f of value k, then ∃ matching with k edges

(can be verified with sample graph given before).

PROOF BY CONTRADICTION

Proof that M is as large as possible

We find the maximum flow f (with k edges), which corresponds to a matching of k edges.

If there was a matching with $> k$ edges, then we would have found a flow with value $> k$, which contradicts that f was maximum

∴ M is maximum and we have found a perfect matching

(It was given that G contains a perfect matching)

Q.E.D

—x—

→ Collaborated with Yatharth Gupta on problems 3 and 4.

→ For problem 6, referred to
- Algorithm Design - Kleinberg and Tardos
- cs.cmu.edu lecture slides.