CS345 Assignment 4

Siddharth Agrawal(150716)
AviRaj(150168)

October 31, 2017

1 Question 1

Algorithm 1 Algorithm to find the sequence of paths $P_0, P_1, ..., P_b$

```
procedure FindSeq(G_0, G_1, ....., G_b, s, t)
    sP[0] ← shortest s − t path in G_0
    Div[0] ← 0
    minCost[0] ← distance length of sP[0]
    i ← 1
    while i ≤ b do
        tP ← shortest s − t path in G_i [1]
        d ← length of path tP
        minCost[i] ← d + minCost[i − 1] + K
        j ← i − 1
        while j ≥ 0 do
            tP ← shortest s − t path in G_i ∩ G_{i−1} ..... ∩ G_j [1]
            if tP does not exist then
                break
            end if
            d ← length of path tP
            if j ≠ 0 then
                tempCost ← (i − j + 1) × d + K + minCost[j − 1]
            else
                tempCost ← (i + 1) × d
            end if
            if tempCost < minCost[i] then
                minCost[i] ← tempCost
                Div[i] ← j
                sP[i] ← tP
            else
                break
            end if
            j ← j − 1
        end while
        i ← i + 1
    end while

    i ← b
    while i ≥ 0 do
        j ← i
        while j ≥ Div[i] do
            P_j ← sP[i]
            j ← j − 1
        end while
        i ← Div[i] − 1
    end while
    return P_0, P_1, ....., P_b
end procedure
```

[1] calculated using BFS algorithm

In the above algorithm, the meaning of the various variables used are -

$$\begin{aligned} Div[i] &\rightarrow j \text{ such that } P_j = P_{j+1} = ... = P_i \\ sP[i] &\rightarrow \text{path } P_i = P_{i-1} = ... = P_{Div[i]} \text{ such that cost is minimized} \\ minCost[i] &\rightarrow \text{minimum cost of paths } P_0, P_1, ..., P_i \end{aligned}$$

Time Complexity

The outer while loop performs b iterations.
In the inner loop - $O(m+n)$ time to find the shortest $s-t$ path and $O(p(n))$ time to find the intersection of the graphs where $p(n)$ is a polynomial in n.
Thus time complexity $= O(b^2 \times (m + p(n))) =$ polynomial.

Proof of Correctness

We prove this using induction on b.
Base Case. Since only one graph is considered, hence we only have to minimize the $l(P_0)$ term, hence we choose shortest $s-t$ path in G_0.
Induction Hypothesis. Given the $minCost[i]$ values for $i = 0, 1, 2, ..., k$, the value of $minCost[k+1]$ is correctly calculated by our algorithm.
Proof for Induction Hypothesis. The path P_{k+1} can potentially be the shortest $s-t$ path in G_{k+1} or in $G_{k+1} \cap G_k$ (here $P_{k+1} = P_k$) ... or in $G_{k+1} \cap G_k ...... \cap G_0$ (here $P_{k+1} = P_k = ... = P_0$).
Note:- no other path can be potentially P_k since $changes()$ function only considers adjacent paths,i.e., $P_i \& P_{i+1}$ are compared.
Out of all these potential values of the paths, we have to chose that which minimizes the $minCost[k+1]$.
This is what is done in the inner loop of our algorithm.

2 Question 2

Part 2

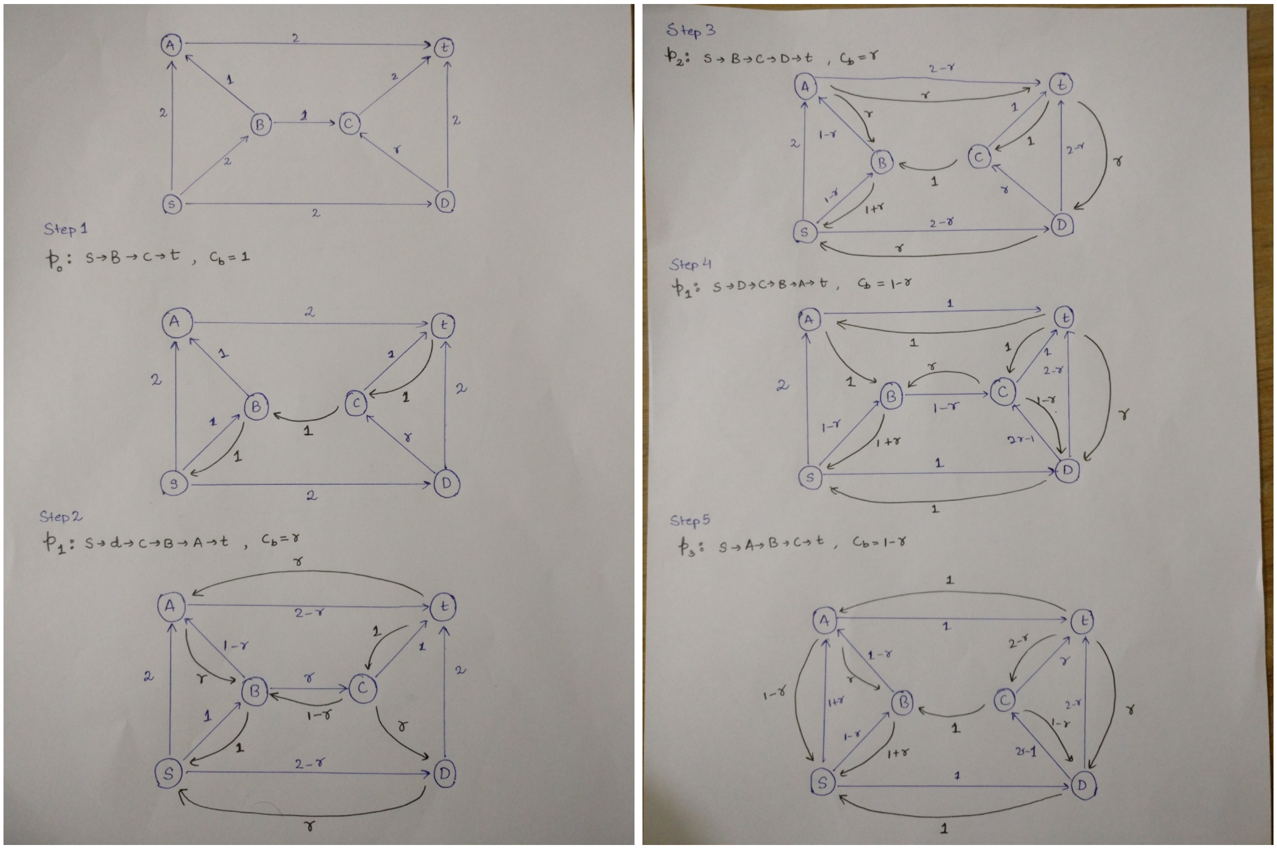

Figure 1: The above two figures show the change in flow and G_f according to the paths taken in the Ford-Fulkerson algorithm

Non-terminating example of Ford-Fulkerson algorithm

Let p_0 , p_1, p_2, p_3 be three augmenting paths in residual network G_f (evident from the diagram) defined as-

p_0: s→B→C→t

p_1: s→D→C→B→A→t

p_2: s→B→C→D→t

p_3: s→A→B→C→t

If we choose the augmenting paths as p_0, p_1, p_2, p_1, p_3 , p_1, p_2, p_1, p_3 ,…. i.e, infinitely repeating the sequence of augmenting paths (p_1, p_2, p_1, p_3), we can prove that the Ford-Fulkerson algorithm never terminates

Observation 1: The bottleneck edges are going to be one of the edges from the set of directed edges E_1 ={ BA , BC , DC }

As there are 4 paths repeating, there can be 4 cases for k≥0

Step $(4k)^{th}$: bottle neck capacity = r^{2k} : bottle neck edge= BA : via path p_1

Step $(4k+1)^{th}$: bottle neck capacity = r^{2k} : bottle neck edge= BC : via path p_3

Step $(4k+2)^{th}$: bottle neck capacity = r^{2k+1} : bottle neck edge= DC :via path p_1

Step $(4k+3)^{th}$: bottle neck capacity = r^{2k+1} : bottle neck edge= BC :via path p_2

It can be shown that apart from these edges, no other edge is going to be the bottle

Neck edge in any of the four cases. For example in path of step $(4k+2)^{th}$, via path p_1

i.e, s→D→C→B→A→t ,

Observation 2: Edges in set E_1 are in geometric progression w.r.t quadruples .

After every 4 steps, they get multiplied by r^2.

Considering step $(4k+1)^{th}$

After Step 1: { BA, BC, DC } ={ r^0 , 0 , r^1 }

After Step 5: { BA, BC, DC } ={ r^2 , 0 , r^3 }

After Step $(4k+1)^{th}$: { BA, BC, DC } ={ r^{2k} , 0 , r^{2k+1} }

Similarly, we can consider the step $(4k)^{th}$ step $(4k+1)^{th}$ and step $(4k+3)^{th}$. There will be a multiplying factor of r^2 between consecutive quadruples.

From Observation 1 and Observation 2, we can say that is we chose the given sequence of augmenting paths, the flow in every step is bounded by one the edges in set E_1 , which in fact are geometrically decreasing! So, the sum of flow will converge to a finite value.

So, if we chose the sequence $\rightarrow p_0,\ p_1,\ p_2,\ p_1,\ p_3,\ p_1,\ p_2,\ p_1,\ p_3, \ldots$

Using the given sequence of augmenting paths, after (4k+1)th such path, the total sum of flow from s to t will be:

FLOW$= 1 + r + r + r^2 + r^2 + r^3 + r^3 + r^4 + r^4 \ldots\ldots\ldots + (r^{2k} + r^{2k+1} + r^{2k+1} + r^{2k+2}) + \ldots\ldots$

$= 1 + 2 (r + r^2 + r^3 + r^4 \ldots\ldots)$

$= 1 + 2/r$

$= 1 + 2(1+r)$

$= 3+2r = 4.2360$

< 5

Conclusion : In this case, The Ford-Fulkerson algorithm didn't not terminate. Also, it converged to a value not equal to the value of the maximum flow.

3 Question 3

Part 1

Algorithm

We convert the given problem into the problem of *flow with lower bound* (this problem has already been discussed in class and has a polynomial time algorithm).
Define a graph $G = (V, E)$ as-

Definition of vertices V - total $m + n + 2$ vertices

- a source vertex s and a sink vertex t
- m vertices - $b_1, b_2, ..., b_m$ - b_i denotes i^{th} balloon
- n vertices - $c_1, c_2, ..., c_n$ - c_i denotes i^{th} condition

Definition of edges E

- $\exists$ edge (s, b_i) $\forall i \in [m]$ and has lower bound $= 0$ and capacity $= 2$
- $\exists$ edge (b_i, c_j) if $\forall i \in [m], j \in [n]$ i^{th} balloon can measure j^{th} condition; lower bound $= 0$ and capacity $= 1$
- $\exists$ edge (c_j, t) $\forall j \in [n]$ and has lower bound $= k$ and capacity $= k$
- $\exists$ edge (t, s) with lower bound $= nk$ and capacity $= nk$

Now, check if a valid flow (satisfying the corresponding lower bounds) exists in G (using the algorithm discussed in class).

Time Complexity

In class, we have derived a polynomial time algorithm for checking if a valid flow with lower bounds exists in a graph. Hence, here also, the time complexity will be polynomial.

Proof of Correctness

Theorem 3.1. *There exists a way to measure n conditions using m different balloons (subject to the given constraints) if and only if there exists a valid flow in the above defined graph G (satisfying the corresponding lower bounds).*

Proof.

Theorem 3.1.1. *If there exists a way to measure n conditions using m different balloons (subject to the given constraints) then there exists a valid flow in the above defined graph G (satisfying the corresponding lower bounds).*

Proof. Let $f : E \to \mathbb{R}$ denote the flow in the graph G. Define f as follows-

- $\forall$ edges of the type (b_i, c_j), if i^{th} balloon measures the j^{th} condition, assign $f(b_i, c_j) = 1$
- $\forall$ edges of the type (s, b_i), assign $f(s, b_i) =$ number of conditions i^{th} balloon measures
- $\forall$ edges of the type (c_j, t), assign $f(c_j, t) =$ number of balloons which measure j^{th} condition
- assign $f(t, s) = nk$

Clearly, f satisfies all the lower bounds (because of the constraints to the balloon-condition problem).
Also, f follows the conservation of flow.
Hence flow f is valid.

$\square$

Theorem 3.1.2. *If there exists a valid flow in the above defined graph G (satisfying the corresponding lower bounds) then there exists a way to measure n conditions using m different balloons (subject to the given constraints).*

Proof. Let $f : E \rightarrow \mathbb{R}$ denote the given valid flow in the graph G.
$\forall$ edges of the type (b_i, c_j), if $f(b_i, c_j) = 1$ then measure j^{th} condition using i^{th} balloon.
Now to prove that the proposed solution to the balloon-condition problem follows the given constraints -

- since $f(s, b_i) \leq 2 \ \forall i \in [m]$, hence each balloon measures atmost 2 conditions
- since $f(c_j, t) = k \ \forall j \in [n]$, hence each condition is measured be exactly k balloons
- from construction of G it follows that each balloon only measures those conditions which it can measure (since edge exists between a balloon and condition only if that balloon can measure that condition)

$\square$

Using theorem 3.1.1 and theorem 3.1.2, our original theorem is proved.

$\square$

Part 2

Algorithm

To, accommodate the extra constraint - *sub-contractors* we change the construction of the graph G as described below -
Change in V - *new* total $m + 4n + 2$ vertices

- original $m + n + 2$ vertices remain as is
- add $3n$ more vertices - $c_{1a}, c_{1b}, c_{1c}, c_{2a}, c_{2b}, c_{2c}, ..., c_{na}, c_{nb}, c_{nc}$ - c_{jl} denotes the j^{th} condition measured by sub-contractor l (k=a,b,c; a,b,c denote the three different sub-contractors)

Change in E

- Remove all edges of the type b_i, c_j. Rest of the edges and their bounds & capacities remain same
- Add edges (b_i, c_{jl}), if $\forall i \in [n], j \in [m], l \in \{a, b, c\}$ i^{th} balloon belonging to sub-contractor l can measure j^{th} condition; lower bound=0 and capacity = 1
- Add edges (c_{jl}, c_j) $\forall j \in [m], l \in \{a, b, c\}$; lower bound = 0 and capacity = $k - 1$

Now, we check if a valid flow (satisfying the corresponding lower bounds) exists in this modified G (using the algorithm discussed in class).

Time Complexity

Using the same arguments as in part 1, our algorithm has polytime complexity

Proof of Correctness

The proof is the same as in part 1, except the part that the condition of the sub-contractors has been taken care of.
Since the capacity of edges of type (c_{jl}, c_k) is $k - 1$, hence a single condition (say j) cannot be measured by all k balloons belonging to the same sub-contractor (say l). Hence the constraint is satisfied.