CS345A

Assignment-4

Deepanshu Bansal(150219)

Q1

1.) Let $a_1, a_2, ..., a_k$ be an optimal solution J with corresponding execution times $t_{a1}, t_{a2}, ..., t_{ak}$ and deadlines $d_{a1}, d_{a2}, ..., d_{ak}$. So let's say we have i and j in this set J such that a_i is scheduled before than a_j with $d_{ai} > d_{aj}$. Let's say time just before executing i^{th} job is t. So we have

$$t + t_{ai} < d_{ai} \quad \text{and} \quad t + t_{ai} + t_{aj} < d_{aj}$$

Now if we prove that these two jobs can be swapped (there exists an optimal solution)that would mean that we can schedule all jobs according to the increasing order of their deadlines.

So we sure have

$$t + t_{aj} < d_{aj} \quad \text{as} \quad t_{ai} > 0$$

And using $t + t_{ai} + t_{aj} < d_{aj}$ and $d_{aj} < d_{ai}$ we get

$$t + t_{ai} + t_{aj} < d_{ai}$$

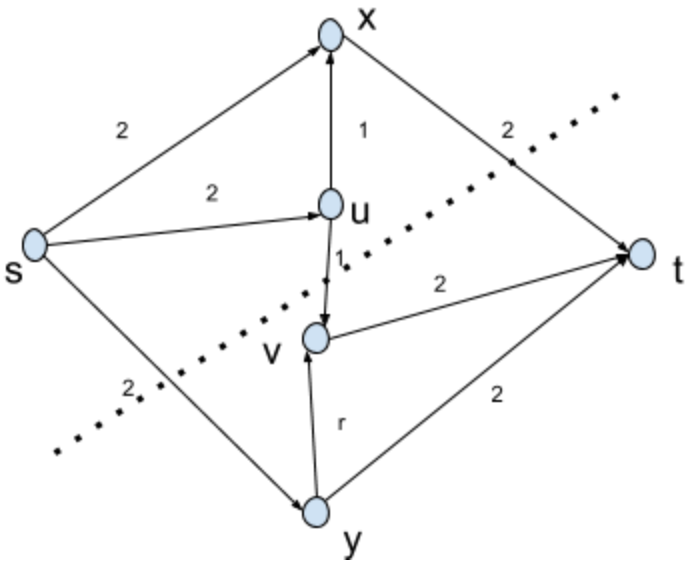
So we have an optimal solution with these two jobs scheduled according to their increasing deadlines.

Now we have $b_1, b_2, ..., b_k$ an optimal solution J with increasing order of deadlines and b_p is scheduled before b_q such that $d_{bp} < d_{bq}$. Hence proved.

2.) Scheduling Algorithm

- Sort all jobs according to increasing deadlines. Let it be A.
- For all elements of A
    - Pick the element with the earliest deadline. Let it be a_k.
    - Add a_k to J (optimal set).
    - Remove all the elements from A those clashing with a_k and whose deadlines cann't be met.
- Return J

Q2



We observed that before iterating sequence <p1,p2,p1,p3> that is for i = 0 we have flow f = 1.

After first time of iterating through paths <p1,p2,p1,p3> flow increases to f = 1 + 2r.

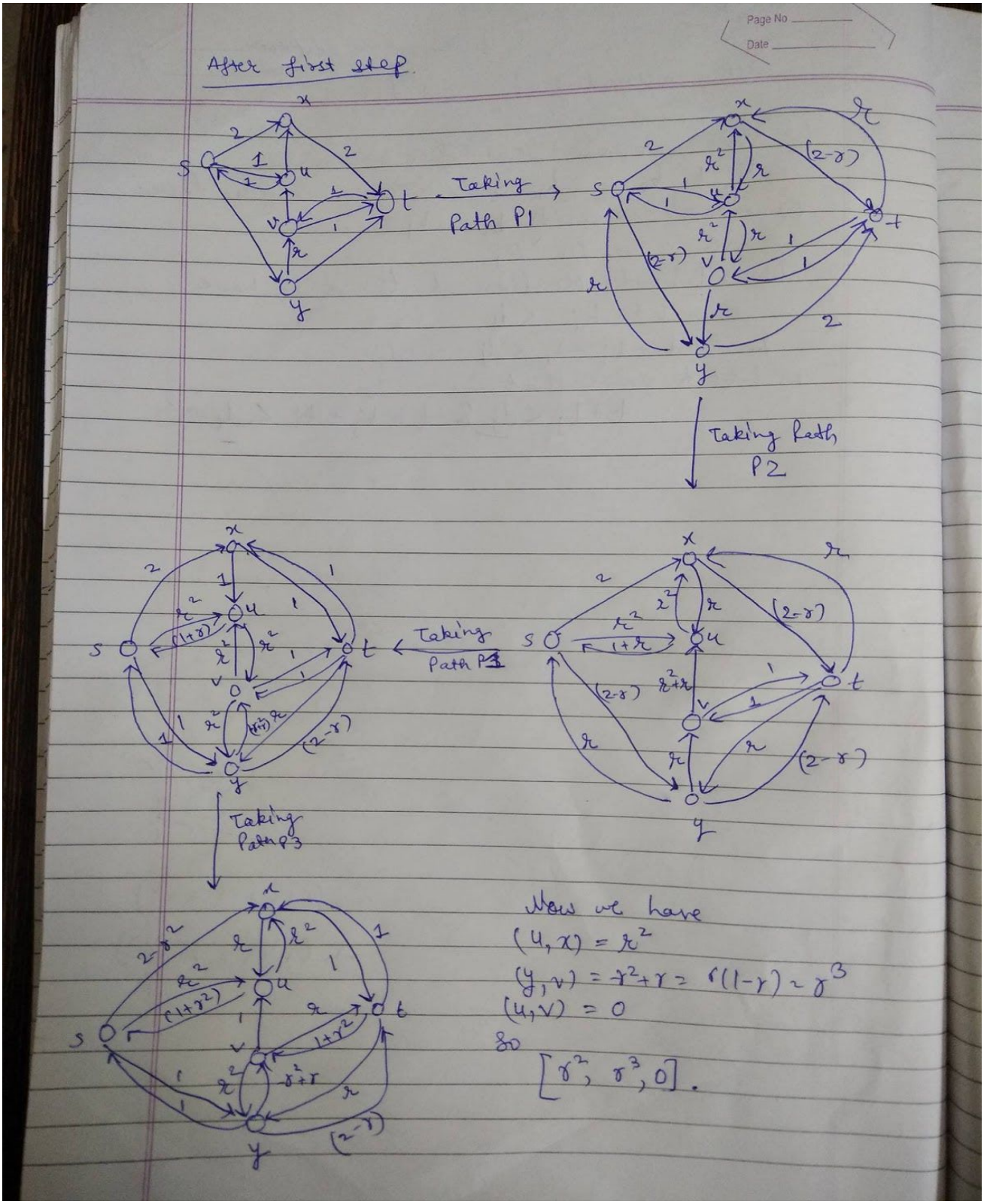
After second time of iterating through paths <p1,p2,p1,p3> flow increases to f = 1 + 2r + $2r^2$.

....

After i^{th} time of iterating through paths <p1,p2,p1,p3> flow increases to f = 1 + $2\sum r^i$

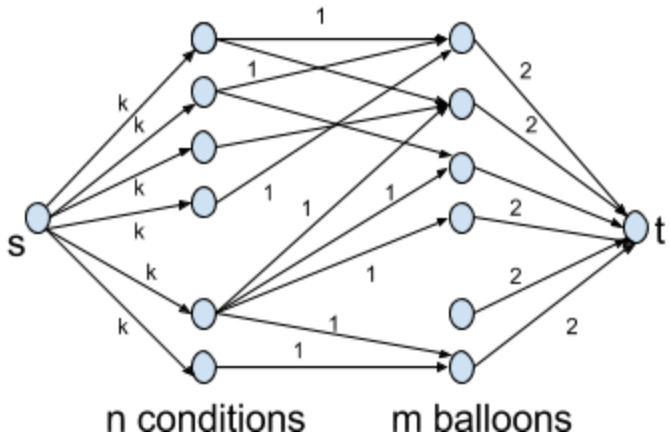
And this sum converges to sum f = 1 + 2/(r) = 3 + 2r .

But we know that max flow for this problem is 5 according to shown min-cut. And Ford-Fulkerson algorithm will continue to try to reach it but as we saw that even after infinite iterations it can make upto 3+2r at max . So it just keeps on running forever.


After first step
Taking Path P1
Taking Path P2
Taking Path P3
Now we have
(u, x) = r²
(y, v) = r² + r = r(1 − r) = r³
(u, v) = 0
so
[r², r³, 0].

Q3

a)



Create the bipartite graph G = (V, E) as following and compute the max flow with lower bound of this graph.

V = all conditions(n) and balloons(m) and a source s and sink t.

Let V1 be vertices corresponding to conditions and V2 corresponds to balloons.

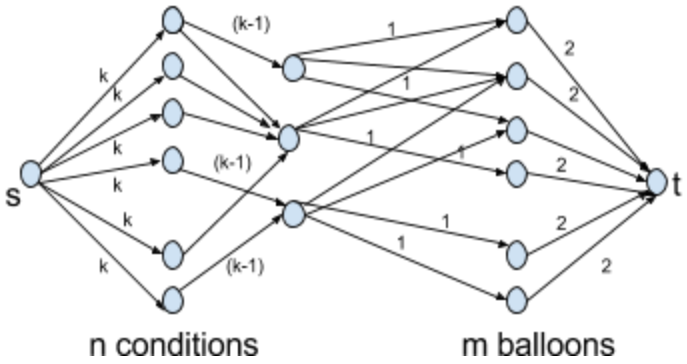
E = 1) Add an edge (i,j) for i ϵ V1 and j ϵ V2 if balloon j can measure condition i. (With the help of set S) . Give each of these edges the capacity c(i,j) = 1.

2) Add an edge (j,t) for j ϵ V2 with capacity c(j,t) = 2.

3) Add an edge (s,i) for i ϵ V1 with capacity l(s,i) = k.

Now compute the max flow of this graph with these integer capacities and using Ford-Fulkerson algorithm and if it turns out to be at least “nk” so we can output that there is a way to measure each condition by k different balloon otherwise not.

b)



Now change the graph with V3 as vertices corresponding to contractors.ie.

(i,j) for i ϵ V1 and j ϵ V2 we break it into two edges (i,p) and (p,j) where p ϵ V3 .

With c (i,p) = k-1 and c(p,j) = 1 this will make sure that all k measurements are not from a single contractor. Now we just compute the max flow by using Ford-Fulkerson algorithm and if it turns out to be at least “nk” so we can output that there is a way to measure each condition by k different balloon otherwise not.