CS345 : Theoretical Assignment 4

Pranjal Prasoon (150508)
Raktim Mitra (150562)

September 2017

1 Question 1 (Hard Version)

1.1 Algorithm (with implicit Proof of Correctness)

The central idea is that in the sequence $G_0, G_1, G_2, ...G_i$ for all i less than equal b, there will be some $0 \leq j \leq i$ such that from $j+1$ to i, the $P_k = P_{k+1}$ for $j+1 \leq k \leq i$.

Now, for $0 \leq m, n \leq b$, we define a graph $G'(m,n)$ such that $G'(m,n)$ is made up of common edges in $G_m, G_{m+1}, ..., G_n$.

A path $P'(m,n)$ is length of the shortest s-t path in $G'(m,n)$.

Now, we use **DP** and define a function $min_{cost}(i)$which stores the minimum cost of paths from $G_0, ..., G_i$. Now, we define the DP-equations as follows :

$$min_{cost}(i) = min[min[min_{cost}(j) + (i-j)P'(j+1,i) + K; 0 < j < i], (i+1)P'(0,i)] \quad (1)$$

Note that the first part corresponds to the last "change" being made as we go from G_j to G_{j+1} and from G_{j+1} to G_i, the shortest s-t path is the same and so, it has to be the shortest s-t path in G' (according to the definition of G') and hence, its length will be $P'(j+1,i)$. We run this part over every j from 0 to less than i and take the minimum of it.

The second part corresponds to the case when there is no change in the path sequence at all and hence, the $P'(0,i)$ will be the path in each case and hence, the expression.

These two cases are totally exhaustive and are the only two cases, as is trivially observed.We thus accept the path sequence corresponding to $min_{cost}(b)$.

1.2 Pseudo code

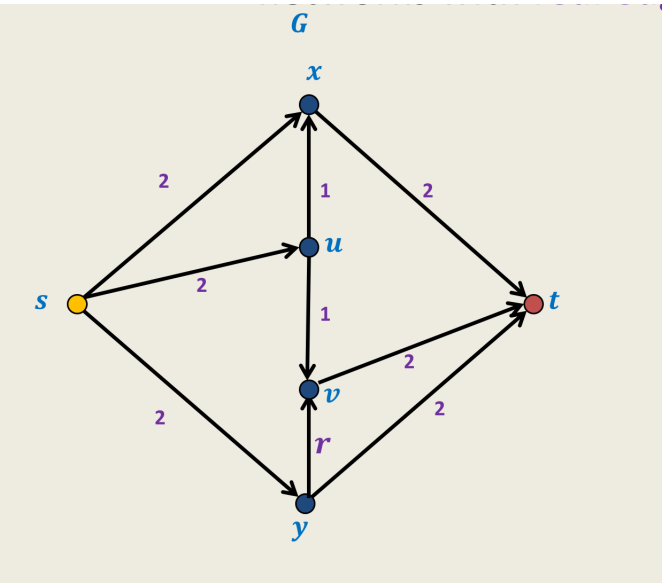
1.3 Time complexity analysis

- Pre-processing time : For forming one $G'(i,j)$, time taken is $O(n^2 * (i-j))$ which reduces to $O(n^2 b)$, so for all $O(b^2)$ possible pairs, time taken will be $O(n^2 b^3)$
- Algorithm's running time : There are two for loops, fro time will be $O(n^2)$.
- Overall time complexity = $O(n^2 b^3 + n^2) = O(n^2 b^3)$

Algorithm 1: Finding Path sequence with minimum cost

$P_0 \leftarrow ShortestS - TpathInG_0$
Function *assign weights(T)*
  for *i in [0,n]* **do**
    $temp_1 \leftarrow (i+1)P'(0,i)$
    $temp_{path} \leftarrow P_0$
    for *j in [1,i-1]* **do**
      $temp_2 \leftarrow min_{cost}(j) + (i-j)P'(j+1,i) + K$
      if $temp_1 > temp_2$ **then**
        $temp_1 \leftarrow temp_2$
        $temp_{path} \leftarrow P_j$
    $P_i \leftarrow \text{temp}_{path}$

2 Question 2 (Easy Version)



As was observed in the class, the maximum flow is 5. We label the **u-x** edge as e_1 , the **y-v** edge as e_2 and the **u-v** edge as e_3.

Let $a_0 = 1$ and $a_1 = r$, where r is the solution of $1 - r = r^2$. We are assuming P_1, P_2, P_3 as mentioned in the class. We begin with a zero flow and send a flow of 1 via the **s-u-v-t** route, saturating e_3 . The residual flow of the edge triplet $[e_1, e_2, e_3]$ now becomes $[a_0, a_1, 0]$.

Now, let the residual capacities of these edges at a later stage be represented as $[a_n, a_{n+1}, 0]$ (holds for $n = 0$) with the residual capacities of other edges being ≥ 1.

It is easily observed that in any augmenting path in this network that has at least one of $[e_1, e_2, e_3]$ in forward direction, the path with the smallest residual capacity will be one of $[e_1, e_2, e_3]$.

Now, apply augmenting path P_1 (where residual edge is e_2 with a residual capacity of a_{n+1}) on $[a_n, a_{n+1}, 0]$ to get $[a_{n+2}, 0, a_{n+1}]$. Similarly, apply augmenting path P_2 (where residual edge is e_3

with a residual capacity of a_{n+1}) to get $[a_{n+2}, a_{n+1}, 0]$. Again, apply P_1 (where residual edge is e_1 with a residual capacity of a_{n+2}) to get $[0, a_{n+3}, a_{n+2}]$ and finally, apply P_3 (where residual edge is e_3 with a residual capacity of a_{n+2}) to get $[a_{n+2}, a_{n+3}, 0]$.

The net increase in flow after one (P_1, P_2, P_1, P_3) application is $2a_n + 2a_{n+1}$ and we can again apply the same combination of path augmentations on the result.

We know that $a_{n+2} = a_n - a_{n-1}$ with $a_0 = 1, a_1 = r$ gives $a_n = r^n$ as the solution

Thus,using this, we get an infinite sequence of flow with value $1+2(a_2+a_{3+}....)$ which converges to 3 (easy to see). But the maximum flow is 5. Thus, we get a non-terminating network of six nodes in which the flow doesn't even converge to the maximum flow.

3 Question 3:

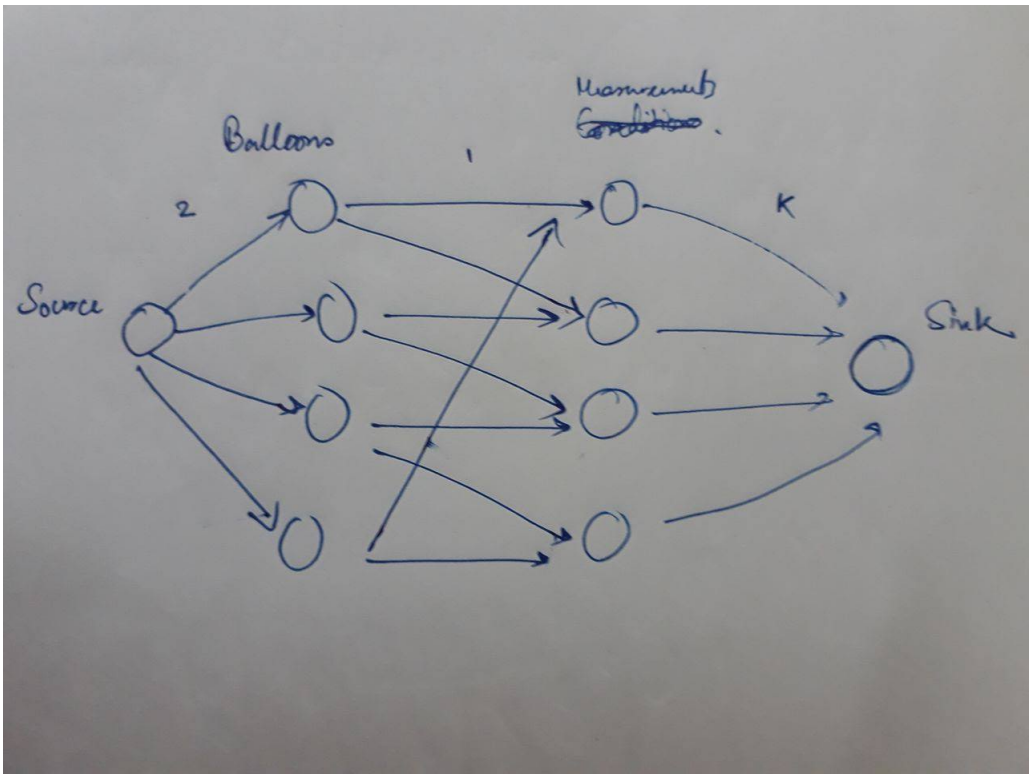
Part 1

We can reproduce this problem as a network. In that case deciding if $\exists$ way to measure each condition k times is same as determining if $\exists$ flow of the graph of value k.

Reason :

The constraints are :

- Every balloon takes at most two measurements.
- Every balloon takes a measurement at most once.
- Every measurement is taken $\geq k$ times.

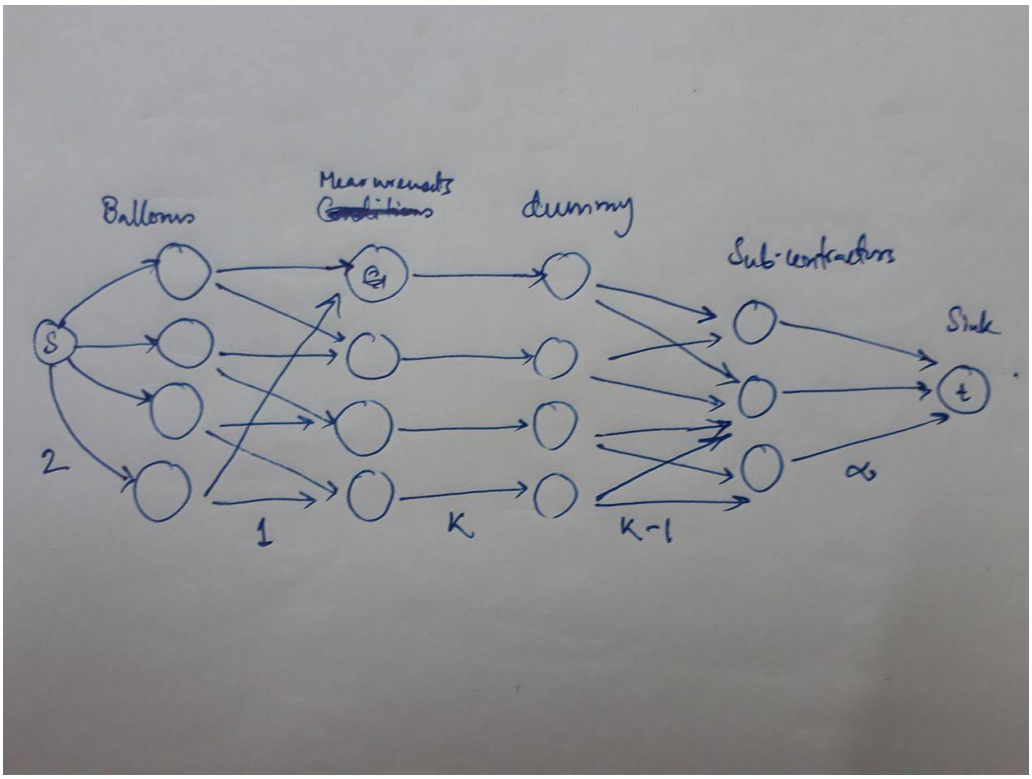


The constraints can be represented as groups of edges. First constraint is represented as, since each balloon will have a max flow of 2 , all single incoming edges will have capacity 2. For the second, since each measurement will be taken at most once by each balloon each outgoing edges from the balloons will have capacity 1. For the last constraint, we can bind the flow through each measurement node by setting capacity k and if under maximum flow each of the k weighted edges are saturated then we can take k observations of each measurement.

Now, for taking k observations for each measurements we want to have a maximum flow, which is achieved by Ford-Fulkerson algorithm in polynomial time.

Part 2

In this case we again transform the problem into a network flow problem by modifying the previous one with one dummy layer extra which is one to one mapped with the measurements layer followed by a sub-contractor layer .



The Constraints are:

- all the constraints from previous part.
- It cannot happen that all k measurements come from balloons produced by single sub-contractor.

  The previous constraints are implemented the same way except we make the outbound edges from the measurements to the dummy layer of capacity k, as one to one maps. Thus the fact that each measurement can be observed at most k times is kept. Now, to encode the fourth constraint we consider the edges from the dummy nodes to sub-contractor nodes. We give

capacity k-1 to each edge. This takes care of the fact that no single subcontractor can take k observations alone for each measurement. Subcontractor nodes are connected to the sink with infinite capacity. We want to find the max flow in the network to find our answer satisfying the constraints.

This can bbe achieved by Ford-Fulkerson algorithm in polynomial time.
