Problem session 6

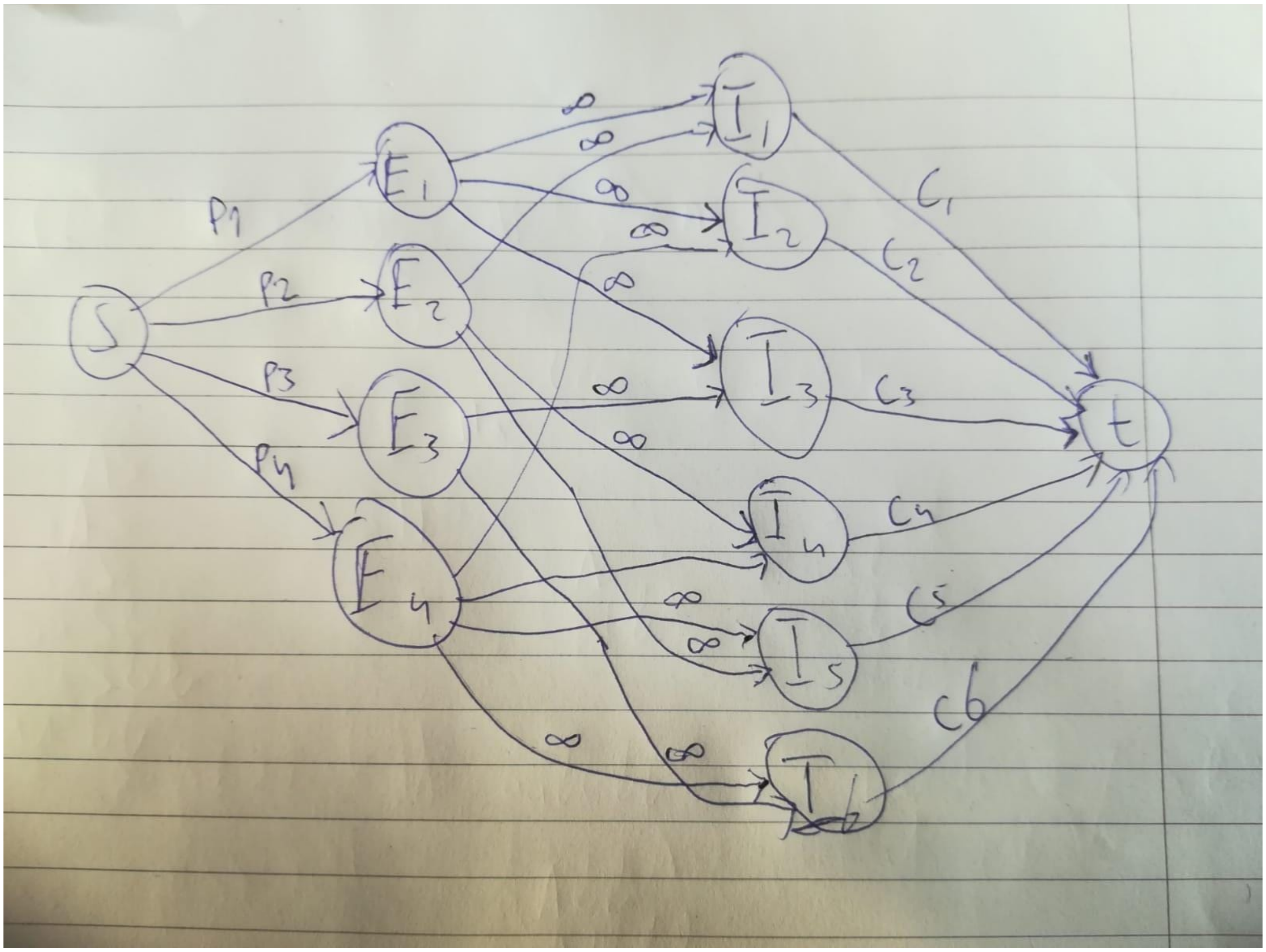
Dibran Dokter 1047390 & Marnix Lukasse 1047400

October 10, 2019

6

6.1

1)



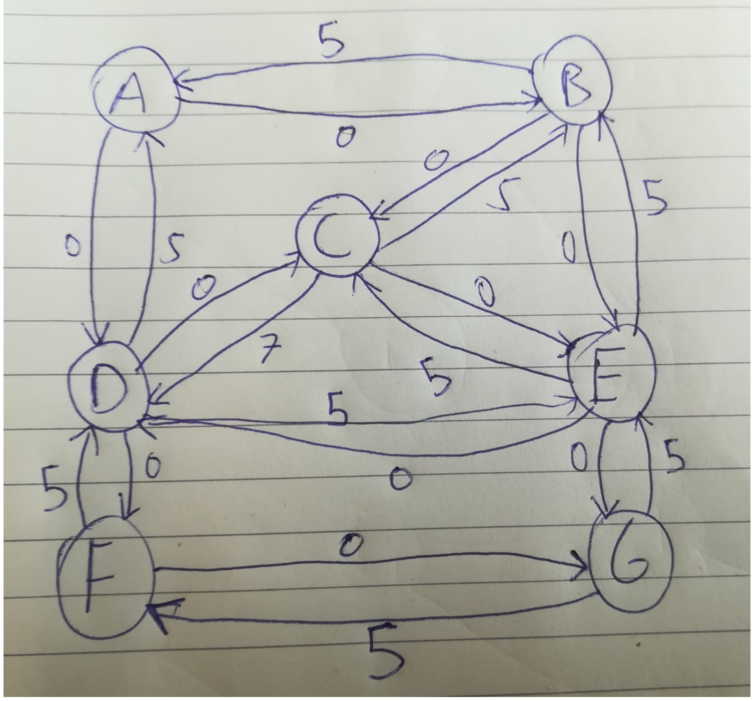
2)

To find the best combination of experiments we can look at the difference between the cost and the profit for every combination of the experiments.

For example, for combination E_1 and E_2 we sum the total payout, look what tools are needed and sum the total cost. Profit = sum of payouts - sum of costs Whichever combination gives the highest profit is the best solution.
This gives a time complexity of $(O)(n!)$, there surely must be better ways to approach this. We weren't able to come up with any though.

The total profits can be calculated by adding up the profits and subtracting the costs. The profits is the total flow from s to the E_i edges and the cost is the total flow from I_i to t.

6.2



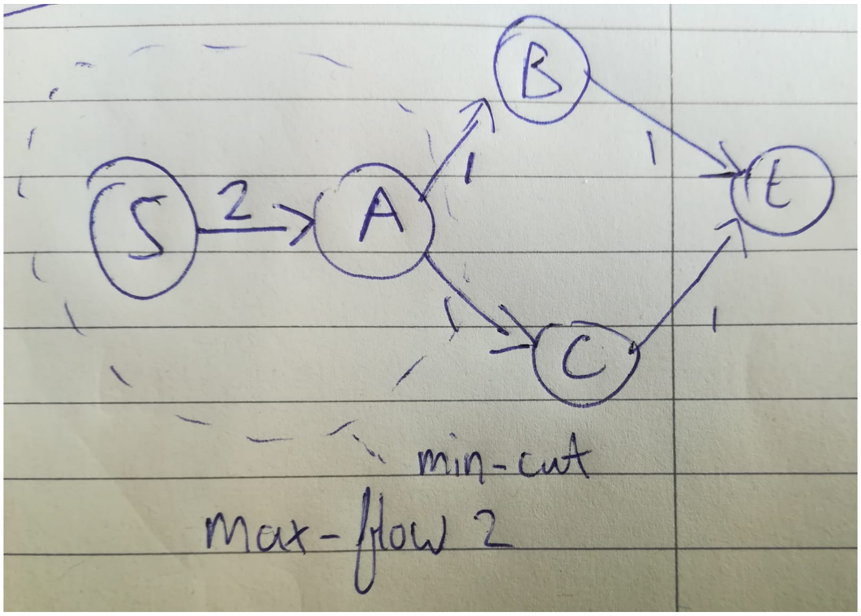
6.3

To check whether its possible to connect each client to a base, we do the following:

For each client, check to which possible bases they can connect according to the range limitation. Store the results in RESULT_TABLE($\mathcal{O}(M^N)$).
Make a graph, with a starting vertex s, and a sink t. Let clients and bases also be vertices. Connect s to all of the clients, and let the cost be 1. Connect all bases to sink t, and let the cost be the max load.
Now connect clients to bases based on possible connections according to RESULT_TABLE, with a cost of 1.
Now run a max-flow algorithm like shortest-augmenting-path first ($\mathcal{O}(M^3)$). If the max-flow turns out to be equal to the number of clients, we can conclude that the answer to our main question is yes. If its lower, we can conclude the answer is no.

6.4

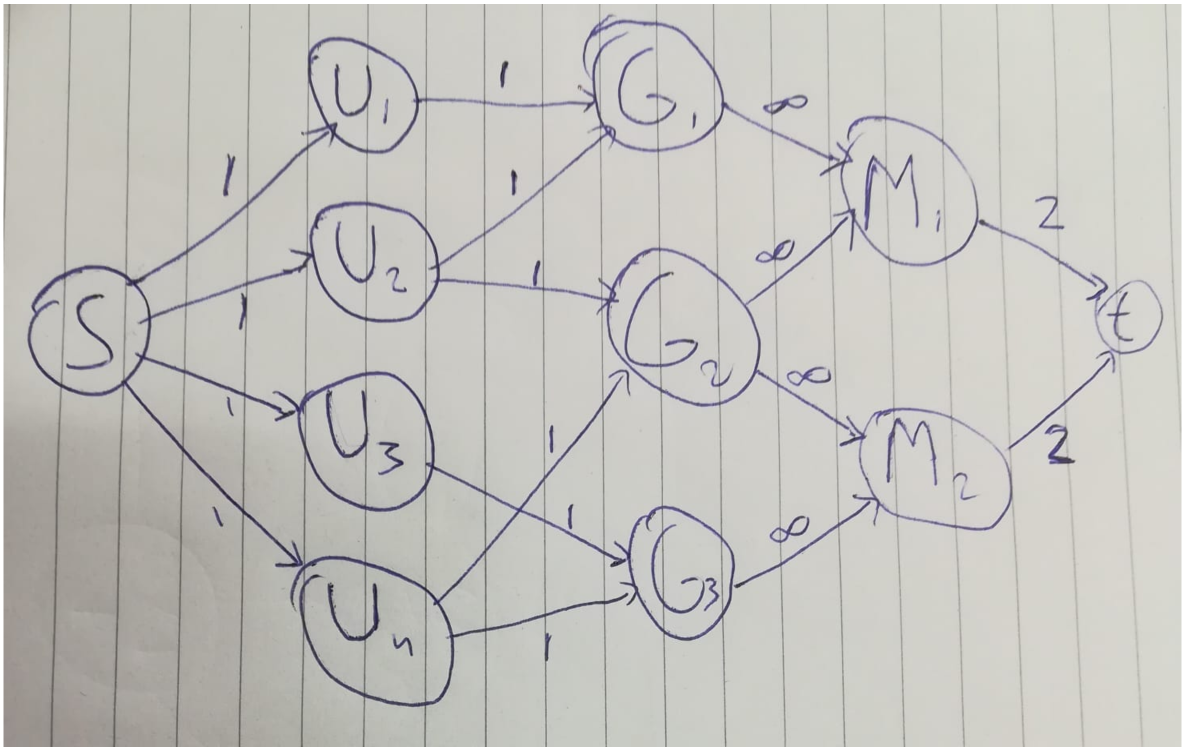
False. In the following case it does not hold:



At first the min-cut is from S->A. When we add 1 to all the edges S->A is no longer a min-cut. Thus displacing the min-cut.

6.5

Example:



The figure contains an example graph, the exact process for making the graph is as follows:

- Draw a source vertex s and a sink vertex t.
- Add all the current users as a vertex U_i
- Add all the different demographic groups as a vertex G_i
- Add all the different advertisers as a vertex M_i
- Let there be edges with cost 1 from s to each U_i
- Let there be edges from each U_i to each G_j if the user is part of that demographic group, let the cost be 1.
- Let there be edges from each G_j to each M_k if the group G_j is part of the audience of advertiser M_k, let the cost be infinite.
- Let there be edges from each M_k to the sink t, the cost here should be the demand of the advertiser M_k. (If M_k wants to advertise to 10 people per minute, it should be 10).

When we have the graph, the only thing we need to do is run a max-flow algorithm. Then we need to compare the max-flow to the sum of all the edges from M -> t, as those represent the demands of the different advertisers. If the max-flow is equal to that sum, then we can satisfy the demands of the advertisers. If no such flow exists, we won't be able to.