Theory Assignment 3

CSE222: *Algorithm Design & Analysis*

Divyajeet Singh (2021529) Agamdeep Singh (2021306)

April 27, 2023

Problem

The towns and villages of the Island of Sunland are connected by an extensive rail network. Doweltown is the capital of Sunland. Due to a deadly contagious disease, a few casualties have been reported in the village of Tinkmoth. To prevent the disease from spreading to Doweltown, the Ministry of Railway of Sunland wants to completely cut down the rail communication between Tinkmoth and Doweltown.
For this, they wanted to put traffic blocks between pairs of stations, x and y, that are directly connected by a railway line (i.e., there are no stations between them on that line). If a traffic block is put in between x and y, then no train can travel from x to y from that line. To minimize expense (and public notice), the authority wants to put as few traffic blocks as possible.
Formulate the above as a flow-network problem and design a polynomial-time algorithm to solve it. Give a precise justification of the running time of your algorithm.

Input

An graph $G = (V, E)$ representing the layout[1] of the island, along with v_t and $v_d \in V$, the railway stations of Tinkmoth and Doweltown.
V is the set of all railway stations and E is the set of all railway lines in Sunland.

Note: *Formulation of the problem as a flow-network problem is given in the subsequent subsection.*

Output

The minimum number of traffic blocks to be installed to cut down the rail communication between Tinkmoth and Doweltown.

Solution

The given problem can be solved using the concept of minimum cut in a flow-network. A minimum cut in a flow-network is a minimal set of edges whose removal disconnects the source vertex from the sink. Since we are required to find the minimum number of traffic blocks required to disconnect the railway network between Tinkmoth and Doweltown, we can find the number of edges in a minimum cut of G. The following sections discuss the apporach to solve the problem in detail.

[1] We assume that Tinkmoth and Doweltown have two distinct railway stations, $v_t, v_d \in V$.

Notations Used

Notation 1 (E_t) *The set of all edges directed towards the source vertex v_t. Formally,*

$$E_t = \{(u \to v_t) \in E \mid u \in V \setminus \{v_t\}\} \tag{1}$$

Notation 2 (E_d) *The set of all edges directed away from the sink vertex v_d. Formally,*

$$E_d = \{(v_d \to w) \in E \mid w \in V \setminus \{v_d\}\} \tag{2}$$

Notation 3 ($C(E_s)$) *The capacity of the cut E_s, i.e., the sum of the capacities of all edges in E_s.*

$$C(E_s) = \sum_{(u \to v) \in E_s} c(u \to v) \tag{3}$$

Notation 4 ($f(u \to v)$) *The flow of the edge $(u \to v) \in E$.*

Formulation as a flow-network problem

Since this problem deals with the railway network of a city, it can be naturally solved using graphs. Each station on the Island can be considered as a vertex $v \in V$. The railway lines connecting pairs of stations u and v, can be considered as two directed edges $(u \to v) \in E$ and $(v \to u) \in E$.
Since the Ministry of Railway intends to stop the spread of infection from Tinkmoth to Doweltown, v_t and v_d can be considered as the source and the sink respectively. Finally, the problem can be converted to a flow-network by assigning a capacity $c(u \to v)$ to each edge $(u \to v) \in E$ as follows:

$$c(u \to v) = \begin{cases} 0 & \text{if } (u \to v) \in E_t \cup E_d \\ 1 & \text{otherwise} \end{cases} \tag{4}$$

The capacity of all edges in E_t is 0 to indicate that v_t is the source and no flow can enter it. Similarly, the capacity of edges in E_d is 0, as v_d is the sink and no flow can exit it. The capacity of all other edges is 1 as we can cut down communication between two stations using only one traffic block.
In this manner, we create a flow-network $G = (V, E)$ with v_t as the source and v_d as the sink, based on the island's railway network.

Figure (1) shows a rough diagram depicting the flow-network[2] G for the Island of Sunland. Here, the red edges have a capacity of 0 and the rest have a capacity of 1. This construction ensures that in the beginning, no flow can enter the source v_t and no flow can exit the sink v_d.

Note: *Since flow is always calculated using simple paths, the maximum flow is actually independent of the initial capacities of the red edges. This is because including them in any $v_t - v_d$ path would produce a cycle. However, for the sake of argument, setting their capacities to 0 ensures that v_t and v_d behave as the source and the sink, even visibly.*

[2] v_t and v_d represent the railway stations for Tinkmoth and Doweltown respectively.

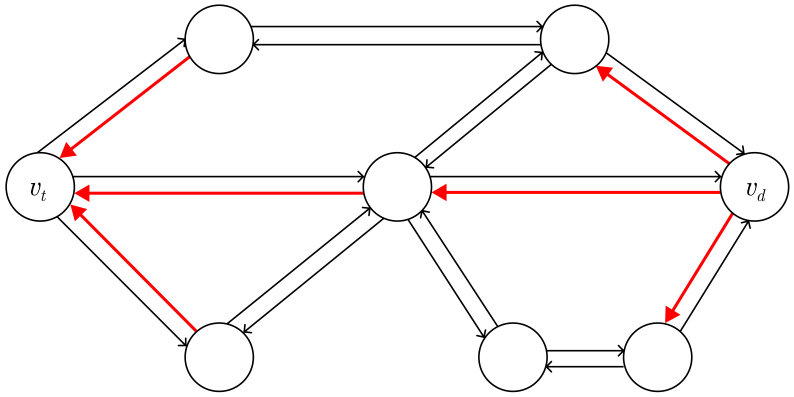

Figure 1: An example flow-network G for the Island of Sunland.

The Solution Description

The problem can be solved by using the Ford-Fulkerson Algorithm[3] to find the maximum flow of the flow-network. In our case, the maximum flow of the flow-network gives the maximum number of *paths* through which the disease can spread from the source to the sink.

Claim 1 *The maximum flow f^* of the flow-network G is equal n^*, to the minimum number of traffic blocks to be installed to cut down the rail communication between Tinkmoth and Doweltown.*

Proof.

> By assignment (4), f^* is equal to the number of simple paths from v_t to v_d that can be traversed *simultaneously* in G. This is because any edge $(u \to v)$ with $f(u \to v) = 1$ is already at its bottle-neck, and cannot be used to traverse any other path.
> Each traffic block blocks one path, as an already used/blocked line cannot be involved in another path. So, there are exactly n^* simple paths from v_t to v_d in G.
> Hence, $f^* = n^*$. □

However, the Ford-Fulkerson Algorithm runs in pseudo-polynomial time, as it is a multiplicative of the maximum flow of the network, which can be arbitrarily large (exponential in the input size). Hence, to solve the given problem in polynomial time, we make the following (very crucial) claim.

[3]Ford-Fulkerson Algorithm was discussed in Lecture-17 of the course **CSE222** by Dr Debarka Sengupta, Winter 2023.

Claim 2 *The maximum flow of a flow-network is linear in the number of edges when the capacities are assigned as described in (4).*

Proof.

Let $G = (V, E)$ be a flow-network with a source v_t and a sink v_d, with maximum flow f^*. Assume that capacities assigned to all edges in E as described in (4).

Any minimum cut $M^* = (V_t, V_d)$ is a partition of the vertices, or a subset of the edges in G. Then $C(M^*) \leq |E|$.
This is because, by construction, all edges $(u \to v)$ with $u \in V_t$ and $v \in V_d$ have a capacity of 1. So, $C(M^*) = O(|E|)$. By the *Max-Cut Min-Flow* Theorem[4], the maximum flow of a flow-network is equal to the capacity of its minimum cut, i.e., $f^* = C(M^*)$.

Hence, $f^* = O(|E|)$. □

The above claim is key to solving the problem in polynomial time. The Ford-Fulkerson Algorithm runs in polynomial time when the maximum flow is polynomial in the number of edges.

The complete Algorithm and Psuedocode

The complete algorithm involves the following steps:

1. Assign capacities to each edge as described in (4).
2. Apply the Ford-Fulkerson Algorithm on the flow-network with v_t as the source and v_d as the sink.
3. Return the value of the maximum flow.

The pseudocode for the complete algorithm is given in Algorithm (1).

[4]Max-Cut Min-Flow theorem was discussed in Lecture-17 of the course **CSE222** by Dr Debarka Sengupta, Winter 2023.

Algorithm 1 An algorithm to find the minimum number of traffic blocks required to cut down the rail communication between Tinkmoth and Doweltown

```
1: procedure MIN-TRAFFIC-BLOCKS(G = (V, E), v_t, v_d)
2:     for (u → v) ∈ E do
3:         if v = v_t or u = v_d then
4:             c(u → v) ← 0
5:         else
6:             c(u → v) ← 1
7:         end if
8:     end for
9:     f* ← FORD-FULKERSON(G, v_t, v_d)
10:    return f*
11: end procedure
```

Runtime Analysis of the Algorithm

The algorithm assigns capacities to each edge in $\Theta(|E|)$ steps. It then makes use of the Ford-Fulkerson algorithm, which takes $O(|E|f^*)$ time to find the maximum flow of the flow-network[5].
However, claim (2) proves that $f^* = O(|E|)$. Therefore, in the above solution, Ford-Fulkerson algorithm takes $O(|E|^2)$ time.

Hence, dominated by the runtime of the Ford-Fulkerson algorithm, the time complexity of the algorithm is $O(|E|^2)$.

[5]Here, f^* denotes the maximum flow of the flow-network G.