

# Networks and Markets

Hw2 submission

Part 5: Experimental Evaluations

Alon Polski 206530461

Anna Petrenko 320460306

Ariel Chiskis 322442112

9.

An important comment:

It should take the code of question 9 around 2 hours to run.

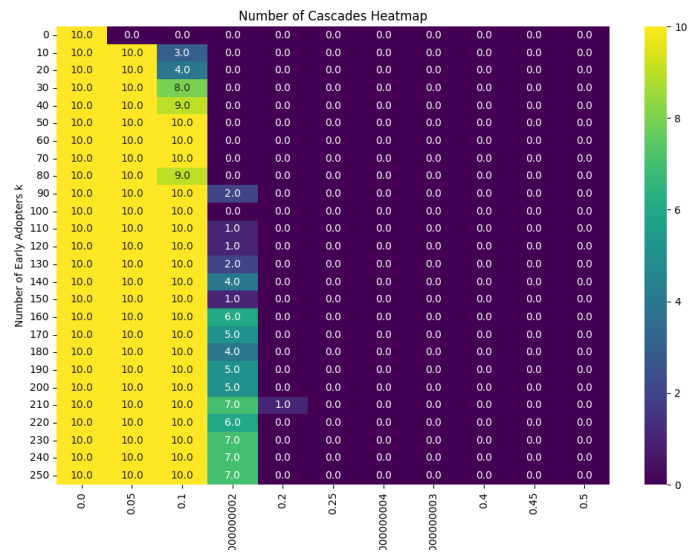
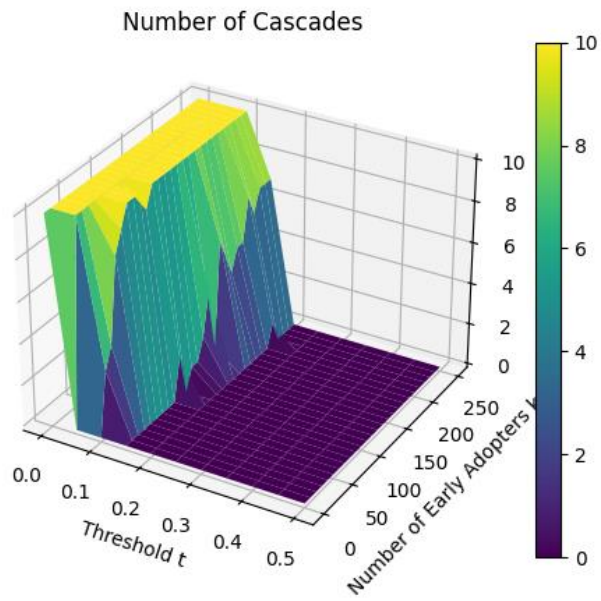
(b) We get a complete cascade in 28 of the BRD runs.

On average, there are 3438.23 nodes infected.

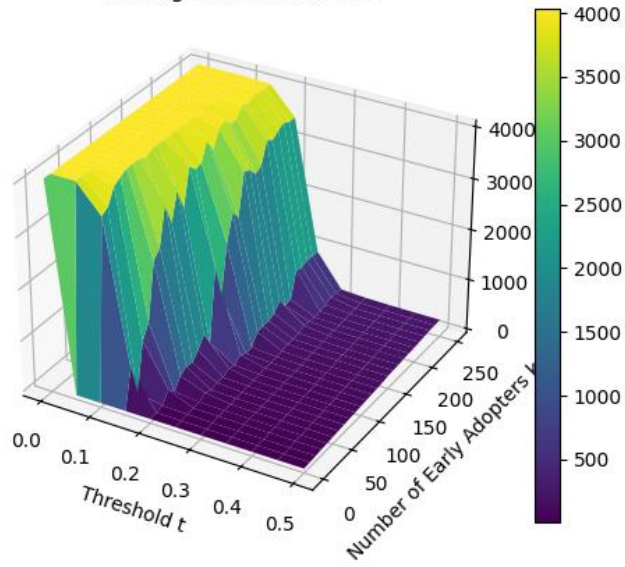
This means that there are some  $1 - t = 0.9$  dense subsets of nodes in the graph and there's a non-neglectable probability for a node to be a member of such set, it also means there are not a lot of disjoint sets like this (less than 10 at least.), because otherwise it would have been rare to infect all of them.

Also, we have a probability that is around  $\frac{28}{100}$  to hit nodes in all of them if you sample 10 nodes randomly.

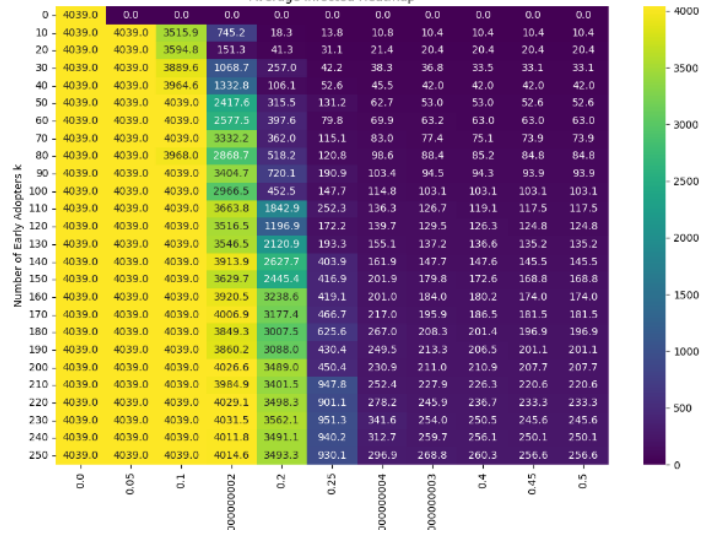
(c) The maximal value of  $t$  that seems to cause a complete cascade is around 0.2, but for such a high  $t$  the probability of a complete cascade is relatively low (around 0.1), we observe probability that is around 1 for a complete cascades for maximal  $t$ - around 0.1 where  $k$  is at least 50, and probability higher than  $\frac{1}{2}$  for a complete cascade for maximal  $t$  around 0.18, where  $k$  is at least 160. However small is  $k$  we can still see complete cascade if  $t$  is small enough but we start seeing cascades even for  $k = 10$  where  $t$  is a most- around 0.1, for that  $k$  complete cascade can get to around 1, when  $t$  is at most- around 0.05, for that  $k$  the probability of a complete can get to higher than  $\frac{1}{2}$  where  $t$  is at most somewhere in  $[0.05, 1)$ . As  $t$  decreases the probability for a complete cascade and the average size of a cascade seem to grow exponentially, as  $k$  increases the probability for a complete cascade as the average size of a cascade seem to grow linearly.



Average Infected Nodes



Average Infected Heatmap



10.

(b)

i. Let

$$G := (V, c)$$

be a capacitated graph, where:

$$V := \{s, t\} \cup \{v_i\}_{i=1}^n \cup \{u_i\}_{i=1}^m$$

$$c(x, y) := \begin{cases} 1, & (x = s) \wedge (y \in \{v_i\}_{i=1}^n) \\ 1, & (x \in \{u_i\}_{i=1}^m) \wedge (y = t) \\ 1, & (\exists i \in [n]. v_i = x) \wedge (\exists j \in [m]. u_j = y) \wedge (\text{driver } i \text{ can pickup rider } j) \\ 0, & \text{else} \end{cases}$$

for every matching  $M$ , we can define the following flow in  $G$ :

$$f(x, y) := \begin{cases} 1, & (x = s) \wedge (\exists i \in [n]. v_i = y) \wedge (\text{driver } i \text{ is matched}) \\ 1, & (\exists i \in [m]. u_i = x) \wedge (y = t) \wedge (\text{rider } i \text{ is matched}) \\ 1, & (\exists i \in [n]. x = v_i) \wedge (\exists j \in [m]. y = u_j) \wedge (\text{driver } i \text{ is matched to rider } j) \\ 0, & (x = s) \wedge ((\exists i \in [n]. y = v_i) \wedge (\text{driver } i \text{ is unmatched})) \vee (y \in \{u_i\}_{i=1}^m) \vee (y = t) \vee (y = s) \\ 0, & (\exists i \in [n]. v_i = x) \wedge ((\exists j \in [m]. u_j = y) \wedge (\text{driver } i \text{ isn't matched to rider } j)) \vee (y \in \{v_j\}_{j=1}^n) \vee (y = t) \\ 0, & (\exists i \in [m]. x = u_i) \wedge ((y \in \{u_j\}_{j=1}^m) \vee ((y = t) \wedge (\text{rider } i \text{ unmatched}))) \\ 0, & x = y = t \\ -f(y, x), & \text{else} \end{cases}$$

And that's easy to see that  $|f| = |M|$ .

Also, let  $f'$  be an integer max flow in the graph, then we can define a matching  $M'$  by

$$(*) M' := \{(i, j) : f(v_i, u_j) = 1\}$$

$M$  is a matching because there's only one edge that goes into each  $v_i$  and its capacity is 1, and therefore, there can't be more than one  $u_j$  s.t  $f(v_i, u_j) = 1$ .

We've seen that for every matching  $M$ , there's a flow  $f$ , s.t.  $|M| = |f|$ , and that there's a matching  $M'$  for the integer max-flow  $f'$ , s.t.  $|f'| = |M|$ , and therefore if there's an integer max-flow, its value is the maximal number of matches.

However, as a corollary from the Augmenting Path algorithm there's an integer max-flow, and therefore we can run on  $G$  the augmenting path

algorithm to find an integer max-flow and output the maximal flow's value as the maximum number of matches.

ii. In part i, we've seen how to define a maximal matching from an integer max-flow in  $G$  (definition (\*)), we can find that integer max-flow by running the augmenting path algorithm on  $G$  and then use definition (\*) to find the maximal matching.

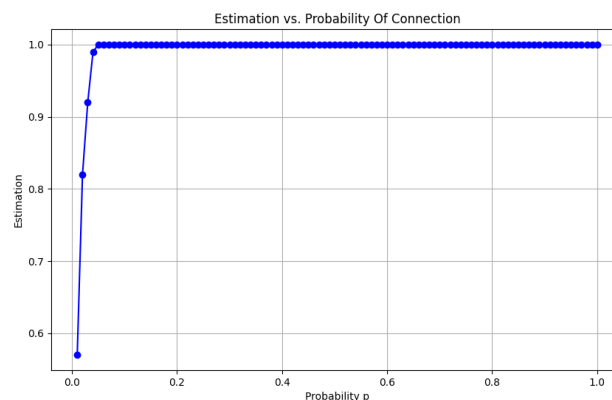
(c) In our first example, there were five drivers and five riders, driver 0 could only be matched to rider 1, driver 1 could only be matched to riders 0, 1, driver 2 could only be matched to rider 0, driver 3 could only be matched to riders 2 and 4, driver 4 could only be matched to riders 2, 3.

In that example we found out that the size of the maximal matching is four and an instance of such matching is the instance where driver 0 is matched to rider 1, driver 1 is matched to rider 0, driver 3 is matched to rider 2, rider 3 is matched to rider 3 and driver 3 and rider 4 are left unmatched.

In our second example, there were four drivers and four riders, where driver  $i$  could be matched to riders  $i$  and  $i + 1$ , except the last driver that could only be matched to rider 3.

In that example we found that the size of the maximal matching is 4 and that in that matching driver  $i$  is matched to rider  $i$

(d) We fixed  $n = 100$  because  $n = 1000$  was too much. We got the following results:



to estimate the probability, we computed a case of  $n$  drivers and  $n$  riders where each driver could be connected to each rider with probability  $p$ , then we fixed the connections according to that  $p$  to 0 or 1, and divided the size of the maximal matching by  $p$ .