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 two and a half 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 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 lease.), because otherwise it would have been rare to infect all of them.

Also, we have a probability that is around to hit nodes in all of them if you sample 10 nodes randomly.

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

תמונה שמכילה צילום מסך, מלבן, תרשים, עיצוב

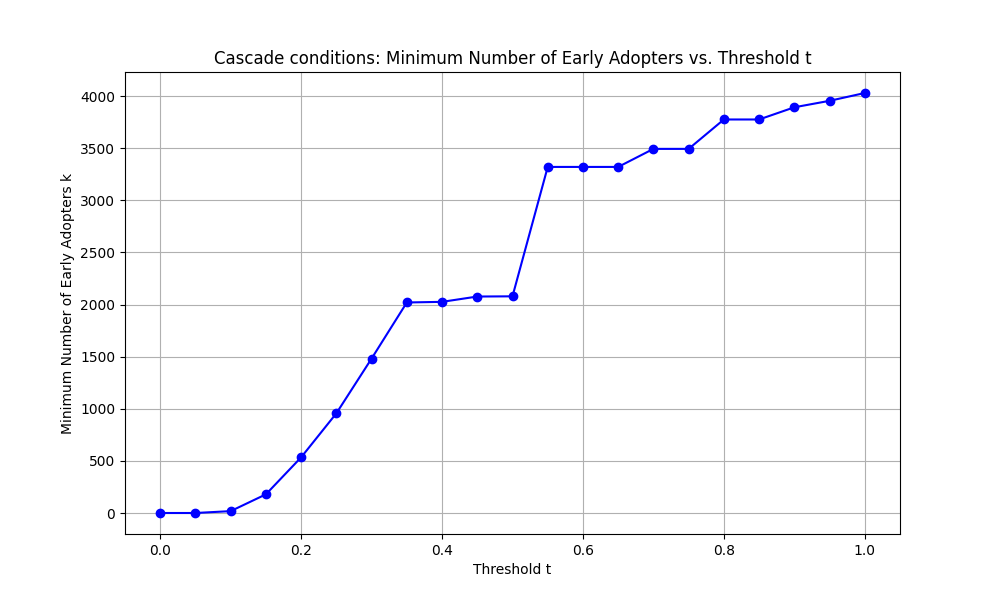
התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, צילום מסך, צבעוני, מלבן

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, צילום מסך, תרשים, עיצוב

התיאור נוצר באופן אוטומטיתמונה שמכילה טקסט, צילום מסך, צבעוני, מלבן

התיאור נוצר באופן אוטומטי

(d) We got the following graph:



We thought about adding on top of a binary search on , a clustering algorithm to find dense disjoint sets, , and then for each set find the node that reaches the minimum of the function:

and then build as the set of those nodes, but unfortunately, we did not have time to implement that and decided to just implement a binary search on .

10.

(b)  
i. Let

be a capacitated graph, where:

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

And that's easy to see that .

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

is a matching because there's only one edge that goes into each and its capacity is , and therefore, there can't be more than one s.t   
.

We've seen that for every matching , there's a flow , s.t. , and that there's a matching for the integer max-flow , s.t.   
, 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 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 (definition ), we can find that integer max-flow by running the augmenting path algorithm on and then use definition to find the maximal matching.

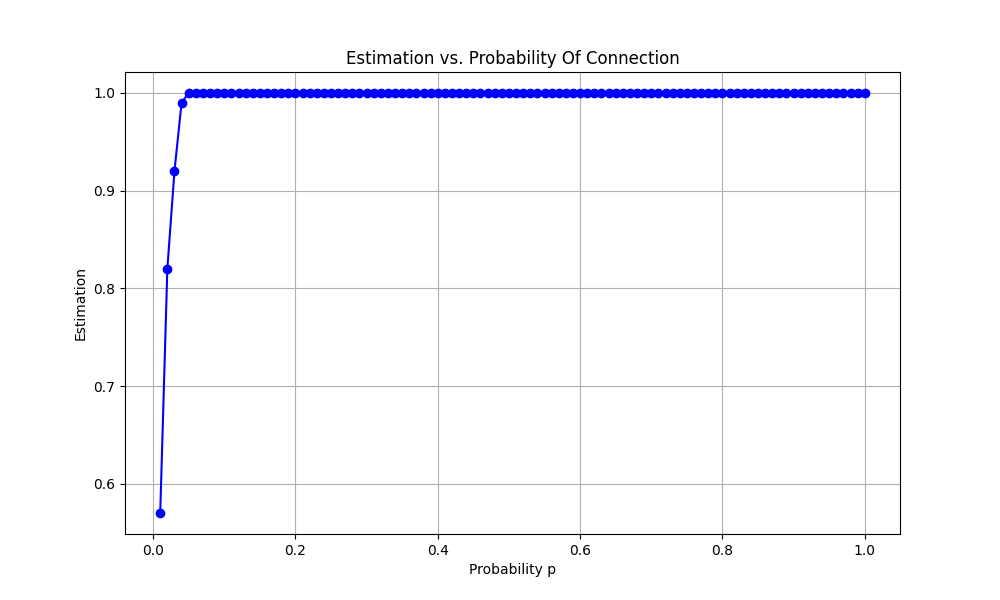
(c) In our first example, there were five drivers and five riders, driver 0 could only be matched to rider 1, driver 1 could only be matched to riders 0, 1, driver 2 could only be matched to rider 0, driver 3 could only be matched to riders 2 and 4, driver 4 could only be matched to riders 2, 3.

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 could be matched to riders and , except the last driver that could only be matched to rider .

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

(d) We fixed because was too much. We got the following results:



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