Zachary Cain & Bryan Oswald

Sukumar Ghosh

P2P and Social Networks

December 8, 2015

Final Project Report

1. Introduction

For our project we studied Erdös-Rényi graphs with sizes n = 500, 1000, 2000, 5000 and with probability ranging from 0.0001 to 0.05. With these graphs there was a variety of things we wanted to test out, to start we wanted to look at the graphs and see if there is a single connected component, or a single giant component or multiple small trees. Next we wanted to look into the diameters for the graphs generated (given the graph is connected) and compare the results we got to theoretical results. Next we wanted to take these graphs and have a population of target nodes ranging from 0.001 to 0.01. Then with these target nodes in place we wanted to run different search algorithms on them, seeing how each algorithm did in terms of total nodes visited and time required before each of the algorithms found the target node. The three algorithms we decided to test were: gnutella-type flooding, random walker model and k-random walker model (with one-hop replication).

1. Our Approach and Results

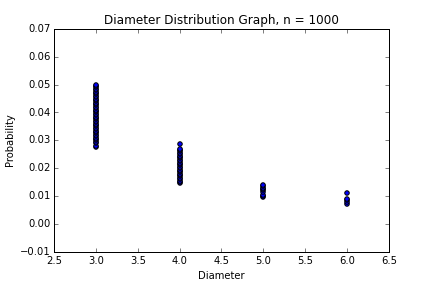
For this project we decided to use networkx to generate our graphs and matplotlib to chart them. We planned on using these charts to help visualize how our diameters changed with different probabilities and then compare charts with different values of n to see how the diameters changed at the same probability values. Then we also wanted to compare these with theoretical results, which state that the diameter should be within the range: MAX(log(n) / log(n\*p)) -4 to MAX(log(n) / log(n\*p)) +1 (n = number of nodes and p = probability). For this we went through a loop with around 100 intervals making Erdös-Rényi graphs at each increasing probability interval, then given the graph was connected we found the diameter and charted it. Running our code gave us the following results (Figure 1) and match up accordingly with the theoretical results (Table 1).

Figure 1 and Table 1

|  |  |  |
| --- | --- | --- |
| Probability | Expected Diameter | Our Diameter |
| 0.04 | -2 to 3 | 3 |
| 0.02 | -1 to 4 | 4 |
| 0.009 | 0 to 5 | 5 |

As shown the diameters match up well with theoretical results, however the diameters always were on the higher end of the expected diameter (even at n=5000). This result was interesting, however still within the expected range of diameters.

1. The Search Algorithms

To start this project we began by writing the code needed for the random walker. So for random walker we began by selecting a random node and checking if it was the target node, if not then we set this to be a visited node and randomly picked one of its neighbors that have not been visited yet. This process was repeated until a target node was found or until there was no possible next step. If the target node was found then the time and nodes visited were returned, if there was no target node found, then 0’s were returned for both.

-TALK ABOUT RESULTS FOR RANDOM WALKER HERE

Moving onto the k-random walker we selected random start node(s) depending on the number of k (five was used). Then we went over all of these checking to see if each one was the target node and marking it as visited. Then we looked at each nodes neighbors to see if any of them were target nodes (one-hop replication). If not, then we went to one of the neighbors that wasn’t marked as visited and repeated this process until a target node was found or there were no more possible steps. If a target node was found we returned the number of nodes visited, the time it took and the number of hops (loops) that were needed to find the target node. However if it failed to find the target node then zeros were returned for all three.

-TALK ABOUT RESULTS FOR K-RANDOM WALKER HERE

Finally for the gnutella-type flooding we selected a random start node then check to see if this node is the target node. If not then we add all the neighbors into a list and remove the node we just looked at from that list. Then this process is repeated for the next node in the list until a target node is found or the time-to-live is met or there are no more possible next steps. If a target node is found then the nodes visited and the time are returned. However if a target node is not found, then both of these are returned as zeros.

-TALK ABOUT RESULTS FOR GNUTELLA-TYPE FLOODING HERE

1. Evaluations of the Search Algorithms

COMPARE RESULTS HERE

1. Conclusions and Further Work

SAY WHICH ONE IS THE BEST OR WHAT IS THE BEST IN EACH CIRCOMESTANCE