SAPIENZA
UNIVERSITÀ DI ROMA

Networking for Big Data and Laboratory Challenge 1

Group name: OHM

Candi	Protani	Tarantino
Matteo	Andrea	Ramona
1884760	1860126	2082006

Connectivity & random graphs

From the plot represented in Figure 1 it can be seen that the best method, in terms of time, to verify if a graph is connected is through the BFS algorithm. In fact, it can be seen how the curve (the time values in seconds have been represented in a logarithmic scale to better visualize the results) relating to this method is constantly lower than the methods based on the irreducibility and the eigenvalues of the Laplacian matrix. With regard to these two methods it is possible to appreciate a particular trend, for N up to ≈ 275, the method based on irreducibility is faster while beyond this threshold the method based on the eigenvalues of the Laplacian matrix becomes preferable.

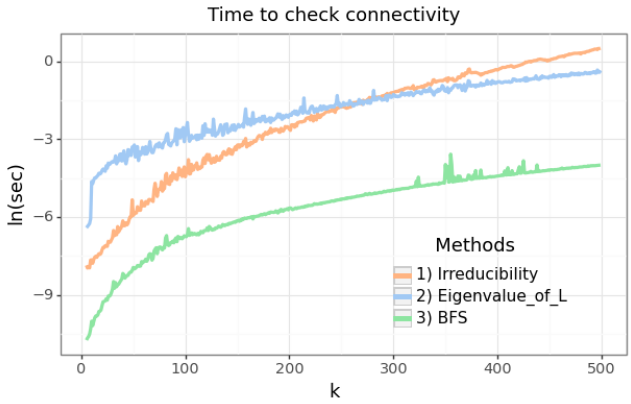

Figure 1: Complexity versus number of nodes

Figure 2 shows the trend of the probability that a graph of 100 nodes generated according to the Erdos-Reiny model is a single component connected to the variation of the probability p (represented in logarithmic scale) that an edge has of being generated between two nodes. We can see that the probability grows very rapidly in $p \in [0.03, 0.1]$ and that beyond this interval the graph will almost certainly be composed of a single connected component.

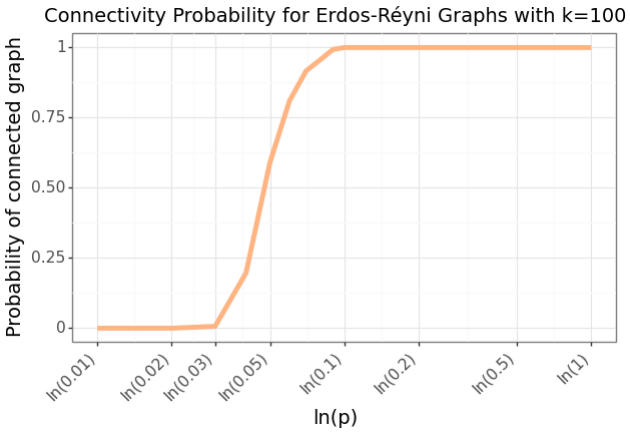

Figure 2: Probability of a connected ER random graph

From the plot in Figure 3 we see that the line for $r = 8$ shows a constant connection probability of 1, while that for $r = 2$ starts from 1 and decreases as k increases. This is because, for $r = 8$, each node is connected to enough nodes to ensure connectivity of the entire graph, while for $r = 2$, many small loops are created so as the number of nodes increases, it becomes increasingly difficult for each node to be connected to enough nodes to ensure connectivity of the graph.

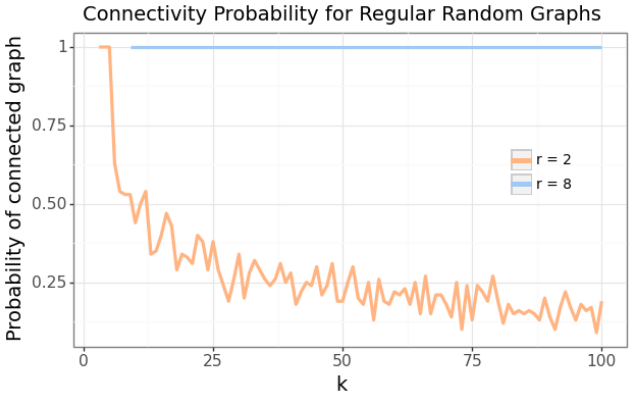

Figure 3: Probability of a connected r-regular random graph

Mean response time & Job running cost

Algorithm 1 Mean response time

Given a value of N, the function returns the Mean Response Time $\mathbb{E}[\mathcal{R}]$ as the average of simulations ran on M random servers of the network that split the job with their N nearest servers.

Require: $Networkgraph$, N, M, C, L_f, L_o, $\mathbb{E}[X]$, f, T_0
Ensure: $\mathbb{E}[\mathcal{R}]$

1: **function** Mean_Response_Time($Networkgraph$, N, M, C, L_f, L_o, $\mathbb{E}[X]$, f, T_0)
2: $R\_sample \leftarrow$ empty vector of size M
3: **for** $m \leftarrow 1$ to M **do**
4: $\mathcal{A} \leftarrow$ randomly select a *server* from the network topology
5: $servers \leftarrow$ select N servers closest to $\mathcal{A}$ based on the number of hops
6: $h_i \leftarrow$ get number of hops of the N closest servers
7: $T_i = 2\tau h_i \leftarrow$ get RTT between each server in *servers* and server $\mathcal{A}$
8: $\theta_i = C\frac{1/T_i}{\sum_{j=1}^{N} 1/T_j} \leftarrow$ average throughput for each server via TCP connection
9: $L_{o,i} \leftarrow$ generate a vector of uniform random variables in the interval $[0, \frac{2\,L_o}{N}]$
10: $X_i \leftarrow$ generate a vector of negative exponential random variables with mean $\frac{\mathbb{E}[X]}{N}$
11: $sending\_time = \frac{(1+f)L_f}{N\theta_i} \leftarrow$ time to transfer data from server $\mathcal{A}$ to each server i
12: $job\_time = T_0 + X_i \leftarrow$ time that each server i requires to compute the job
13: $receiving\_time = \frac{(1+f)L_{o,i}}{\theta_i} \leftarrow$ time to transfer back data from each server i to server $\mathcal{A}$
14: $R\_sample[m] = max(sending\_time + job\_time + receiving\_time) \leftarrow$ take the value of the server which takes the longest time to get, compute and send back data (i.e., the response time)
15: **end for**
16: **return** mean of the R_sample

The plot in Figure 4 shows the *Normalized Mean Response Time* trend for two types of network graphs: Jellyfish and Fat-Tree. The x-axis represents the number of servers used based on the value of the natural logarithm, while the y-axis shows the normalized value of the average response time. Before the first jump at $N = 31$ the trend of the two curves is similar since both networks use servers at the same distance (2). At $N = 31$ the Jellyfish goes from 2 to 3 servers, while the Fat-Tree goes from 2 to 4. This results in a larger $\mathcal{R}$ for the Fat-Tree compared to the Jellyfish. Before the second jump, there is a realignment of the curves: both graphs use all available servers at that distance. At $N = 1023$, the number of servers changes:

the Jellyfish goes from 3 to 4, while the Fat-Tree goes from 4 to 6. This results in a further increase in the average response time. Finally, the plot shows that the best choice of N for the Fat Tree based on average response time is 1023, while for the Jellyfish it is 1047. This means that by using these values, the best possible average response time can be obtained for each topology.

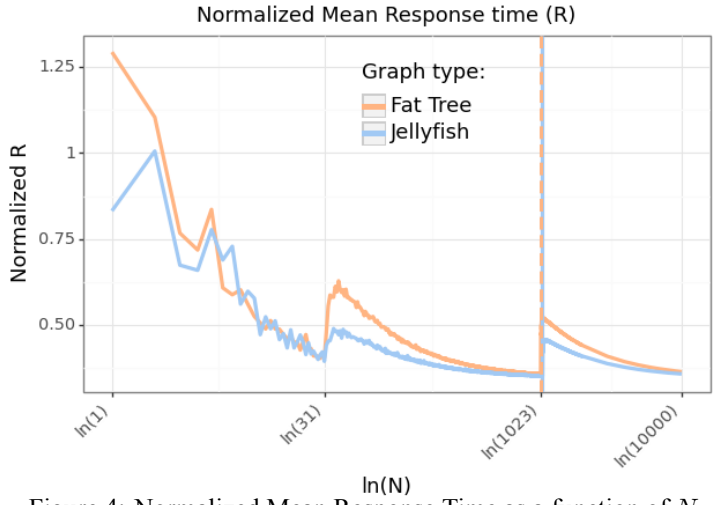

Figure 4: Normalized Mean Response Time as a function of N

The Figure 5 shows the *Normalized Job Running Cost* ($\mathcal{S}$) for the two types of the network as a function of the number of servers used N (represented on a logarithmic scale). To obtain more consistent results, a simulation was performed and the average values of $\mathcal{S}$ were taken. It can be seen that the job running cost is very similar up to $N = 31$ since for both topologies the number of hops (distance) is equal to 2. At this point ($N = 31$) the minimum value of $\mathcal{S}$ for the Fat-Tree is reached, meaning it is the best number of servers to use. At $N = 32$ there is a jump caused by switching to using servers located at the next distance (3 for Jellyfish and 4 for Fat-Tree), from here up to $N = 1023$ (accurate for Fat-Tree while ≈ 1023 for Jellyfish) both graphs have a U-shaped trend and in this area the Jellyfish reaches its minimum at $N = 180$. We can then see a new jump when switching to using server at the next distance (4 for Jellyfish and 6 for Fat-Tree) but from this point on the value of $\mathcal{S}$ tends to grow very rapidly well beyond the baseline value.

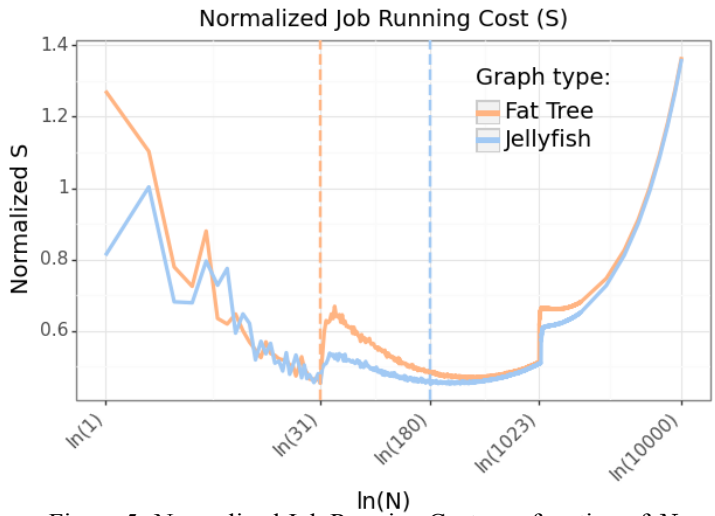

Figure 5: Normalized Job Running Cost as a function of N