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 and random graphs

We see from the plot in Figure 1 that:

- The Irreducibility method is the one that takes the most time because needs to compute always bigger powers of adjacency matrix A.
- Checking if the second-last eigenvalue is greater than 0 increases as the number of nodes increases because is needed to decompose a bigger laplacian matrix L but is less expensive than the previous one.
- BFS is faster because it doesn't directly depend on the number of nodes of the graph but on its connectivity.

For the next point, we use the BFS method to check the connectivity of the graphs.

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 that an edge has of being generated between two nodes. We can see that the probability grows very rapidly in $p \in [0, 0.1]$ and that beyond this interval the graph will almost certainly be composed of a single connected component.

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.

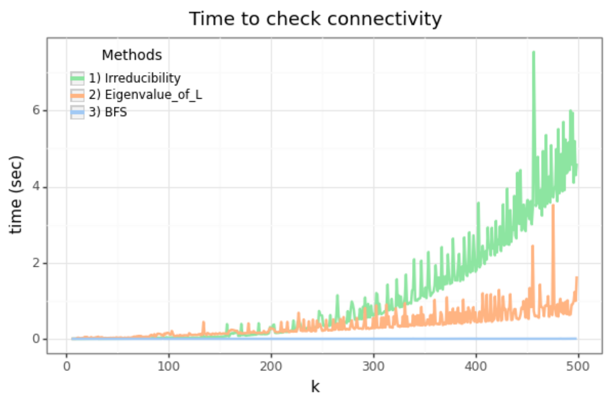

Figure 1: Complexity versus number of nodes

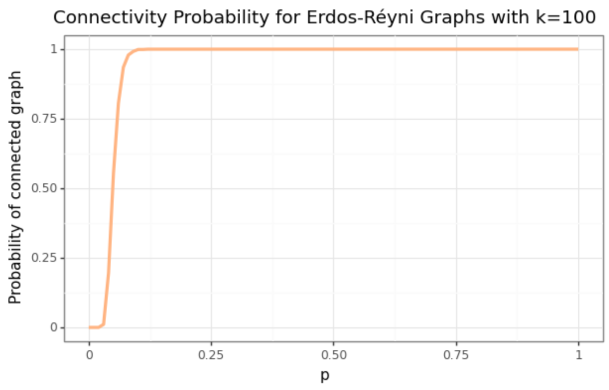

Figure 2: Probability of a connected ER random graph

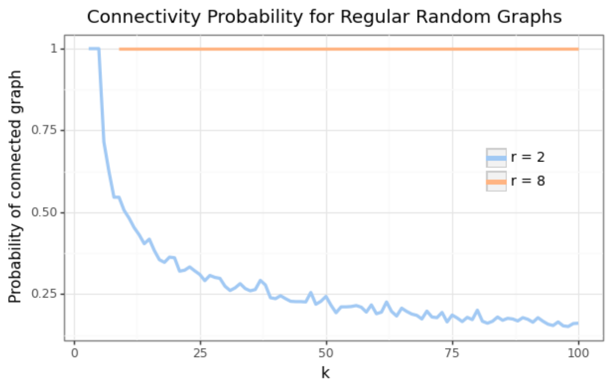

Figure 3: Probability of a connected r-regular random graph

Mean response time

Given a network topology the algorithm for the evaluation of the mean response time $\mathbb{E}[\mathcal{R}]$ is based on the following steps:

1. Select N servers closest to server $\mathcal{A}$ (based on the number of hops) in the given Data Center network topology
2. Split the computation job into N parallel tasks, assigning each job to one of the N servers
3. Calculate the time required to transfer data to and from server $\mathcal{A}$ for each server i, using TCP connection with average throughput θ_i
4. Calculate the amount of data L_i transferred to and from server i, which is equal to $\frac{Lf}{N}$
5. Calculate the time required to run the computation task assigned to each server i, which is a negative exponential random variable with mean $\mathbb{E}[Xi] = \frac{\mathbb{E}[X]}{N}$
6. Calculate the amount of output data $L_{o,i}$ produced by each server i, which is uniformly distributed in the interval $[0, 2\frac{L_o}{N}]$
7. Calculate the completion time $\mathcal{R}_i$ for each server i as the sum of the time to transfer the input data to the server i, time to process the data $(T_0 + X_i)$ and the time required to transfer the output data $L_{o,i}$ back to server $\mathcal{A}$
8. Define the response time $\mathcal{R}$ as the maximum completion time among all servers
9. Repeat steps $2 - 8$ M times to obtain a sample of response times given a value of N
10. Calculate the mean response time by averaging the sample obtained in the previous step

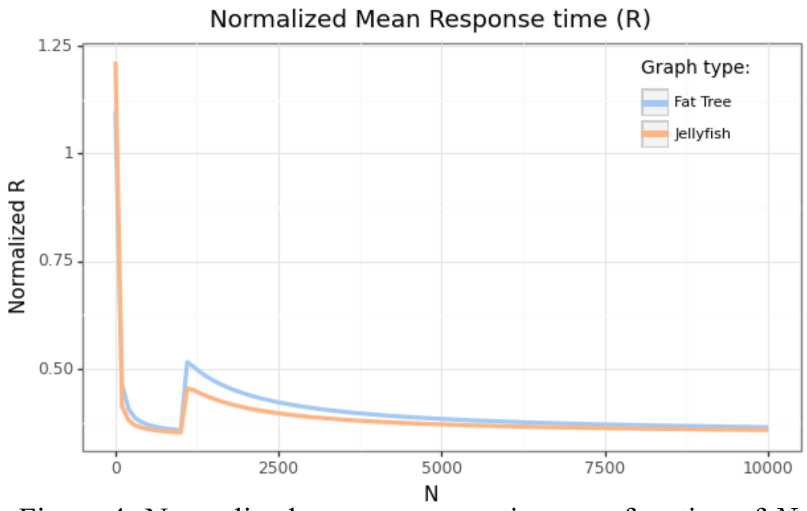

Figure 4: Normalized mean response time as a function of N

From the plot showed in Figure 4 it can be seen that both types of networks, i.e. Fat-tree and Jellyfish, tend to follow the same trend, both start from above the baseline and rapidly decrease until all the servers at the minimum distance are used, i.e. 2, from this point until all servers located at the following distance are used (3 for Jellyfish and 4 for Fat-tree) a slow descent continues ending in what appears to be the global minimum when all servers located at this distance are used distance. We then notice a jump with the increase of the distance and a subsequent very slow decrease of $\mathbb{E}[\mathcal{R}]$ which, however, does not seem to reach the previously found minimum. We can therefore conclude that if we were in the position of having to choose how many servers to use to perform the work we should use all the servers at a distance of 3 in the case of a Jellyfish structure, and 4 in the case of a Fat-tree hop structure, from the starting server.

Job running cost

From Figure 5 it can be seen how the two types of networks behave in the same way also as regards the Job running cost ($\mathcal{S}$). Also in this case we can see how both start from a very high level for $N = 1$ (even higher than the baseline therefore using only server $\mathcal{A}$) and how rapidly the value of $\mathcal{S}$ drops to touch the minimum which this time however corresponds to the use all servers at a minimum distance from the starting server (i.e. 2). There is then a rapid ascent which features even more marked jumps as the distance increases (hops).

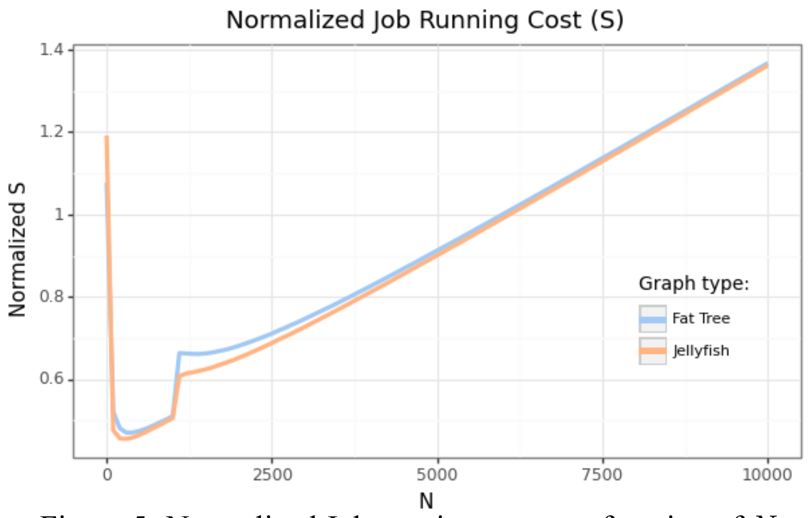

Figure 5: Normalized Job running cost as a function of N