NETWORKING FOR BIG DATA AND LABORATORY

Data Center, Challenge 1

Group Nickname: Wiener

Aur Marina Iuliana, 1809715
Balestrucci Sophia, 1713638
Musacchio Michele, 2070948

A.Y. 2022-2023

1 Part 1

1.1 Complexity versus K of the three algorithms checking connectivity

Figure 1: Complexity of the three algorithms using a p-ER random graph with K in range(3, 1000) and p = 0.2

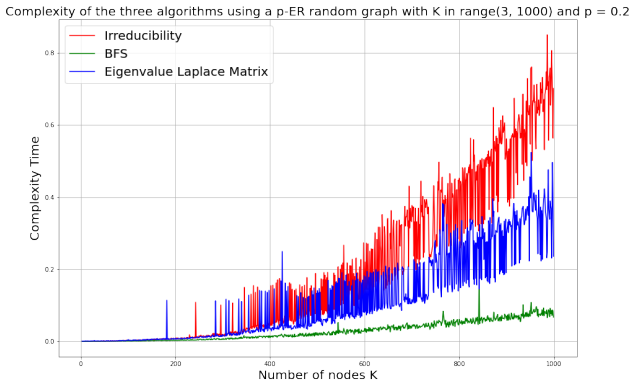
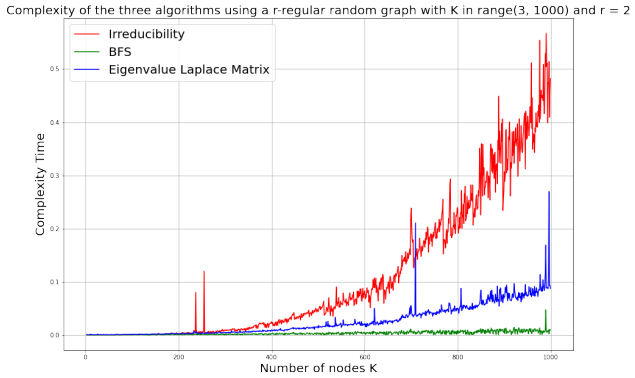


Figure 2: Complexity of the three algorithms using a r-regular random graph with K in range(3, 1000) and r = 2



We can observe that the **Breadth-First search algorithm** (green curve) is **notably more efficient** compared to the other two methods. Its time complexity is $O(K^2)$, whereas the Irreducibility (red curve) and Laplacian Eigvalue Matrix (blue curve) methods have a time complexity of $O(K^3)$.

1.2 Monte Carlo Simulation

Figure 3: Probability of a connected ER random graph as a function of p for K = 100 nodes.

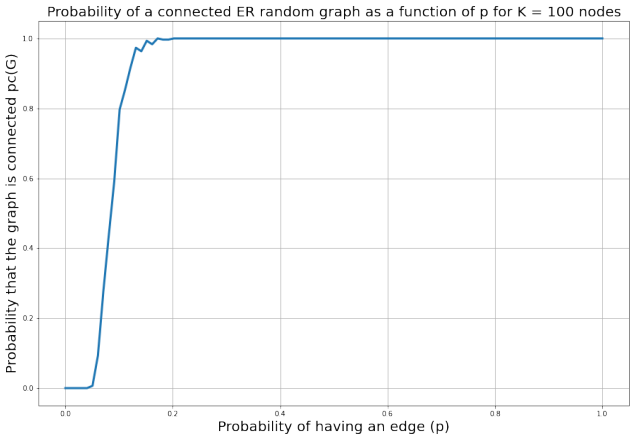
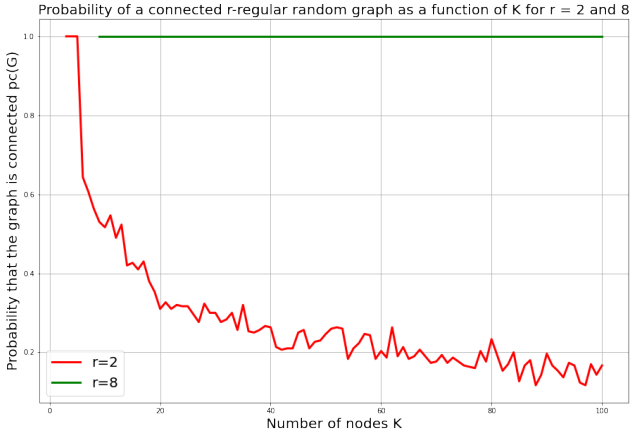


Figure 4: Probability of a connected r-regular random graph as a function of K for r = 2 and 8.



In a **p-ER random graph**, the probability of the graph being connected increases significantly when p is approximately 0.2 or greater (Figure 3).

In an **r-regular graph**, when r = 2, the probability of the graph being connected remains high up to approximately with K = 10. When r = 8, the probability that the graph is connected stays high even as the number of nodes K increases (Figure 4).

2 Part 2

2.1 Evaluation of the Mean Response Time

In order to collect samples of R, we repeat our algorithm multiple times while varying N from 1 to 10000. The purpose of the algorithm is to simulate the behavior of a distributed system with N servers and two different topologies design (FatTree and Jellyfish) and evaluate its mean response time E_R. We compute it as follows:

$$E_R = max(TotalTransferTime) + ServerTime$$

1. The **Server Time** is the mean processing time of the server to complete its task and is equal to $T_0 + E_X/N$, where T_0 is a fixed processing time, $E_X$is the variable processing time and N is the number of servers;

2. The **Total Transfer Time** is the sum of the average time taken to transfer input data from server A to the N servers (*input time*) and the average time taken to transfer output data from each server to server A (*output time*). We transfer data using a **TCP connection** between server A and server i (i = 1,...,N): the choice of topology affects the transmission time value since it depends on the number of hops to reach all N servers.

3. When transmitting data, there is a certain amount of **overhead** involved. The total bits transmitted through the TCP connection are determined by the sum of the application job data and an additional overhead, which is a **fraction f of the initial data**.

4. Since transmission occurs in parallel, the **maximum value of the Total Transfer Time** corresponds to the last server that sends back the output data (this value defines the actual transfer time).

5. Finally, we normalize the mean response time by dividing each result with the baseline, which is the time taken when only server A is used and is equal to $T_0 + E_X$.

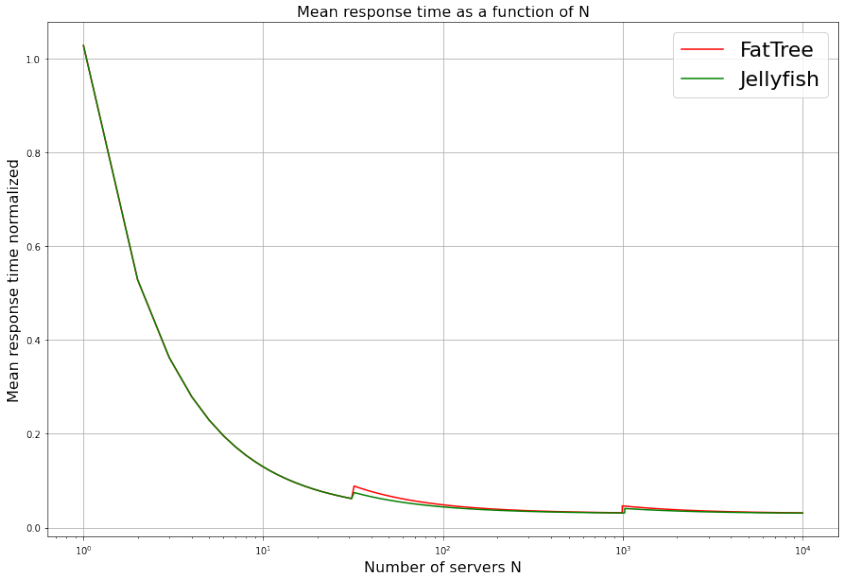


Figure 5: Plot of the normalized Mean Response Time

2.2 Evaluation of the Job Running Cost

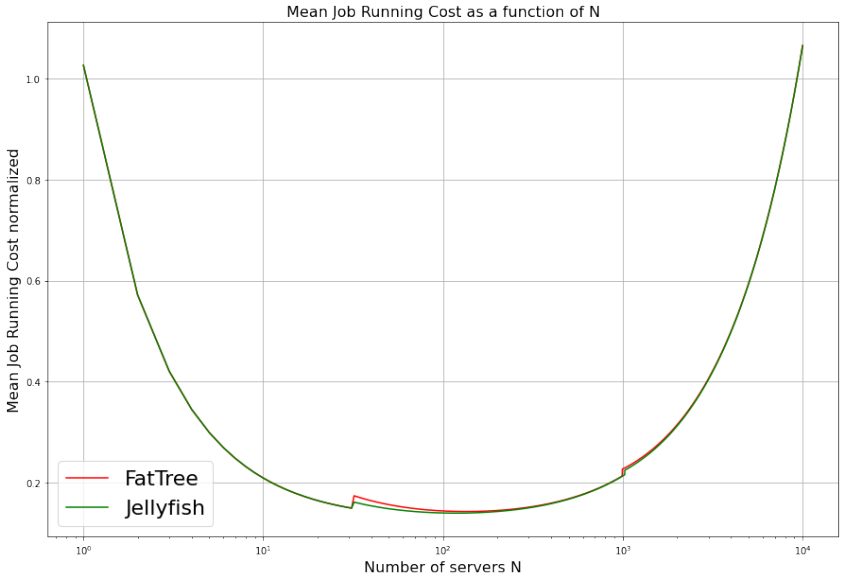


Figure 6: Plot of the normalized Job Running Cost

2.3 Analysis of results and takeaways

In computing Mean Response Time, we have observed a slight performance decrease in some cases where there is a change in the number of hops, and this seems to **affect FatTree more than Jellyfish**:

- Between approximately 1 and 32 servers, both FatTree and Jellyfish requires 2 hops to reach them;
- Between approximately 32 and 1000 servers, FatTree requires one more hop than Jellyfish;
- Between approximately 1000 and 10000 servers, FatTree requires two more hops than Jellyfish.

The **minimum Mean Response Time is achieved when 10000 servers are involved in both topologies**. This is due to the fact that the server processing time is $T_0 + E_X/N$, which approaches T_0 as N approaches infinity. Additionally, packets tend to become smaller and faster to transmit on average in both input and output.

The **minimum Mean Job Running Cost is archieved when around 100 servers are involved in both topologies** (135 for FatTree, 118 for JellyFish).

Overall, increasing the number of servers does not necessarily result in any significant cost savings: in fact, using 10000 servers we have the same Mean Job Running Cost of using only one server to process all the task (baseline).

In conclusion, the decision of number of servers N should be based on a **careful analysis** of the Data Center Topology Design, Response Time and Cost constraints.