SAPIENZA
UNIVERSITÀ DI ROMA

DEPARTMENT OF DATA SCIENCE

NBD - DC - Challenge 1

NETWORKING FOR BIG DATA AND LAB.

GROUP NAME: POSTEL

Professor:
Baiocchi Andrea
Cianfrani Antonio

Students:

Alvetreti Federico	1846936
Barba Paolo	1885324
Corrias Ioan	2079420

Academic Year 2022/2023

Part One

In order to compare the complexity of the three connectivity checking algorithms, we decided to run a Monte Carlo simulation using a ER random graph with $p = 0.2$.

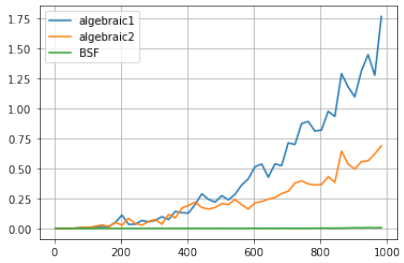

Figure 1: Complexity versus number of nodes K

As highlighted in the plot the best method is the BFS, this is due to the fact that BFS's complexity is $\mathcal{O}(|V| + |E|)$, where $|V| = K$ is the number of nodes and $|E|$ is the number of edges of the graph. The algebraic methods instead have respectively a complexity of $\mathcal{O}(K^3 \log(K))$ and $\mathcal{O}(K^2)$.

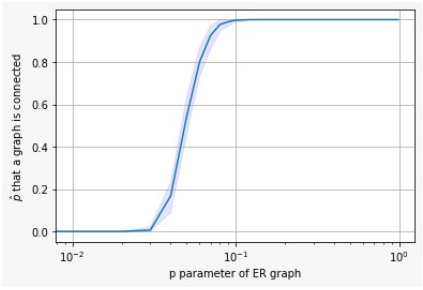

Figure 2: P(G) vs p for Erdos-Renyi graphs with 100 nodes

As shown above $\mathbb{P}(G)$ is a non-decreasing function with respect to the probability parameter, in fact the number of links expected in a ER graph is $\binom{n}{2}p$ and as p increases the graph will have enough edges in order to be connected.

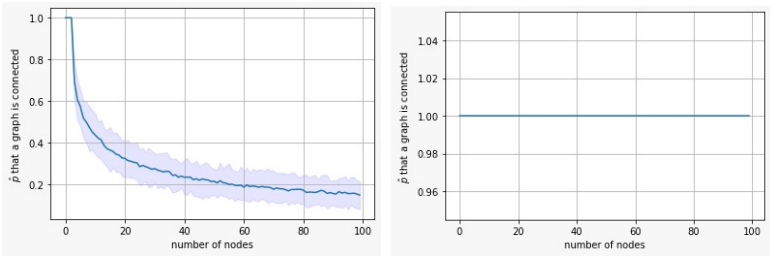

Figure 3: Probability of a regular random graph being connected

For $r = 2$, the only case for the regular graph to be connected is that it has a circular structure, because each node must have two neighbours, no clique are allowed and as the number of nodes increase this is very unlikely to happen.
For $r = 8$ the estimated probability that a regular graph is connected is 1 for any choice of n.

Part Two

To get a fair comparison between the **Fat Tree** and the **Jellyfish** topologies we assumed they had both the same number of servers (in our framework $\frac{k^3}{4} = 65536$).

The main difference between the two topologies, with regards to the **response time** function, is the number of hops between a fixed server A and it's n-th nearest **free** server.

In the case of the **Fat Tree** topology it can be easily shown that:

$$\text{fat\_tree\_hops}(n) = \begin{cases} 2 \text{ if } n \in [0, 32) \\ 4 \text{ if } n \in [32, 1024) \\ 6 \text{ if } n \in [1024, 65536] \end{cases}$$

For the Jellyfish topology the number of servers with a distance $d \geq 4$ hops from server A is random. However, the distribution of the number of servers at distance 4 from a fixed server A is the following:

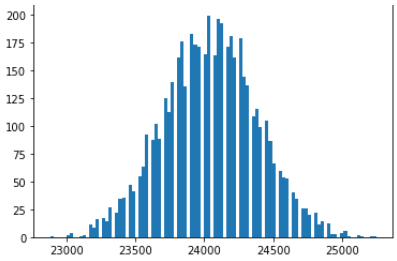

Figure 4: Distribution of #N with distance 4 from A

Since $\mathbb{P}(N < 1000) \approx 0$ we can avoid taking into account the randomness of this process and conclude that, even in the worst case, the number of hops between A and S_i, for $i \in [1024, 10000]$, is 4.

Our final step function is the following:

$$\text{jellyfish\_hops}(n) = \begin{cases} 2 \text{ if } n \in [0, 32) \\ 3 \text{ if } n \in [32, 1024) \\ 4 \text{ if } n \in [1024, 10000] \end{cases}$$

Let us fix the server A and the nearest n servers S_i, $i \in [1, n]$.
Let us call time(S_i) the time needed by each server to receive, execute and send back the packet. This is obtained by summing up three different times:

- **Transmission time from** A **to** S_i, dependent on input data size (which is fixed), hops number between A and S_i and connection relying parameters;
- **Time for** S_i **to execute the job**, obtained summing a fixed time constant to a random component;
- **Transmission time from** S_i **to** A, dependent on output data size (which is uniformly distributed in $[0, \frac{2L_0}{N}]$) and again the same hops number between A and S_i and connection relying parameters.

Since the process will end when the last packet will have arrived to server A, we finally compute the **response time** by taking the maximum:

$$\mathbf{RT}(n) = max_{i \in [1,n]} time(S_i)$$

In order to estimate the **average response time** we have performed a Monte-Carlo simulation for each $n \in [1, 10000]$.

The **job running cost** is defined as the average response time summed up to $\xi \cdot \mathbb{E}(\Theta)$, where Θ is the time that server A is used, if the job runs locally on A. Otherwise, Θ is the sum of the times that all N servers are used to run their respective tasks.

Again, in order to estimate the **average job running cost** we have performed a Monte-Carlo simulation for each $n \in [1, 10000]$.

These are the resulting plots:

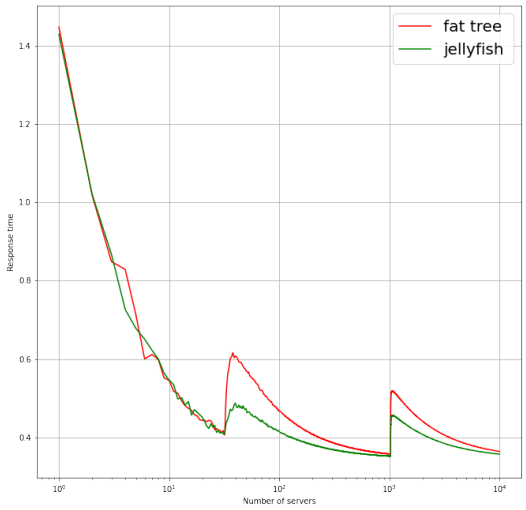

Figure 5: Normalized average response time

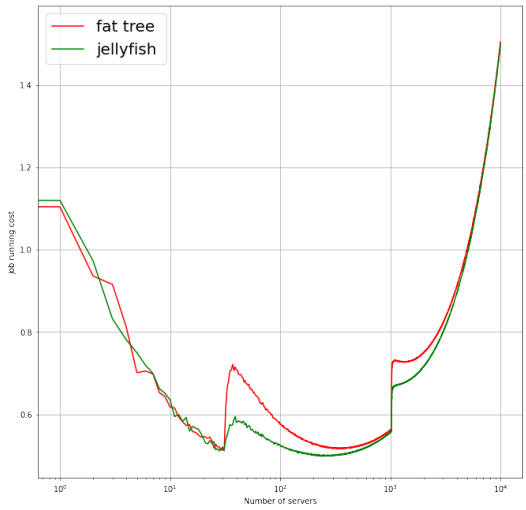

Figure 6: Normalized average job running cost

Normalized average response time Analysis

The main difference of the average response time between the two topologies relies on the two different hops functions. Indeed, for $N < 32$ (where the hops are equal) the two topologies present the same behaviour, while for $N > 32$ the Jellyfish average response time is always faster.

From the plot is clear that, passed a "hop-level", is always "time-convenient" to parallelize the task with all the servers in that same "level". Indeed, fixed a "hop-level", the functions always have a decreasing trend.

Normalized average job running cost Analysis

As expected the normalized average job running cost is a convex function.

The optimal number of servers are:

- 31 for the Fat Tree topology;
- 247 for the Jellyfish topology.

It is convenient to parallelize when:

- $N \in [2, 4958]$ for the Fat Tree topology;
- $N \in [2, 5097]$ for the Jellyfish topology.