Sapienza University of Rome

Networking for Big Data

Master in Data Science

Challenge #1

Group #24 Zipf

Miguel Ángel Sánchez Cortés
Pasquale Luca Tommasino
Sofia Noemi Crobeddu
Francesco Proietti

May 14, 2024

1 First Part

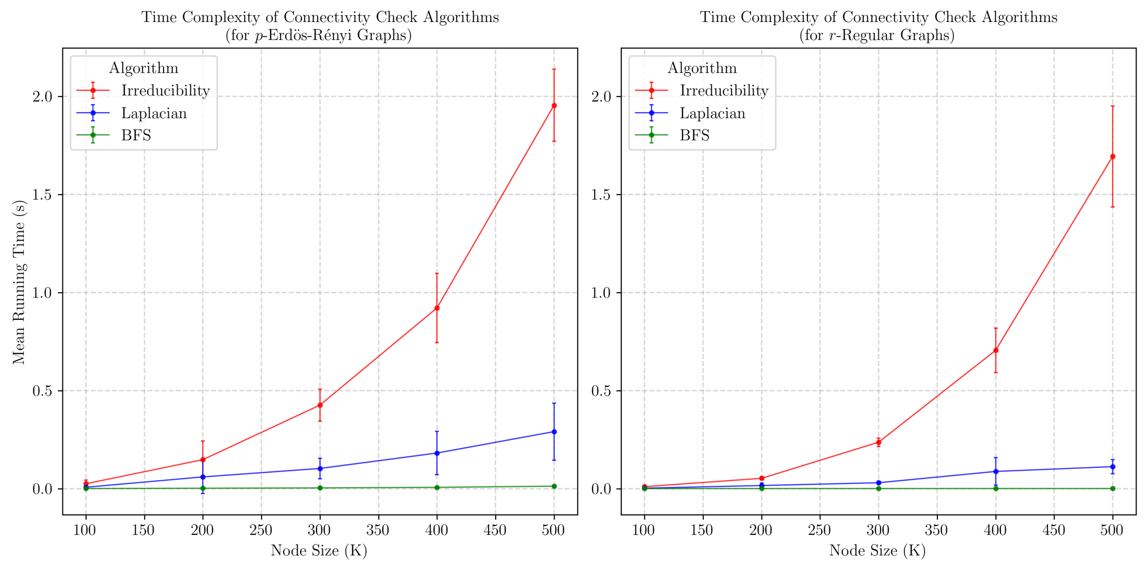

Figure 1: As we can observe from the plots above, the most efficient algorithm to check for connectivity for both a p-ER and r-Regular random graphs is the BFS algorithm, where the most inneficient is the Irreducibility algorithm. We used the Mean Running time (s) as a measure of complexity and calculated it by running 10 simulations for each algorithm.

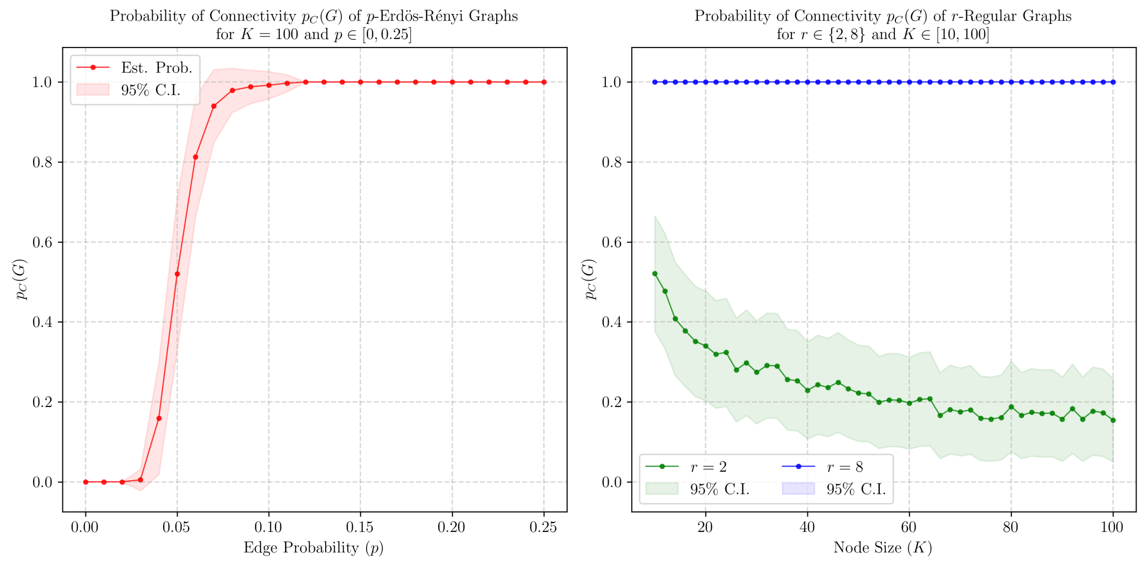

Figure 2: **(a)** As we can observe from the plot above, a p-ER random graph has a 100% probability of connectivity for edge probabilities $p \geq 0.1$. **(b)** In the case of r-Regular random graphs, we can observe that when $r = 2$, the probability of connectivity decreases with the number of nodes, whereas for $r = 8$ this probability is always 1, suggesting that small values of $r \approx 8$ are sufficient to guarantee connectivity for this random model. To obtain both plots we performed 1000 Monte-Carlo simulations using the BFS algorithm to check for connectivity.

2 Second Part: Algorithm

The *response time function* we wrote to evaluate the response time R takes as input the given data of the challenge $(C, \tau, L_f, L_o, E_X, T_0, f)$, the number of server N and a *hops list* which contains as items the hops from the starting server A to each different servers in the network, sorted in ascending order. In this way the index represents the node and the item is the number of hops from server A. The nearest one is *hops list[0]*, the further one is *hops list[N]* and the function select only the first N element of the *hops list*. The function returns as output *theta*, which is the sum of the times that all N servers are used to run their respective tasks, and the response time R, computed as the sum of the time to and back from server i plus the execution time: the biggest value among all N servers is the response time R (the time in which the last L_{o_i} output file is delivered to server A).

The code schema is the following:

1. creation of the topology (Fat Tree or Jellyfish);
2. random choise of a starting server (server A) and creation of the hops list as defined earlier;
3. one hundred simulations for each number of servers N, from 1 to 10000, of the *response time function* to evaluate the *mean response time* $E[R]$ as a function of N, and also $E[\Theta]$;
4. computation of the $E[R_{baseline}]$, always over 100 simulations, and $E[R]$ normalization;
5. computation of the *job running cost* S;
6. $S_{baseline}$ calculation and S normalization.

Algorithm 1: Simulations to find $E[R]$. This pseudo-code shows the simulation's iteration in order to collect samples of R. In the end the mean of R is computed among all the values of R collected.

```
num_sim ← 100
E_R_list ← [ ]
E_Θ_list ← [ ]
for $N = 1$ to 10000 do
    num_server ← $N$
    sample_R ← [ ]
    sample_Θ ← [ ]
    for $i = 1$ to num_sim do
        $R, \Theta$ ← response_time_func(given_data, N, hops)
        sample_R.append(R)
        sample_Θ.append(Θ)
    end
    E_R ← mean(sample_R)
    E_R_list.append(E_R)
    E_Θ ← mean(sample_Θ)
    E_Θ_list.append(E_Θ)
end
```

3 Second Part: Results

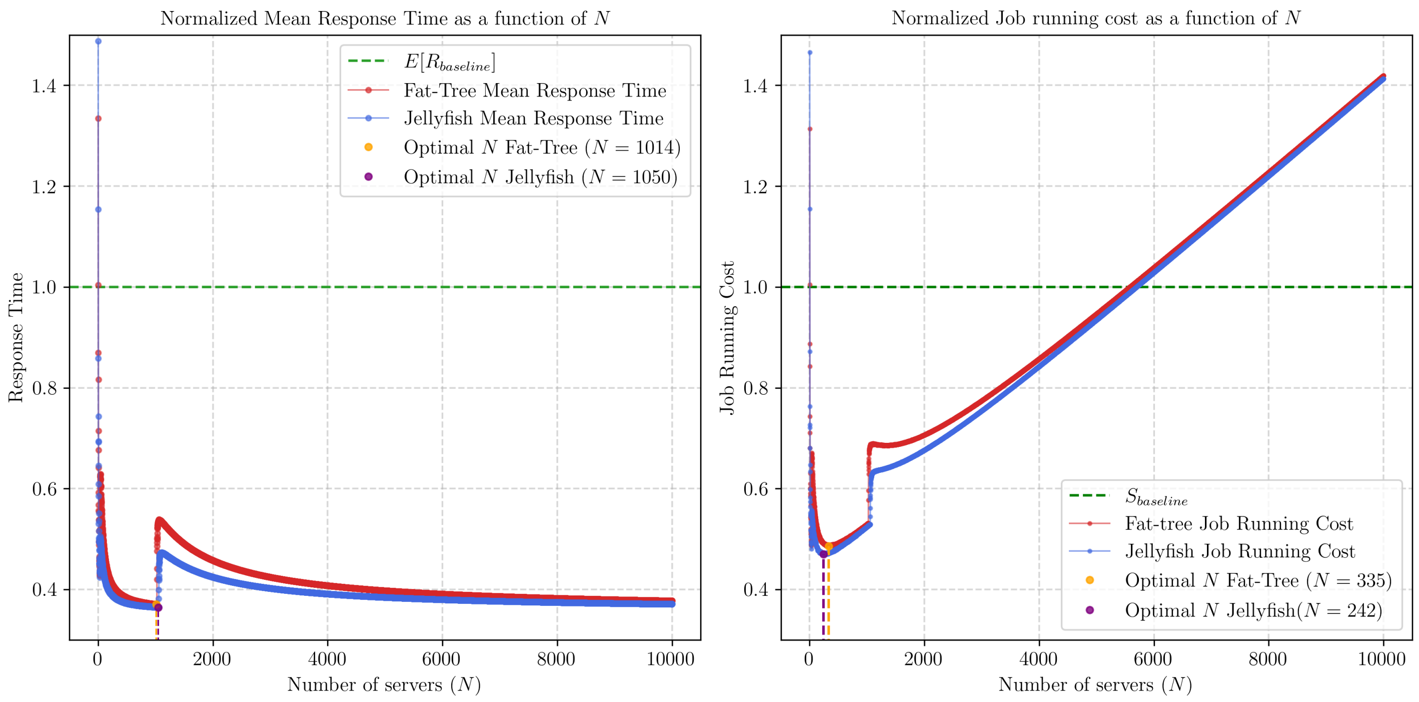

Figure 3: **(a)** Normalized mean response time $E[R]$ as a function of the number of servers N. **(b)** Normalized job running cost S as a function of the number of servers N. The optimal N server points are shown in the plot with an orange and a purple point, respectively for Fat Tree and Jellyfish DC networks, for both plot.

In the Normalized Mean Response Time, the two curves are strongly similar. In fact, the number of servers yielding the best performance in terms of average response time is around 1014 number of servers for the Fat-tree, and around 1050 number of servers for the Jellyfish typology. However, starting from 4000 servers, the curve tends to have a horizontal asymptote and tend to at the minimum value of the type, and the average time consequently tends to be the same as the minimum (not reaching it).Likewise, in the Normalized Job running cost, the two curves are strongly similar. In terms of Job running cost, we have the best performance around 335 number of servers for Fat-tree, and around 242 number of servers for Jellyfish. In contrast to the first graph, once the minimum is touched, the Job running cost tends to increase a lot.

We ran several simulations on the same algorithm to have different samples but the structure of the curves tends to be the same: the best performance is always achieved for an amount of servers around 1000 for both topologies (Fat-tree and Jellyfish) in terms of average time response, and around 300 for Fat-tree and around 250 for Jellyfish in terms of Job running cost.

In conclusion, we can see that the blue curve (Jellyfish topology) is always below the red curve (Flat-tree), consequently we can say that the Jellyfish is the best type to use and that dwelling on the around 1000 servers we can benefit from low cost of both Job Running Cost and Mean Response Time.