Complexity Report

The last part of the Operations Research is the complexity study of our functions.

What's the complexity ? :

The complexity is simply about studying the execution times of each function, to know which one is the best to use and which one is the worst, based on the results we get from these two functions.

To do that, we'll generate randomly the provisions and the orders, with numbers between 1 and 100 (inclusive). We'll also generate randomly the cost matrix, with numbers between 1 and 100 (inclusive).

Then the goal will be to do that a hundred times for each algorithm, in order to get a pretty much accurate results. It's like probability, to get a result that seems closed to the reality, you need to try it 100, 1000 times and then conclude.

We got several functions to do that :

- The first one is generate_random_values(n). This function will generate three random list, one for the provisions, one for the order and a 2D cost matrix. The size of these matrix will be respectively n, n and $n_x n$. n is given as a parameter of the function.

- The second function is complexity_[name_of_the_algorithm]_initial(n). This function will create a empty list exécution_times and will make a loop of a hundred iterations, which will execute our function generate_random_values at each iteration and then apply one of the algorithm we chose, between Northwest and Balas-Hammer. Two variables surrounding the execution of the function and denoted start_time and end_time, are given their values thanks to the function time.process_time(), which will take the time of the CPU at the start of the function, and at the end. Then we'll just add to the list the substraction of these two values to get the execution times of each iteration

- The third function is plot_complexity(exécution_times, execution_times2). This function take as parameter exécution_times and exécution_times2, which are supposed to be list of exécution times, in order to be able to plot all exécution times value of each list in the same Scatter plots, with different color, in order to differentiate them.

North-west and Balas-Hammer initialization :

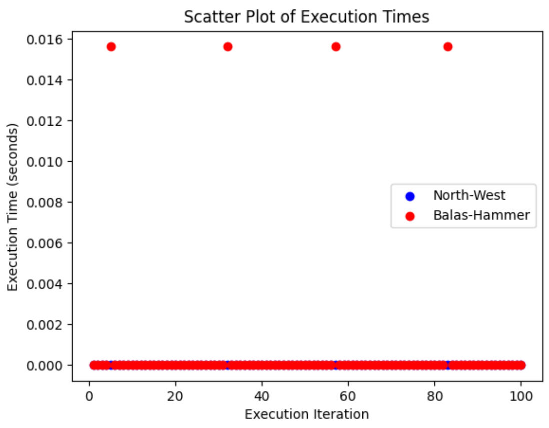
We'll start with the complexity of two of our algorithms, which are North-west and Balas-Hammer. The goal is to find which one seems more complex and take more time to be executed, and see if it seems obvious due to what each function does.

To get that result, we'll simulate each algorithm with n = 10, 40, 100, 400, 1000(only for North-west).

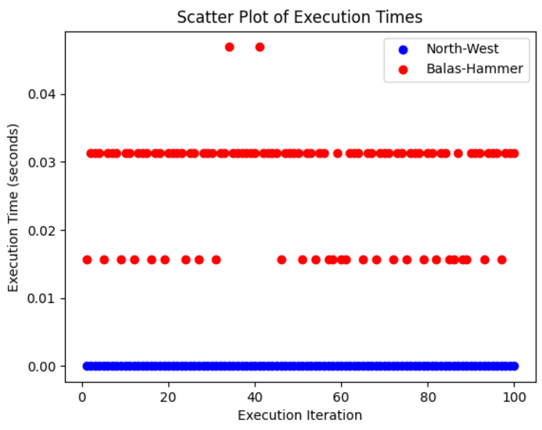
Due to a really high time of executing Balas-Hammer, we won't have the results for n = 400, 1000, 4000, 10000. It would have taken weeks to compute them, time that we unfortunately don't have.

Results:

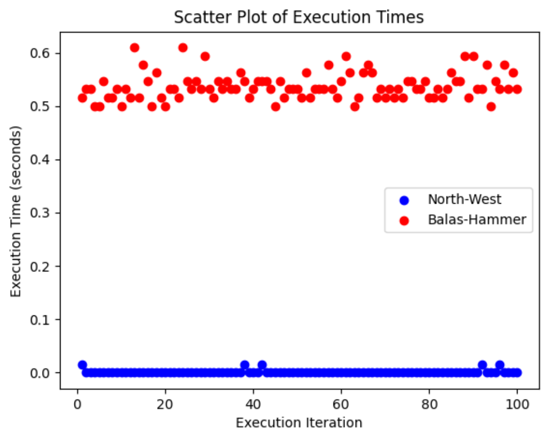
n = 10 :



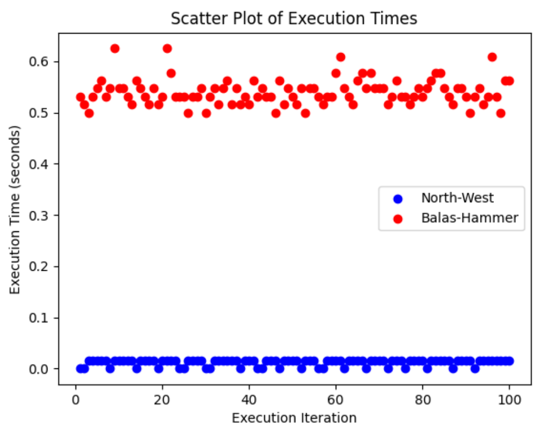
n = 40 :



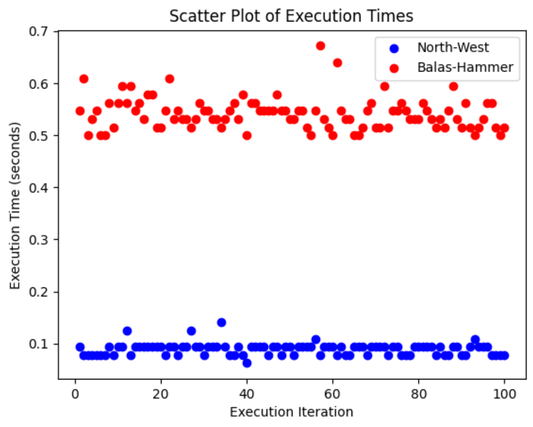
n = 100 :



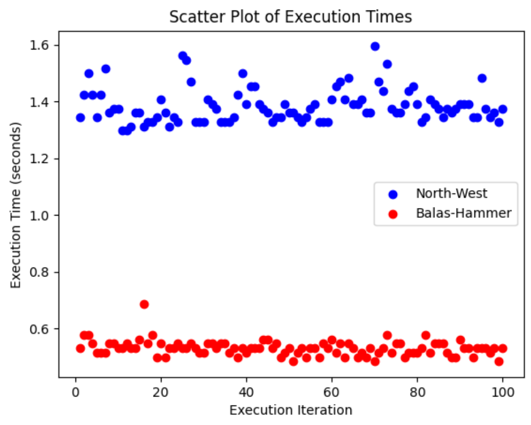
n = 400(for North-West only) :



n = 1000(for North-West only) :



n = 4000(for North-West only) :



What can we conclude from these results:

For the results above we can observe that the North-West initialization is way faster to execute than the Balas-Hammer. This is due to the difference of complexity between the tw o functions. Even when N = 1000 for the North-West and N = 100 for the Balas-Hammer, we still have a much higher exécution time for Balas-Hammer. Unfortunately it is not possible to compute the Balas-Hammer initialization for N > 400, because it takes too much time for the CPU to compute it. However, we can see that it's only when N = 4000 for North-West that the exécution times for it will be much higher than for Balas-Hammer with N = 100.

Conclusion & Possible Improvements:

To conclude, we would say that our function may not be very optimal, and if they were more optimal we could have tested them on a larger scale of transportation problem.

However, we can still say that North-West initialization is way less costly in terms of CPU ressources, and require way less time to execute. Nevertheless, even if we didn't have time to do it, we know that with the optimization, North-West would be even more longer than Balas-Hammer as it's not a very optimized way of initializing.

To improve our complexity study, we wanted to be able to compute the exécution times of each initialization method, followed by the stepping-stone method. However, we didn't give enough time to this part of the project to be able to do it.