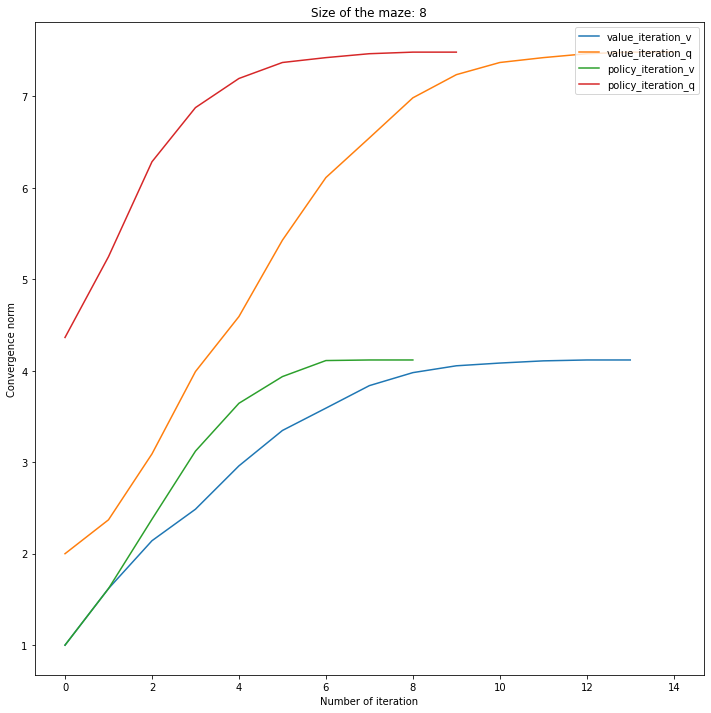
DAM Damien & DURAND Enzo

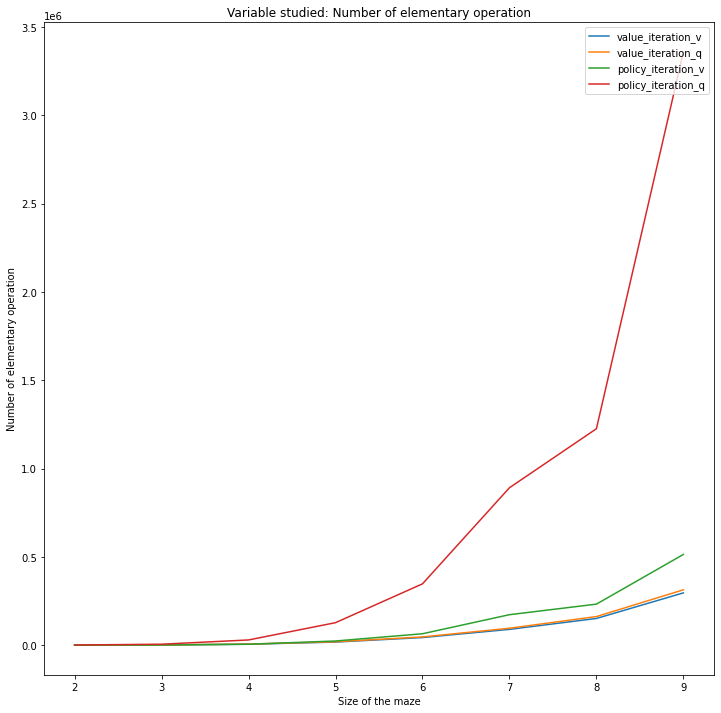
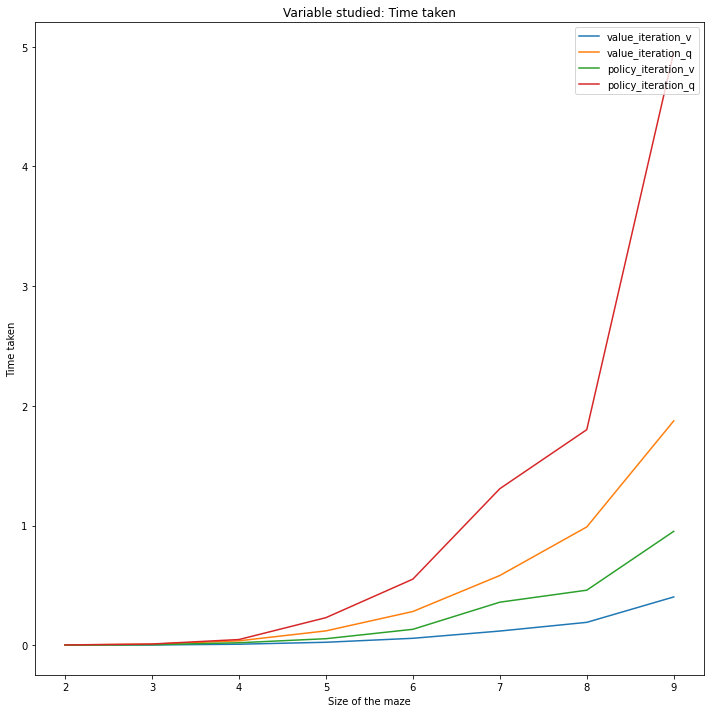
**TME 1**

In this first mini project, we studied the basic of Reinforcement Learning via two dynamic programming algorithms being Value Iteration (VI) and Policy Iteration (PI). In a maze environment with perfect knowledge of the transition function and its associated reward, both algorithms are able to efficiently output the optimal policy. What is left for us is comparing the two, which one is more efficient in terms of execution time, number of iterations, or number of operations?

First, we look at the number of iterations until convergence for each algorithm in the image below.



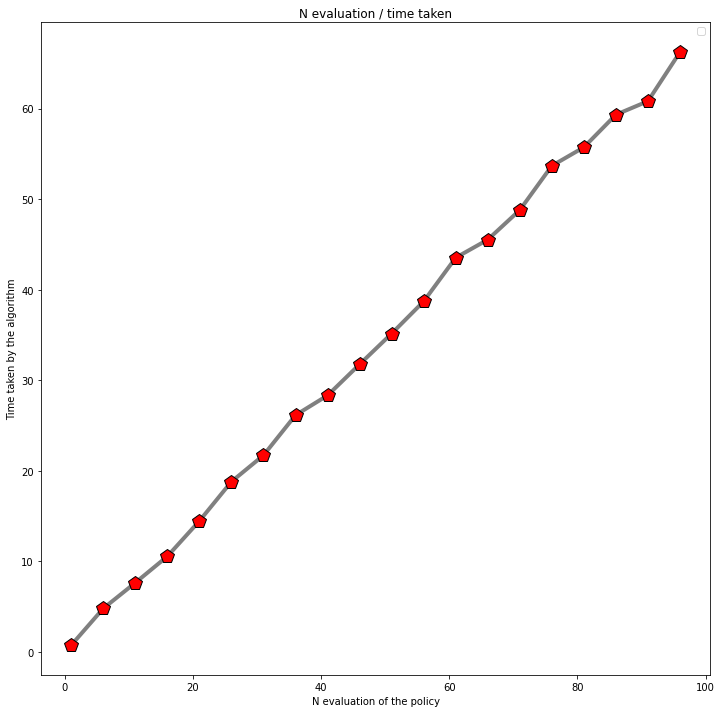
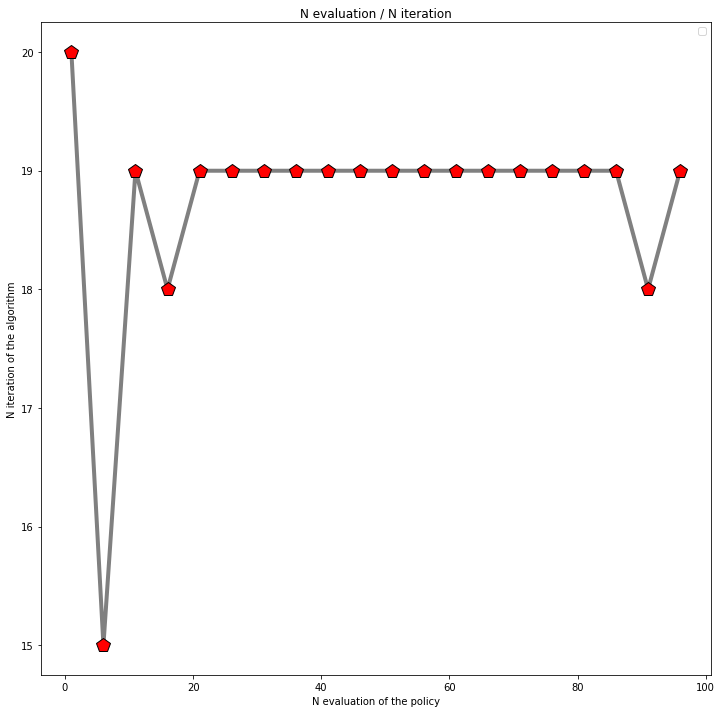
We can see that PI requires much less iteration runs than VI. This happens because in one iteration, this algorithm updates the policy multiple times while VI makes a copy of the current policy before each iteration and updates it fully once it is done. However, the images below tell another side of the story.



As we can see, even though PI requires less iteration than VI, its execution time is much greater than VI. As the size of the maze increases linearly, the time taken for the algorithms to convert grows exponentially. The explanation lies in the image on the right, which depicts the number of elementary operations done by each algorithm. If we look closely, we can observe that while the number of iterations in VI is greater, the algorithm only performs a small number of operations in each iteration. On the other hand, during each iteration of PI, the algorithm performs lots of calculation, especially during the policy improvement step. This is why the total number of operations in PI is far greater than that of VI.

As for comparing between using V-value and Q-value in an algorithm, we see that using V-value has a slight advantage over Q-value, since the former looks only at all possible states of the environment, while the latter considers all pairs of state-action values.

The last part of the project consists of coding the Generalized Policy Iteration.

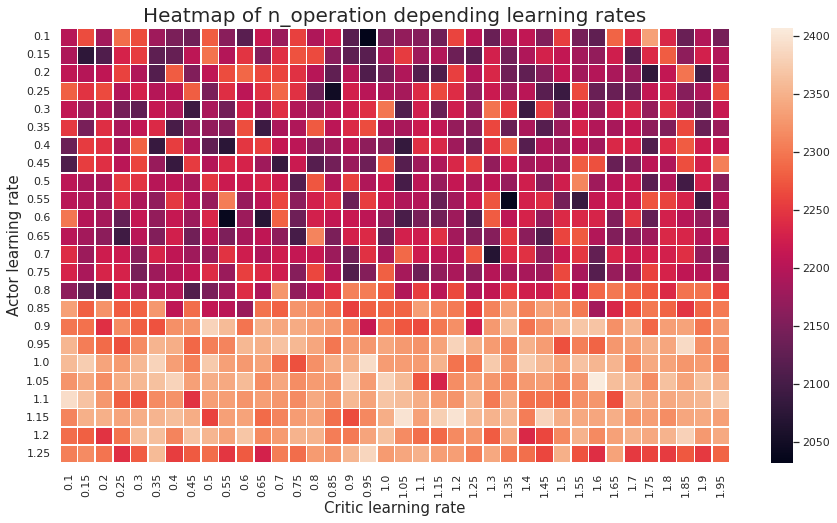
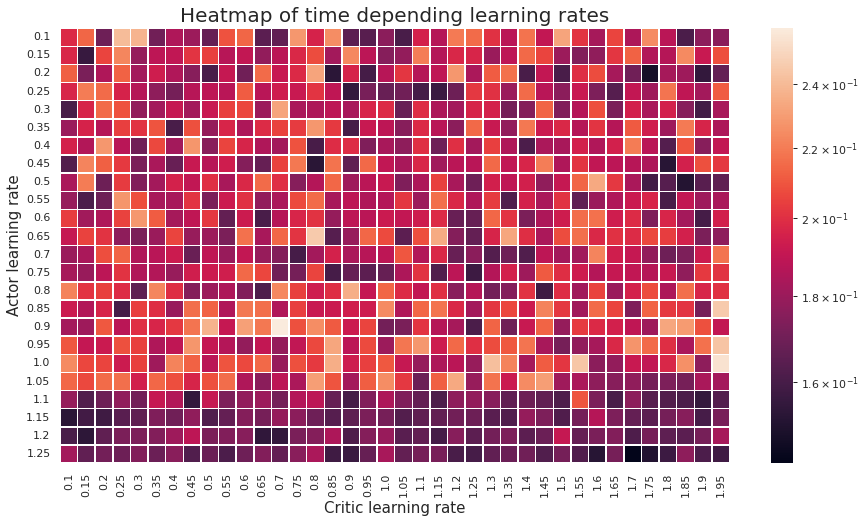


We can see from the image on the left that we obtain the same convergence rate no matter what the number of Q-evaluations (N) is before updating the policy. Thus, it seems to be reasonable to choose a small value for N so that convergence is faster in terms of execution time (image on the right).

**TME 2**

For this second mini-project, we programmed an agent using the actor-critic algorithm trying to navigate its way in a 10x10 maze. The code for the algorithm is included with this report.

The hyper-parameter tuning step for this algorithm consists of choosing the most optimal pair of learning rates for the critic and the actor, being and . To find the best pair, besides beginning each run with the same random seed to ensure the same starting point for all experiments, we also used the *Grid search* technique that iterates over all combinations of a limited set of values for both parameters, run the algorithm, and outputs some metrics that we can compare. Below are the heatmaps showing how average execution time over 100 runs of the same parameters, and the number of operations are dependent on the values of and .



Based on the color of the heatmaps, we can see that for this particular environment (10x10 maze), the optimal values are around and and that should usually be greater than .

To further illustrate the efficiency of the optimal pair of hyper-parameters, we compare its convergence against the worst pair of values. The figure below clearly shows that after episode 45, the optimal model has already converged, while the worst one is still stuck.

