ECOLE POLYTECHNIQUE DE BRUXELLES

Academic year : 2023-2024

INFO-H410

Project Report : Techniques of Artificial Intelligence

Group Members :

BOISTEL Julien

NKWINGA Eurielle

ÖKTEM Oguzhan

Teacher :

Hugues BERSINI

Table des matières

1 Introduction

Initially, the code was designed to solve the shortest path problem, which was then modified to address the Traveling Salesman Problem (TSP). The TSP is a classic problem in graph theory, involving finding the shortest route that allows a traveler to visit a set of cities exactly once before returning to the starting city. Unlike the shortest path problem, where the objective is to find a path with the lowest cost between different points, the TSP requires all cities to be visited before returning to the starting city, significantly increasing the complexity of the problem.

To take this complexity, an approach based on Q-learning, a reinforcement learning method, was chosen. This report details the implementation choices, challenges encountered, and solutions provided during the development of this optimization method to solve the TSP.

2 Implementation of Q-learning algorithm

To implement the Q-learning algorithm, we started by initializing the Q and R matrices. The Q matrix stores quality values for transitions between states, while the R matrix represents rewards for transitions, which are the negative distances between points (the shorter the distance, the higher the reward).

For each episode, the algorithm randomly selects a starting point and explores the graph following an epsilon-greedy policy. The traveler traverses the cities by choosing either a random action (exploration) or the best action according to the Q matrix (exploitation). Transitions are updated according to the Q-learning rule, which favors transitions with the highest rewards, using the Bellman equation :

$$Q\left(S,S'\right) = Q\left(S,S'\right) + \alpha \cdot \left(reward + \gamma \cdot \max\left\{Q\left(S',:\right)\right\} - Q\left(S,S'\right)\right) \tag{1}$$

After each episode, the value of epsilon is reduced to encourage exploitation. The length of the tour is recorded at each episode, allowing for tracking the improvement of the algorithm over time. At the end of the training, the best tour found is extracted and displayed, along with the evolution of tour lengths over the episodes. This approach allows the algorithm to gradually find shorter and shorter solutions by balancing exploration and exploitation.

3 Discussion of solutions

At the beginning of the execution, we observe many perturbations in our curve before noticing a convergence towards the end, indicating the approach to the best solution. During the execution of our algorithm, the solution obtained at the end is not necessarily the best one. To address this, we created a variable best-tour that compares the costs of the tours during the execution and stores the best tour found. This variable ensures that we retain the tour with the lowest cost encountered throughout the algorithm.

3.1 Influence of the learning rate

We can observe on Fig. 1 that the learning rate has a significant influence on the aspect of the curve. A higher learning rate results in a faster initial learning but is less constant due to more aggressive updates of the Q-values. With a lower learning rate, it can be seen of the right that the convergence is smoother but slower on the first episodes.

FIGURE 1: Influence of the learning rate

3.2 Influence of the epsilon decay

The decay of the epsilon value also has a visible influence on the results. The epsilon is related to the ϵ-greedy policy. The lower the value of ϵ, the less exploration will be done.

```
if np.random.uniform(0,1) < epsilon:
    # Explore: go to a random node
    next_state = np.random.randint(0,self.V)
else:
    # Exploit: go to the best node (the closest one)
    possible_next_states = Q[state,:]
    masked_states = np.ma.array(possible_next_states, mask=[i
                    in visited for i in range(self.V)])
    next_state = masked_states.argmax()
```

The decay of this epsilon allows to explore much more at the beginning of the algorithm. Then focus more on exploitation of the better values to achieve a better convergence. This is well seen on Fig. 2 : when there is no decay (on the right), there is less convergence because the algorithm still explores a lot for solutions.

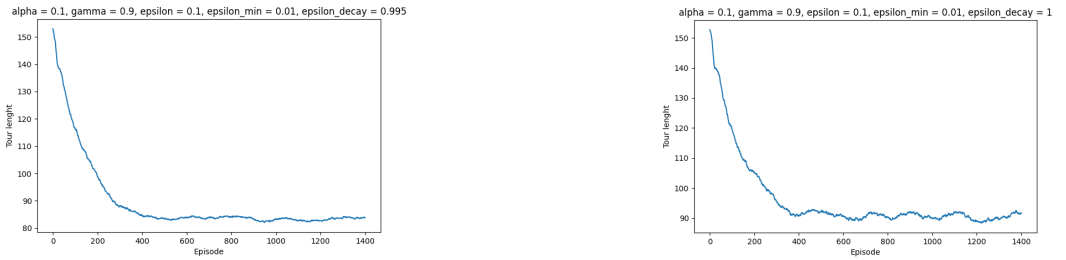

FIGURE 2: Influence of the epsilon decay

3.3 Number of nodes

While these results may seem intuitive, it's important to explore the details. As the number of nodes in the problem increases, so too does the number of iterations required for the solutions to converge. This can be attributed to several factors :

1. State Space Complexity : The problem's state space grows exponentially with the addition of each node. More nodes mean more possible path combinations, which significantly increases the problem's complexity.
2. Exploration vs. Exploitation : In Q-learning, balancing exploration (trying new paths) and exploitation (using known good paths) is crucial. A larger number of nodes necessitates exploring a greater variety of potential paths to find the optimal solutions, thus requiring more iterations.
3. Q-value Updates : Q-learning depends on iteratively updating the quality values (Q-values) for each state-action pair. With more nodes, there are more Q-values to update, which slows down the convergence process.
4. Convergence and Optimality : Achieving convergence to an optimal solution takes longer because Q-learning must identify and rectify sub-optimalities within a larger state space. This often necessitates more iterations for the Q-values to accurately reflect the accumulated rewards from different trajectories.

FIGURE 3: Influence of the number of nodes

4 Conclusion

The primary objective of the project was to utilize an algorithm optimization method, with a focus on applying Q-learning to optimize the solution of the Traveling Salesman Problem (TSP).

The use of the Q-learning algorithm provided a solid foundation for route optimization, thus simplifying the process of finding the shortest paths. Rewriting the code with exploration-exploitation policies (epsilon-greedy) ensured efficient exploration of possible solutions while converging towards optimal solutions.

Ultimately, this project allowed for a deeper understanding of reinforcement learning mechanisms, the practical application of Q-learning concepts, reward and quality matrices, and demonstrated the effectiveness of these techniques in solving complex problems like the TSP.