Exercise 5: An Auctioning Agent for the Pickup and Delivery Problem

Group №: Student 1, Student 2

November 25, 2018

1 Bidding strategy

Our bidding strategy depends on three parameters, our current estimated cost per task, the probability distribution of the tasks, and the previous bids of our best performing opponent. Our current cost is determined as being the average cost per task. Meaning that at each auction we calculate the optimal cost to pick up and deliver all the previously won tasks, adding the potential task of this round. For this we use our Steady state genetic algorithm that runs to the maximum of the bid timeout. Then we divide this total cost by the number of tasks.

To calculate our next bid we perform as follow:

At each auction we calculate the difference between our estimated cost and the bid of the opponent. We then apply a sigmod function $s(x) = 1 - \frac{1}{1+e^{-\sigma \cdot (x - shift)}}$ to these values and we create a weighted average of this difference.

$$weightedMean = \sum_{i=0}^{number\ of\ bids} \frac{s(i)}{\sum_{j=0}^{number\ of\ bids} s(j)} \cdot (bid_{Enemy}(i) - cost_{Agent}(i))$$

The sigmod function has 2 parameters the slope σ and the $shift$ that change dynamically. We define σ as the normalized standard deviation of the previously calculated differences between our cost and the opponent bid. It follows the formula: $\sigma = \sqrt{\frac{var_{bid\ Enemy}}{mean_{bid\ Enemy}}}$. This allows us to dynamically adapt to the large changes in the bid of our opponent while maximizing precision when his bids are more consistent.

The shift is always equal to the current number of auctions that finished. This allows the shifting slope be at the edge of our values and change the distribution by modifying σ.

We also use the task distribution in the calculation of our cost. To do so we try to forecast the potential future average cost. The steps are as follow: We begin by taking the last city C of the current configuration. Then we add to it one possible task from the cities around C. This is when we calculate the cost with this added task. We repeat this operation for each possible task in all neighbor cities. Finally, a weighted average is calculated using the probability distribution of said tasks. This weighted average is our predicted cost.

2 Results

2.1 Experiment 1: Comparisons with dummy agents

2.1.1 Setting

We performed the first experiment by fighting our main agent vs the most naive one. The map used is the Netherlands and all the parameters are set as default. The seed we used was 123456 and we performed 2 sets of experiments one with 20 tasks and the other with 60 tasks. Each set is run 4 times each 2 with the agents switching positions and the displayed results is the average between the 2 for each position. The Naive algorithm calculates its cost without optimizing the path and chooses its bids randomly between -10000 and 10000.

2.1.2 Observations

The total rewards we got are the listed in the table below:

Number of task used	60 Tasks		20 tasks	
Initial position of the agents	position1	position2	position1	position2
Main	928294.5	1457376	308835	663201
Naive	-928213699	-1175010799	-1333254	-168194732.5

Table 1: Test with the Naive algorithm

We can see from the results that our main agent not only beats the naive one but also maximizes its winnings by analyzing the behaviour of Naive.

2.2 Experiment 2

2.2.1 Setting

In this experiment, we are using a previous version of algorithm which is called Manuel. This algorithm doesn't include the sigmod function that creates a weighted mean of the differences between the opponent bids and our costs. It also doesn't take into account the probability distribution of the tasks.
The conditions of this experiment are the same as the previous one.

2.2.2 Observations

The total rewards we got are the listed in the table below:

Number of task used	60 tasks		20 tasks	
Initial position of the agents	position1	position2	position1	position2
Main	6583	16553.5	8185	3452
Manuel	2881	13197	5473.5	-4140

Table 2: Test with the algorithm Manuel

We can see that our main algorithm outperforms Manuel as it dynamically changes the weights of the values that are taken into account in the bid estimation. Manuel tends to send bids that are too high because it is highly impacted by high previous results.

2.3 Experiment 3

2.3.1 Setting

In this experiment, we are using a previous version of algorithm which is called Joseph. This algorithm differs from the main one as it doesn't take into account the probability distribution of the tasks. The conditions of the experiment remains the same.

Number of task used	60 tasks		20 tasks	
Initial position of the agents	position1	position2	position1	position2
Main	14329	6251	13638	4900
Joseph	12719	1730	15917	2561

Table 3: Test with the algorithm Joseph

2.3.2 Observations

We can see that the main algorithm slightly outperforms Joseph. However, the prediction of the tasks doesn't always gives us the optimal results as we can see in the table 3.

3 Improvement

By comparing with some peers, we noticed that our algorithm performs particularly well when the number of tasks is large. However, it seems to us that our strategy could be improved adapting the strategy relatively to the number of tasks auctioned.