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

Group №89: Wenuka Gunarathna, Sofia Dandjee

November 22, 2020

1 Bidding strategy

1.1 Marginal cost computation

We compute the marginal cost of the tasks auctioned by using our centralized planner that implements stochastic local search (SLS) to find an optimal plan. In order to use the previous optimal solution (without the current auctioned task), we insert the new task to the solution by appending its pickup and delivery (consecutively) to a vehicle whose plan goes through a city that is closest to the task pickup city. Then, we run the search starting from this initial solution, which makes the finding of the optimal solution faster.

1.2 Estimating the opponent's strategy

To guess the opponent bid, we record its marginal costs and bid history and we compute its optimal plan every time it wins a task. At each auction, we estimate the ratio by which it multiplies its marginal cost to make its final bid. This is done by averaging the ratio between its bids and its marginal costs. This estimate is done during the last `maxLookBackForEstimate` rounds. We also make sure this estimate is within 0 and `maxEstimateRatio`.

1.3 Fooling the opponent

When the opponent's predicted marginal cost is 0 and ours is not, we will most likely lose the bid. In such instant, we want to make the most out of this situation. Therefore, we bid a value of ∞ to interfere with our opponent's estimates of our bids so that if the opponent is simply taking a mean of our bids, it will get fooled. However we will only do this once, so that the opponent will not get benefits when it predicts a higher value in future.

To evade the same thing happening to our agent, our agent will only look back to the last `maxLookBackForEstimate` bids so that the effect of one very high bid will be removed eventually.

1.4 Future tasks

We consider future tasks that might be auctioned by generating random future tasks according to the task probability distribution provided by the Logist platform. As we know there is going to be a minimum of 5 auctions, during the first 4 auctions, we generate $5 - numAuctions$ future auctions for which we compute the average marginal costs which gives us an expected marginal cost. The final marginal cost is computed to be the minimum value between the current marginal cost (considering only the current task auctioned) and the expected marginal cost (considering the average cost of the future tasks). Then, we compute the marginal reduction between getting this task or not and if it is positive, we remove it from

the marginal cost. In other words, we will bid lower than the marginal cost expecting to cover that cost in the future.

Predicting future tasks is challenging because the agent does not know how many more tasks will be auctioned, therefore it will get a lesser profit if the auction finishes earlier than it expects. To eliminate such adverse effect to our final profit, we only predict in the first `minAuctions` auctions which we have set to 5 because we know there will be at least 5 tasks for each run.

1.5 Final bid

Finally, we bid the maximum value between the estimate of opponent bid lowered by `epsilon` and our own marginal cost multiplied by a `profitMarkup`. We make sure that this value is not lower than a minimum profit `minProfit` that we want to make in case our marginal cost is 0. We estimate `epsilon` by taking 20% of the averaged winning bids.

2 Results

For this section, we are running the same experiment twice, only exchanging the vehicles, so that both agents have a fair chance. Further, all the below results were derived using 10 tasks distributed in England's topology. Each agent has two vehicles which they can use to deliver tasks.

2.1 Experiment 1: Comparisons with dummy agents

2.1.1 Setting

We compare our agent with the dummy agent provided in the template, an agent that makes use of a naive bidding strategy (without taking the opponent strategy into account). Both agents compute their optimal plans with the SLS algorithm. The parameters of our agent are `epsilon` = 20% of the average winning bids, `maxLookBackForEstimate` = 5, `maxEstimateRatio` = 2.0, `profitMarkup` = 1 and `minProfit`=`epsilon`.

2.1.2 Observations

The dummy agent wins 2.5 tasks in average with an average profit of 4116 while our agent wins 7.5 tasks in average with an average profit of 6175. This shows the importance of looking forward and adjusting bids according to the history of bids.

However, it is also important to see that since the dummy agent does not bid according to the bidding history, the effect of concepts like Fooling can adversely affect our profit.

2.2 Experiment 2: Greedy agent

2.2.1 Setting

The agent has three parameters that define the greediness: `profitMarkup`, `epsilon` and `maxEstimateRatio`. In this experiment, we are seeing the effect of each parameter with respect to the default values of `profitMarkup`=1.0 (i.e- no profit, nor loss), `epsilon`= 20% of average bid values and `maxEstimateRatio`=4.0.

In the first run, we are using two agents each having 1 and 1.05 for `profitMarkup` respectively keeping all other variables fixed. Next we change `epsilon` to 10%, 20%, 30% percent of the average winning bids (as it is better to have a value that depends on the auction environment). Then we try to find the effect of `maxEstimateRatio` with values of 2 and 4.

2.2.2 Observations

For the `profitMarkup` experiment, we got an average of 5 tasks with a profit of 2470 for value 1 and an average of 5 tasks with a profit of 2460 for a value of 1.05. The profit increase is because now we try to get a 5% more for the marginal cost bids. However this might be in expense of losing a few bids as it can make us bid higher than the opponent even if we have a lower marginal cost.

For the `epsilon` experiment, we have got an average of 5 tasks for all the cases and an average profit of 2200, 2186 and 2057 for the cases of 10%, 20% and 30% respectively. In this case, it is important to understand that a lower `epsilon` results in a higher profit, however we might lose an auction due to an error in predicting the opponent's bid.

The `maxEstimateRatio` is also a notion of risk, as it indicates how much we are likely to increase our bid with respect to the predicted bid of the opponent agent. We got an average of 6.5 and 3.5 tasks with profit of 4357 and 2874 for the values of 2 and 4 respectively. In this case, we see that it is important to bound the ratio estimate, otherwise we overestimate the opponent's bid which makes us lose the task.

2.3 Experiment 3: Foresighted agent

2.3.1 Setting

We compare two agents with the default parameters mentioned above, but one considers future tasks and one does not. For better comparison, we have done the same experiment for the number of tasks 5, 10, 20 and 50.

2.3.2 Observations

For the agent which considers future, it took 3, 6, 9 and 20 tasks with an average reward of 1624, 2510, 2345, 8451 for the task count of 5, 10, 20 and 50 respectively. In contrast, for the agent who did not consider the future, it took 2, 4, 11 and 30 tasks with an average reward of 0, 254, 1524, 8716 for the task count of 5, 10, 20 and 50 respectively.

One important factor to consider here was that in some experiments, our agent ended up with a deficit, however that loss was negated when averaging and we can see that in most cases, the agent predicting the future works better.

2.4 Experiment 4: Cunning agent

2.4.1 Setting

For this, we have used one agent who tries to trick the other agent by giving a spike to the bids if it knows it will anyway loose the bid and another agent who does not do it. The second agent was tested for both evade strategy where it takes the average of full opponent history or only average the bids over the last 5 steps.

2.4.2 Observations

In the first case where the not-cunning agent average bids over the last 5 bids, it took 4 tasks with an average profit of 2029 while the cunning agent got 6 tasks with a profit of 3856. In the second case where the not-cunning agent takes the average of the full history, it ended up with 4.5 tasks with an average profit of 304 while the cunning agent took 5.5 tasks with a profit of 1685. This shows the importance of fooling an opponent who is trying to average our agents bids. The evade strategy to look back only on the last 5 tasks helps us a bit in case the other agent tries to trick us the same.