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

Group 43: Michael Martini 331039, Philine Witzig 331038

November 20, 2020

1 Bidding strategy

Our agent computes the plan for delivering tasks using the SLS algorithm in a warm-start setup. Whenever a new task T is available, we compute the new plan $plan_{new}$ by adding $PickUp_T$ and $Delivery_T$ at the end of the old plan $plan_{old}$ and running an upgraded version of the SLS algorithm on it. Instead of fixed iterations, this version uses a cap on the iterations in which no improvement was reached. We compute the marginal costs as

$$c_{Add} = cost(plan_{new}) - cost(plan_{old}).$$

Note that the marginal costs can be negative in case adding T to our already assigned tasks and running SLS again leads to an improved plan with lower costs. Based on that, we construct our bid for task T as follows:

$$bid_T = \begin{cases} c_{Add} \cdot (1.7 - p_{zero}) & \text{, if } c_{Add} > 0 \\ 0.99 \cdot E(bid'_T) & \text{, else} \end{cases}$$

p_{zero} is the probability that picking up this task and executing plan $plan_{new}$ will produce zero-marginal costs for future tasks. More precisely, we look at all cities which we are going to visit when following $plan_{new}$ and sum up the task probabilities for all possible "from-to"-combinations on that path. We then normalize the sum and return it. If $p_{zero} > 0.8$, i.e. it is very likely that future tasks can be picked up and delivered within our current plan, we really want to obtain task T. Thus, we bid slightly below our cost. If our marginal costs are smaller or equal to zero, obtaining T would be very useful for us. To make sure we get the task and still receive some payment, we bid slightly below the estimation of the opponent's bid, i.e. $E(bid'_T)$. This value is computed based on a mixture model:

$$E(bid'_T) = \Big(w_{SLS} \cdot bid'_{SLS}(T) + w_{AVG} \cdot bid'_{AVG}(T) + w_{REG} \cdot bid'_{REG}(T) + w_{MEDIAN} \cdot bid'_{MEDIAN}(T) \Big).$$

$bid'_{model}(T)$ is a bid computation using $model \in \{SLS, AVG, MEDIAN, REG\}$. For $model = SLS$, we run the SLS algorithm for the opponent, starting in a random city (since this is not known in advance) and using two vehicles identical to ours. For $model = AVG$, we simply compute the average from the last 10 bids the opponent has made. In the case where $model = MEDIAN$, we take the median of the 10 last bids. Last but not least, for $model = REG$ we perform linear regression over the last 10 opponent's bids. Initially, each component of the weighted sum will receive a uniform weight. For each round, these weights are updated based on the MSE the model introduces. We keep track of the bids that the opponent has made in each round, as well as of our predictions of the opponent for each model. For each model, we compute its MSE and normalize it to obtain the error rate ϵ_{MODEL}. The weight w_{MODEL} is then simply $1 - \epsilon_{MODEL}$ normalized. This allows us to give the model which is closest to the opponent's bids higher weights.

Table 1: Competing against dummy opponents

Opponent type	Our profit	Opponent's profit
Random	9955	2145
Naive-SLS	1313	437
SLS	3800	1344
SLS + Regression	2521	5035
SLS + Average	4049	4562
Regression	4518	1983
Average	2035.4	2551.8

2 Results

2.1 Experiment 1: Comparisons with dummy agents

We implement multiple dummy agents, which we base on the different aspects of our model. We implement one SLS agent, that computes its marginal cost based on the SLS algorithm, however does neither use zero-marginal cost predictions nor makes estimations about the opponent's bids (negative marginal costs will be simply mapped to 0). Additionally we implement a modular agent based on ours which allows us to choose only certain subsets of the ensemble. We report the average profit for our agent vs. the dummy agent averaged for 5 rounds.

2.1.1 Setting

For this experiment, we make use of the England topology and deploy 20 tasks. For our agent, we use an iteration cap of 1500 for the SLS algorithm. If our opponent also uses SLS in this experiment, the iteration cap will be the same. Furthermore, the rejection probability in SLS will be 0.4. We run our agent against each dummy agent 5 times and average the resulting profits.

2.1.2 Observations

What we can clearly see is that our agent beats the naive strategies, ie. random and Naive-SLS. Moreover, the weighted model for estimating the opponent's bids seems to be useful when the opponent only uses SLS. In cases the opponent uses SLS plus another model, it outperforms our agent on average for 20 tasks. The same holds for an agent using only Average. For a Regression opponent, our agent wins.

2.2 Experiment 2: Comparison of varying number of tasks

In this experiment, we want to investigate how the number of available tasks influences the performance of our agent. Since we saw that the performance of our agent is very close to the SLS + Average opponent, we let those compete against each other while varying the numbers of tasks.

2.2.1 Setting

Again, we chose the England topology for out experiment and keep the iteration cap as well as the rejection probability unchanged. We repeat each configuration 5 times and average the results.

2.2.2 Observations

What we can clearly see in Table 2 is that the SLS + AVG agents performs better than our agent when there is a smaller number of tasks.

Table 2: Comparison of varying number of tasks

# tasks	Our profit	Opponent's profit
10	331.6	1132.2
20	4049	4562
30	9467.4	2652.8
40	8557.4	3899.8

Table 3: Competing against dummy opponents

Cut-Off Prob.	Our profit	Opponent's profit
0.5	1403	0
0.6	1263	48
0.7	1695	643
0.8	1939	177

2.3 Experiment 3: Comparison on different Cut-off Probabilities for Zero-Margin Tasks

2.3.1 Setting

We compare multiple values for the parameter we use in the first case of computing our bid (1.7 above). The smaller the value, the less high p_{zero} has to be in order for the factor to be smaller than 1. If this is the case, we bid below our marginal costs. In this experiment, we let our modular agent compete against a naive agent implementing SLS. We run the auction multiple times with each parameter for 20 tasks on the England topology and average the profit.

2.3.2 Observations

We can observe that for a low-probability the agent wins all tasks, preventing the opponent from making any profit. As the parameters increases, the opponent wins more tasks, leading them to make a profit, while the agent wins fewer tasks, that however allow him to maximize his profit, as he does not need to deliver less profitable tasks.