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

Group 20 №: Rafael Madillo, David FernÃ¡ndez Navarro

November 26, 2017

1 Bidding strategy

Our bidding strategy is defined mainly in the function askPrice, helped by actionResult and some others auxiliary functions. Starting by askPrice function, where we return our best bid for a certain task, first thing we do is to compute the bid price based on certain parameters except the ones that includes the opponent. So for this, we create the initial bid based on the marginal cost, which is, the minimum reward we need for having benefit zero. After having this *initialBid*, we apply over it some factors to improve it, one based on the probability distribution of the tasks and another one based on the weight of the task.

- *probabilityFactor:* we check how popular are the cities that are going to be introduced in the path to take and then delivery the task, the more probability of new tasks the more that taking the task worth. So, if the probability of having new tasks is high we apply up to a 15% of reduction of our bid, on the other side, if probability is low we can increment our bid price up to a 15%.

- *factorCapacity:* if the weight of the task is big compared with mean of our vehicle capacity it doesn't worth taking the task because it will probably penalize us in a future, as we won't be able to carry many in the same time, so we tend to penalize heavy tasks.

Once we have the total factor we apply it to the initial bid and then, we try to use all information we have from the opponents to improve even more our bid. Because of not having enough information, only for the first iteration our bid is not modified by the opponents, next iterations do. Until task ten we compute what we think that approximately is going to be our opponents bid for everyone, then only for the best three opponents, which are the ones that have won more tasks. To compute their plans, we imagine a model where they only have one vehicle and we estimate its initial city by their bid for the first task. We create a fictive vehicle for them with our data and this departure city and on every

iteration we compute their best plans if they take the task. With this we get the best possible bid of the opponent and if the difference between this bid and our is less than a 5% we return our bid as a 95% of the previous, else, if we see that our bid is really good and that opponents are far from us we make it greater (a 5%) so that we can earn more money with it.

It's important to know that for computing the best plan, we reuse the SLS algorithm we created for the previous assignment with small changes, such that the initial solution for every new task is the previous best solution.

This was the *askPrice* function, for *actionResult*, what we do is to add the task to the plan of the winner so that after we can compute the plan in the *askPrice* function. Also, as we explained before, for the first iteration, based on the opponents bid, we create their fictive initial plan.

Summarizing,

- We compute our initial bid based on the marginal cost for this bid in our plan by using the SLS algorithm.
- We use probability distribution to improve o penalize bids as we prefer taking the ones that leads us other tasks.
- We compute and save the plan that we think others are doing by using the bid winners and also, for the first iteration, every bid.
- We have defined how much we think each situation must affect the bid so we combine this into a multiply factor and then we multiply initial bid with this factor.

2 Results

2.1 Experiment 1: Comparisons with dummy agents

2.1.1 Setting

run the experiment against a dummy agent, in Switzerland and with 15 tasks. Both agents have the same two vehicles, same capacity and starting city - Lausanne and Bern-.

2.1.2 Observations

shows taht our agent take 14 while the dummy agent 1, we do it with a profit of 289 and the other agent 270, so we won this first round.

2.2 Experiment 2

2.2.1 Setting

We run this experiment with the same configuration as the previous one but modifying one of our parameters, the maximun *probabilityFactyor*, which goes from 0,05 to 0,15, so, because of this, our bids will tend to be greater.

2.2.2 Observations

Result shows that in this situation we win all of the tasks, so for us is 15 and for the dummy agent 0. Because of this, reward for the opponent is 0, while our is 2343. Comparing this with the previous experiment we can see that the *total reward* is 2343 instead of 559, so for this round we have won all bids but also got a better profit, as supposed. This represents the method we have followed for our bid plan, which is taking too many tasks with a really small bid and not leting the opponent take any.

2.3 Experiment 3

2.3.1 Setting

We repeat the same experiment, the 2.1, but this time changing the *capacityFactor* from 0,3 to 0,6 for maximun value, as before, it will make our bids greater.

2.3.2 Observations

For this observation, we see some results similar to the previous ones, much bigger bids, and rewards. For us, total profit is 1824, for dummy, 654, total reward is 2478, this time total reward is bigger but we get less benefits. Strategy is using higher values for our bids but, as before, as we think that opponents will tend to use small bids we will submit our experiment with greater values than before starting this experiment report. Max value for *capacityFactor* is going to be 0,3 while for *probabilityFactor* 0,08, so that our bids are bigger but we think that small enought for betting other intelligent agents.