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

Group №22: Dubuis Samuel, Facklam Olivér

November 22, 2020

1 Bidding strategy

1.1 Marginal Cost

For each task the auction house proposes during the simulation, we use our planner to find the best plan including that task, in order to compute the marginal cost defined as:

$$marginalCost(T, task) = cost(T \cup task) - cost(T)$$

where T is the set of tasks that the agent already has, *task* is the current task being proposed and *cost* represents the cost of the optimal plan.

It is worth noting that we take into consideration the bid timeout parameter when computing the plan to determine this marginal cost.

This technique is also applied to the adversary's state, in order to get a good approximation of their cost. To this end, we model the adversary's vehicles by creating random vehicles similar to ours, since we don't have actual information about the adversary's setup.

1.2 Task Count Heuristic

Generally, the marginal cost of the first tasks is higher than that of later ones. This is because when our agent has few tasks, the marginal cost includes, in addition to the actual delivery cost, some "setup cost" representing the movement to the pickup city for example. However this setup cost will be amortized by later tasks benefiting from this additional movement.

To compensate for this bias, we reduce the marginal cost by a heuristic function based on the number of accepted tasks:

$$\text{TCH}(numOfTask) = \frac{numOfTask + 1}{numOfTask + 2}$$

1.3 Task Distribution Heuristic

As defined in the instructions, we take into account the probability distribution of the tasks to refine our strategy. This multiplicative heuristic influences the marginal cost of a task based on how likely a vehicle has to deliver a later task to its pickup city. This probability is calculated as:

$$\mathbb{P}(c) = \sum_{c' \in C} \frac{1}{n} TD(c', c), \quad \forall c \in C$$

where C is our set of cities and *TD(·)* is the Task Distribution (the probability that a task generated at *c'* will have c as destination).

We defined a specific function to compute this TDH and created a mapping of cities to double values. The previous probability is multiplied by n so as to be centered around 1, and we set a threshold to ensure that the heuristic does not get arbitrarily large (or even infinite). Finally, we rectify this heuristic by raising it to some power so that its importance decreases with the number of tasks already accepted.

$$\mathrm{TDH}(task, numOfTask) = \left(\frac{1}{\max\{n \times \mathbb{P}(task.from), threshold\}}\right)^{\frac{1}{numOfTask+1}}$$

1.4 Connectivity Heuristic

This heuristic is based on how connected the task's pickup and delivery cities are: a better connected task enables lower costs later on. Once again the heuristic is rectified to decrease its importance when the number of tasks increases. With neigh($\cdot$) the set of neighbors of a city, we define:

$$\mathrm{CH}(task, numOfTask) = \left(\frac{averageConnectivity}{|\mathrm{neigh}(task.from)| \times |\mathrm{neigh}(task.to)|}\right)^{\frac{1}{numOfTask+1}}$$

1.5 Biased Marginal Cost

We finally compute the "biased" marginal cost, which combines our basic marginal cost with the various heuristics. It is also worth noting the exponents α and β on TDH and CH respectively in order to give them more or less importance. We used the values $\alpha = 1$ and $\beta = 0.7$.

$$\mathrm{BMC}(T, task) = marginalCost(T, task) \times \mathrm{TCH}(\|T\|) \times \mathrm{TDH}(task, \|T\|)^{\alpha} \times \mathrm{CH}(task, \|T\|)^{\beta}$$

1.6 Adversary's bid

During each auction round, we calculate our estimation of the adversary's marginal cost. After the auction results are announced, we compare their bids with this cost estimation, and determine an "adversary margin" defined as the average difference between the observed bids and the estimated costs. This margin is used to estimate the adversary bid in later rounds:

$$\mathrm{AB} = \mathrm{BMC}_{adversary} + ADVERSARY\_MARGIN\_RATIO \times margin$$

1.7 Final decision

We calculate our biased marginal cost BMC and the adversary's bid AB as described above. As our goal is to beat the other agent, our strategy is to place bids between these two values. We define Δ_{BMC} as the absolute difference between them. In the next equation, the two constants `AGGRESSIVITY_*` define how aggressive we want our agent to be.

$$\begin{cases} bid = \mathrm{BMC} + \Delta_{BMC} * AGGRESSIVITY\_POSITIVE, & \text{if BMC} \leq \text{AB} \\ bid = \mathrm{BMC} + \Delta_{BMC} * AGGRESSIVITY\_NEGATIVE, & \text{if BMC} > \text{AB} \end{cases}$$

1.8 Planning

We generate two different plans after the bidding part is over: one from the variables that we have gradually incremented during bidding, and one made "from scratch" as sometimes this could lead to better results. We take into consideration the plan with the lowest cost.

2 Results

2.1 Experiment 1: Comparisons with dummy agents

2.1.1 Setting

In this experiment we want to observe how our implementation of the agent reacts against the random dummy agent. We run our best agent, as seen in experiment 2, against the dummy agent for 10, 20 and

30 tasks available in the simulation. To show more results, we run the same experiment but on different topologies too. We also set our timeouts to 30s each.

2.1.2 Observations

Topology	10 tasks			20 tasks			30 tasks		
	Tasks won	Profit	Win	Tasks won	Profit	Win	Tasks won	Profit	Win
England	9	6283	Yes	19	19833	Yes	27	22494	Yes
The Netherlands	9	1265	Yes	19	6744	Yes	28	10362	Yes

Table 1: Performance of our agent against dummy adversary for different topologies and number of tasks

As we can see in table 1, our agent performs very well against the dummy agent, winning every round whatever the number of tasks is. It seems to perform better in England than in the Netherlands but this could be because of the topology itself and the cost of tasks.

2.2 Experiment 2

2.2.1 Setting

In this experiment, we analyze the impact of 3 internal parameters on the performance of our agent. `AGGRESSIVITY_POSITIVE` (default: 0.8) controls how high we place our bid between the two costs when our cost is lower than the adversary's. `AGGRESSIVITY_NEGATIVE` (default: -0.4) controls how strongly we undercut our cost when the adversary is cheaper. `ADVERSARY_MARGIN_FACTOR` (default: 0.4) controls to what extent we factor in the adversary's bias empirically measured on the bids.

We consider the default setting (English topology, 20 tasks with random seed 123456). We set our agent against the random agent. The bid timeout is 10s, the plan timeout is 30s.

2.2.2 Observations

The following tables show the results of the 3 parameter studies. The second column shows our agent's profit; the last column is the difference between our profit and that of the dummy adversary.

`AGGRESSIVITY_POSITIVE`	Profit	Advantage
0.5	12628	12628
0.6	14488	14356
0.7	18930	18930
0.8	19990	19858
0.9	18887	18414

Table 2: Agent's performance as a function of positive aggressivity

`AGGRESSIVITY_NEGATIVE`	Profit	Advantage
-0.1	16969	16358
-0.2	18029	17418
-0.3	18415	17804
-0.4	20064	19932
-0.5	13896	13423

Table 3: Agent's performance as a function of negative aggressivity

It would seem that a relatively "aggressive" strategy, strongly overbidding when we know we are cheaper, and decently undercutting when we think we are more expensive, works the best in terms of profit and advantage with respect to the adversary. Given the results, `AGGRESSIVITY_POSITIVE` = 0.8 and `AGGRESSIVITY_NEGATIVE` = -0.4 seem a good combination. The last table shows 2 peeks (0.4 & 1); we will choose the safer alternative by setting `ADVERSARY_MARGIN_FACTOR` = 0.4.

`ADVERSARY_MARGIN_FACTOR`	Profit	Advantage
0	13770	13159
0.2	13678	13067
0.4	17956	17345
0.6	13727	12930
0.8	16077	15280
1	18562	17765

Table 4: Agent's performance as a function of adversary margin factor