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

Group №24: Quentin Juppet, Andrea Piccione

November 22, 2019

1 Bidding strategy

1.1 Assumptions

We suppose that the enemy will have the same starting configuration as we have (same vehicles in same city).

1.2 Estimation algorithm

We use an SLS algorithm to estimate the marginal cost of both the enemy and our agent. At each iteration it saves the previous SLS solution, which it's used to improve its plan and then have a lower marginal cost for the new task. Our SLS algorithm is designed to run as long as the timeout allows in order to have the most accurate information.

1.3 Estimation improvement

Since our estimation may not be accurate (better or worst enemy algorithm), we use the bids of the enemy to compute a relative error and improve our estimation. To avoid noise we compute the error from the last n auctions and do not accept too big/low values.

Moreover since sometime a task could cost nothing to the enemy (already on its path), we assume that he would still try to gain at least the cost between the pickup and the delivery city.

We also prevent our enemy estimation to be too low by setting a minimum bid that start from a defined value and stay to the minimal value that the enemy actually bid.

1.4 Bid strategy

Our strategy is based on 3 main points:

- If our bid is lower than the enemy bid, we increase it to earn some money by a factor, we keep a reasonable margin from the enemy bid estimation. This factor is bigger for the first n task since the estimation should be less accurate at this time.
- If our bid is bigger than the enemy bid, we check if we can afford to reduce our bid and then lose some money to better challenge the enemy expected bid. We afford to loose money only if after the loss we still have more profit than the enemy. If we can't afford to loose money, we accept to reduce our bid by some risk factor.
- If the task interest is above some threshold, we reduce the bid by some factor (for both our agent and the enemy). We compute the task interest by looking at the maximum probability to have a task between one of our already earned task and the new one (from delivery to pickup city).

2 Results

2.1 Experiment 1: Comparisons with dummy agents

2.1.1 Setting

In this experiment we tested our optimal agent against other simple agents. The dummy agent is the one provided in the template file, the constant agent is an agent that always offers the same constant bid at every round, instead the cheater agent is the one which always bids the last winning bid from the previous round. We used a tournament configuration with 10 tasks, 2 vehicles for each competitor and we tested this in the England topology. The timeouts have been all set to 10 seconds for simplicity.

Agents	Win - Draw - Lose	agent	dummy	constant	cheater
agent	6 - 0 - 0	-	WIN (4148 : 473)	WIN (-2006 : -10866)	WIN (-2119 : -14139)
dummy	4 - 0 - 2	LOSE (0 : 4452)	-	WIN (0 : -14055)	WIN (0 : -14055)
constant	1 - 0 - 5	LOSE (-10202 : -1692)	LOSE (-13391 : 0)	-	LOSE (-13391 : 0)
cheater	1 - 0 - 5	LOSE (-10202 : -2056)	LOSE (-13391 : 0)	LOSE (-13391 : 0)	-

Figure 1: Scores for tournament with dummy agents in the England topology.

2.1.2 Observations

Figure 1 reports the results of the two tournaments. As we can see, our agent always beats the other agents with a significant margin. Clearly, this is not a surprise but it's interesting to see how the agent adapts its strategy according to not-conventional agents. By running some sample matches between our agent and the other dummy agents we can also notice that almost all rounds are won by our agent which seems to bid with a reasonably good strategy and varies the bid according to the amount of profit it already has. Dealing with dummy agents is not trivial because it's difficult to predict their behaviour by estimating their bid using an optimal plan computation. Indeed, when our agent deals with some weird and unpredictable agents like the constant and the cheater one, we can see that the profit is negative but this is normal because we try to adapt ourselves to the opponent's behaviour since we assume that the other agent is using an optimal strategy like we are. This can lead to sub-optimal solution but since the goal is to beat the opponent we believe this is the correct strategy to use.

2.2 Experiment 2: Comparison with another intelligent agent

Agents	Win - Draw - Lose	agent	agent_old	dummy
agent	4 - 0 - 0	-	WIN (2136 : 1010)	WIN (13490 : 614)
agent_old	2 - 0 - 2	LOSE (-333 : 1693)	-	WIN (11878 : 473)
dummy	0 - 0 - 4	LOSE (77 : 2344)	LOSE (784 : 2396)	-

Figure 2: Scores for tournament with dummy agent and old agent in the England topology.

2.2.1 Setting

In this experiment, we use the same scenario as in the previous experiment by testing our agent against another one which is a previous version of the final agent we presented in the first section. This agent similarly tries to estimate the marginal cost by computing an optimal plan at each round but it doesn't change the offer according to the feedback of the previous rounds, it just tries to optimize the offer based on latest bids but no further improvements are used. The parameters are the same of the previous experiment.

2.2.2 Observations

As shown in Figure 2, even this time our final agent beat all the others. The most significant aspect is not the fact that it beats the old agent but the fact that it beats both the old agent and the dummy with a much higher profit. It seems like our final agent has a much better strategy than before even though the plan computation is almost the same as the old agent.

2.3 Experiment 3: Varying parameters

2.3.1 Setting

We decided to test our optimal agent with different parameters in order to see possible different bidding behaviours in different situations. For this experiment, we used the England topology, with different values for timeouts, number of tasks to be auctioned and the number of vehicles at disposal of each agent. To keep things simple, we assumed that the timeouts for setup, bid and computing a plan are the same. The results, reported in Table 1, have been obtained by running a tournament against the dummy agent provided in the template file.

Timeout	10 tasks		20 tasks	
	2 vehicles	4 vehicles	2 vehicles	4 vehicles
5s	4342	2115	4357	8674
10s	4347	2243	8302	6891
15s	4356	2167	10049	9572

Table 1: Mean profit for different tournament configurations against dummy agent.

2.3.2 Observations

As shown in Table 1, we can see that the different values of the parameters don't change too much the behaviour of our agent. We expected that an higher timeout could improve the profit since the estimation of marginal costs could be more accurate but this gave some sort of impact only when 20 task were used. The results for 10 tasks were always the same for different values of timeouts: in this case, it looks like having more vehicles actually decreases the profit, which could seem counter-intuitive at first but we know that the centralized agent can have such a behaviour and usually relies only one agent to carry all the tasks when there are few tasks available. In fact, when having more tasks to carry, we see there are some few differences: the profit seems to increase a bit when using more agents and also when using an higher timeout we can see there are some improvements in the mean profit. One possible reason could be that computing a more optimal plan with more tasks needs more time and so the more the time available to compute the marginal cost and the plan, the more accurate the estimations and better the outcome.