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

Group 51: Liangwei CHEN, Siyuan LI

November 24, 2019

1 Bidding strategy

Task distribution exploitation For simplicity of the algorithm our implementation of bidding strategy does not take task distribution into account.

Planning To construct a bidding strategy, we first need to evaluate the marginal cost of our plan. Since an agent has multiple vehicles, the distribution of tasks problem can be formulated as the centralized planning problem which have already been resolved in the previous exercise. Thus the stochastic local search (SLS) algorithm has been exploited to search for an optimal task distribution. Specifically for each assignment, we find the neighbours by relocating tasks among vehicles and rearranging task order of one specific vehicle. Then local search algorithm has been utilized to decide the next choice from moving to an optimal neighbour, moving to a random neighbour and staying at current assignment. The estimated optimal marginal cost is then utilized in our bidding algorithm to decide the bid returned to the auction house.

Bidding Strategy Since the winner of each auction round and others' bids are given by the *auctionResult* function, our implementation estimate their bids based on these information from previous auctions. Specifically, we evaluate the estimated marginal cost of other agents using the same SLS algorithm. By utilizing the winner information of previous round, we have already obtained the list of tasks that the other agents have to deliver. Then by estimating the optimal task assignments and calculating the marginal cost by taking current task into account, we obtain an estimate of the opponents' marginal cost. Then our strategy is defined by the following formula:

$$initialBid = max(selfMarginalCost * selfBidRatio, oppMarginalCost * oppBidRatio)$$

$$decidedBid = \begin{cases} max(oppMinBid - 1, 1), & \text{if } initialBid \leq oppMinBid \\ initialBid * 0.5, & \text{if } round \leq 4 \\ initialBid & \text{if } else \end{cases}$$

The $selfBidRatio$ and $oppBidRatio$ are two ratio deciding the initial estimated bid of self and opponents based on the estimated marginal cost. These ratios has range among $[0.75, 0.9]$ and $[0.8, 0.95]$ and are adjusted during auctions using multiplicative update rule. Specifically, an agent winning an auction tends to be more aggressive and will propose higher bid in next round, on the other hand, an agent losing an auction may tend to be more conservative and will propose a lower bid in next round.
The first case of *decidedBid* is used to handle the freeride task bidding. Considering a new task that can be delivered without making further move thanks to previous delivering plan, the estimated marginal cost of this task may be zero. However, to prevent delivering without making making any profit, we instead keep a history of the opponent's lowest bid and set our bid for this "costless" task to be a value

just lower than the opponents' lowest bid. The intuition is that the opponent's lowest bid may capture the opponent's strategy on bidding for "costless" task. Thus by providing a profitable bit just lower than the opponent's proposal, our agent can win the auction while making the maximal possible profit.
The second case of *decidedBid* is to quickly "warmup" the agent. Intuitively an agent should grab more tasks in the first few auctions to provide more possible choices in the following auctions. Thus we decided to be more aggressive in the first 5 round in the sense that we set the *decidedBid* to be half of *initialBid* to increase our possibility in winning the first few auctions.

2 Results

2.1 Experiment 1: Comparisons with dummy agents

2.1.1 Setting

The dummy agents are bestResponse, the agent which obtained the second place in 2015/2016 semester, honest agent and naive agent. Specifically their definitions are:

- BestResponse: An agent that tries to obtain at least Δ profit where delta is a predefined threshold. It will set its bid to *estimatedOpponentBid* if that will bring even more profit than predefined threshold. Namely,

$$bid = max(self.estimatedMarginalCost + \Delta, estimatedOppMarginalCost)$$

- BenchMark Agent: The agent which obtained the second place in 2015/16 semester, which we referred to during construction of our plan and set as a baseline.

- HonestAgent: An agent that only bid for its estimated cost added by a predefined threshold. This agent tries to get a fixed amount of profit from each bid. Namely,

$$bid = self.estimatedMarginalCost + \Delta$$

- NaiveAgent: An agent that bid at a random ratio times the estimated marginal cost but assign all the tasks to the first agent.

2.1.2 Observations

Agents	Win - Draw - Lose	Ours	DummyAgent1	DummyAgent2	DummyAgent3	DummyAgent4
Ours	7 - 0 - 1	-	WIN (7721 : 1893)	WIN (24684 : 2409)	WIN (24983 : 2409)	WIN (27950 : 665)
DummyAgent1	7 - 0 - 1	WIN (10364 : 886)	-	WIN (24931 : 2409)	WIN (21883 : 2409)	WIN (23983 : 665)
DummyAgent2	3 - 0 - 5	LOSE (2208 : 24387)	LOSE (3368 : 21341)	-	WIN (27546 : 12409)	WIN (49492 : 614)
DummyAgent3	3 - 0 - 5	LOSE (2218 : 16987)	LOSE (3378 : 24085)	WIN (22189 : 11084)	-	WIN (24384 : 614)
DummyAgent4	0 - 0 - 8	LOSE (329 : 37278)	LOSE (329 : 38741)	LOSE (29 : 34569)	LOSE (29 : 25243)	-

Table 1: Compare with Dummy Agents in environment 1

Agents	Win - Draw - Lose	Ours	DummyAgent1	DummyAgent2	DummyAgent3	DummyAgent4 ,
Ours	6 - 0 - 2	-	WIN (1341 : 547)	LOSE (456 : 3359)	WIN (6197 : 1185)	WIN (6378 : 223)
DummyAgent1	5 - 0 - 3	LOSE (463 : 572)	-	LOSE (656 : 4635)	WIN (6860 : 1185)	WIN (7219 : 223)
DummyAgent2	5 - 0 - 3	LOSE (3100 : 3341)	LOSE (3263 : 5531)	-	WIN (8887 : 8243)	WIN (11341 : 28)
DummyAgent3	4 - 0 - 4	WIN (3657 : 3404)	LOSE (2748 : 6570)	WIN (12875 : 8025)	-	WIN (20916 : 28)
DummyAgent4	0 - 0 - 8	LOSE (137 : 7358)	LOSE (137 : 7979)	LOSE (100 : 11884)	LOSE (1083 : 18398)	-

Table 2: Compare with Dummy Agents in environment 2

The aforementioned two groups of experiments were conducted in different auctions with distinct number of tasks, vehicle capacity and topology. The results indicated that our agent not only outperforms

the trivial agents such as naive one and honest one, but also achieve better performance in competition against some more complex agents including the best response one and the benchmark. The advantage against the complex agents originates from our design that an moving average of a estimatedRatio has been computed. This ratio serves as an estimate of how much risk an agent tends to suffer so that it will provide a bid denoted by $ratio * estimatedMarginalCost$. This estimate enables our agent to be aware of the opponent's strategy to some extent and thus be able to outperform them in both two settings. ⋮

2.2 Experiment 2

2.2.1 Setting

In the experiment 2, we explore different variants based on our strategy. By twisting our parameters, we create three different variants: Greedy, Conservative and No quick-start.

- Greedy: This agent tries to obtain more tasks than our agent. Namely, it set the bid to be $max(ratio * self.estimatedMarginalCost, oppRatio * estimatedOppMarginalCost)$ s.t. $ratio$ has the range between $[0.5, 0.7]$.
- No quick-start: This agent does not has the quick start phase as our agent. Namely it will not attempt more to obtain tasks at the beginning phase.
- Conservative: This agent does not take the risk of losing at each step. Namely the ratio it multiplies on the $estimatedMarginalCost$ not less than 1.

2.2.2 Observations

Agents	Win - Draw - Lose	Greedy	Ours	Conservative	NoQuickStart
Greedy	5 - 0 - 1	-	WIN (2146 : 432)	WIN (6776 : 643)	WIN (10035 : 903)
Ours	4 - 0 - 2	LOSE (-719 : 1846)	-	WIN (21548 : 752)	WIN (8912 : 3805)
Conservative	2 - 0 - 4	WIN (-564 : -2872)	LOSE (6398 : 7347)	-	WIN (10079 : 4699)
NoQuickStart	1 - 0 - 5	LOSE (1125 : 7821)	LOSE (5140 : 5718)	WIN (11553 : 2487)	-

Table 3: Compare with variants in the environment that has 30 available tasks

Agents	Win - Draw - Lose	Ours	Conservative	Greedy	NoQuickStart
Ours	6 - 0 - 0	-	WIN (2125 : 57)	WIN (-1260 : -2168)	WIN (3178 : 744)
Conservative	3 - 0 - 3	LOSE (-1260 : 2152)	-	WIN (-1630 : -2255)	WIN (2521 : 916)
Greedy	2 - 0 - 4	LOSE (-1007 : -54)	LOSE (-480 : 57)	-	WIN (1970 : 63)
NoQuickStart	1 - 0 - 5	LOSE (-1048 : 1261)	WIN (485 : -1605)	LOSE (-288 : 1580)	-

Table 4: Compare with variants in the environment that has 10 available tasks

From the first experiment one may observe that our agent outperforms the conservative agent and agent without quick start. However, it loses to the greedy agent. By investigation we find out this is due to large number of available tasks. Namely in aforementioned situation, an agent obtaining more tasks will have higher probability to deliver task without extra work in future. Following the same deduction, we infer that the greedy agent will not perform so well in situation with less tasks. The second experiment confirms our intuition. Since we do not know the number of tasks in advance, our agent is more robust and reliable than greedy one.