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

Group №11: Maxime Gardoni, Marcel Dubach

November 22, 2020

1 Bidding strategy

The decentralized agent presented here can take into account the task distribution to try to predict likely future scenarios, and can learn from it's opponent past bid in order to adapt it's profit margin on the fly.

1.1 Marginal cost

At the start of the auction, the agent generates $n_{Scenario}$ of possible likely futures. For each of those scenario, a number h (horizon) potential future tasks is randomly generated according to the probability distribution function of the task appearance.
The agent will keep these generated scenario trough the bidding process, always adding them to S_1 (the set of accepted task) and S_2 (the set of accepted task and the currently auctioned task). By optimizing of these two set via SLS, then taking the difference cost(S_2)-cost(S_1), the agent can deduce a marginal cost for each Scenario. Averaging over all the scenarios, the estimated marginal cost is obtained.

1.1.1 SLS Optimizer Warmup

The optimized solutions found for each scenario are stored and reused as initial value for the next iteration. This warmup-strategy allows faster convergence and better marginal cost estimations.

1.2 Bid

Once the agent has an estimation of the marginal cost of the current tasks, it makes a bid according to the following strategy:

$$bid_{Agent}(T_k) = \mathrm{MC}(T_k) + \mathrm{PM}(t_k) \tag{1}$$

where $MC(T_k)$ is the marginal cost of the task T_k and $\mathrm{PM}(t_k)$ is the profit margin at time t_k (t_k corresponding to the time when task T_k ia auctioned). The profit margin is updated depending on outcomes of previous bids as follows:

$$\mathrm{PM}(t_k) = \mathrm{PM}(t_{k-1}) + a \cdot \Delta(t_k) \tag{2}$$

where $\Delta = bid_{Opp}(T_{k-1}) - bid_{Agent}(T_{k-1})$ depends of the bid of the agent itself (bid_{Agent}) and the bid of the opponent (bid_{Opp}) for the previously auctioned task. a is a parameter in [0,1] corresponding to the adaptation ratio. This parameter defines how much our agent adapts to its opponent, and the parameter needs to be tuned (see experiences). This implies that if the agent won the tasks, then $\Delta > 0$ and the agent tries to increase its profit margin. Else, the agent tries to decrease its profit margin. The profit margin is initialized to zero in order to win auctioned tasks quiskly, since having more tasks may lead to smaller marginal costs for potential future tasks in a lot of cases. The profit margin is constrained to be positive, in order to never deliver a task at loss (no risk taken).

The only case in which the agent denies the task (= bid set to null) is when the agent has no vehicle that has enough capacity to deliver the task.

2 Results

2.1 Experiment 1: Comparisons with dummy agents

2.1.1 Setting

In this experiment, 15 tasks are generated, and the agent *Main* is tested against 2 other agents:

- *Random* : the agent provided in the skeleton ; it uses only 1 vehicle, and adds sequentially new task at the end of it's current plan.

  The marginal cost is then estimated from this plan, and a bid is created by multiplying the marginal cost by a random number between 1 and 2. The final plan is computed using the same naive and sequential method.

- *SemiRandom* : The marginal cost estimation and bidding part are the same as *Random*, but at the end, it computes an efficient centralized plan using SLS, which uses all vehicles.

2.1.2 Observations

Agents	**Win - Draw - Lose**	**Main**	**SemiRandom**	**Random**
Main	4 - 0 - 0	-	WIN (12461 : 1926)	WIN (11575 : 1367)
SemiRandom	2 - 0 - 2	LOSE (3937 : 8758)	-	WIN (17462 : 1245)
Random	0 - 0 - 4	LOSE (701 : 11878)	LOSE (2044 : 4161)	-

Table 1: Agents tournaments results

As expected, SemiRandom beats Random, due to the fact that an efficient final plan has a big impact on the profit (an optimized plan is cheaper than a sequential plan, especially when the number of task is high)

According to expectations, Main beats Random and SemiRandom. It's due to it's ability to compute more accurate marginal cost, and due to it's ability to observe the opponent bid in order to adjust it's profit margin on the fly.

2.2 Experiment 2: Tuning of parameters

The second experience describe the tuning process of the main agent. The changing parameters are the following:

1. $t_{Scenario}$, the time that the SLS algorithm has to compute a solution for each Scenario of potential future tasks
2. h, the horizon, i.e. the number of potential future tasks that are generates
3. a, the adaptation gain that adapts the profit margin depending on bids from the opponent

We tune the parameters through simulation of an auction between our agent at itself with different parameters.

2.2.1 Tuning of the optimization time

In this section, we discuss the optimization time allowed per scenario (an agents typically has multiple scenarios to compute, representing multiple possible future tasks that are auctioned). There is a trade-off between the time to optimize for a given scenario and the number of scenarios. Having more time to optimize a scenario leads to a better solution for that scenario, but since it takes longer, the are a smaller number of scenarios that can be computed in the same time. Having more scenarios to look at may lead to a better estimation of the marginal cost, if the optimization of all scenarios comes close enough to the optimal solution.

2.2.2 Setting, Observations

We simulate the following experience for an optimization time per scenario $t_{Scenario} \in [5s, 10s, 20s]$. The auction parameters are: 15 Tasks to be auctioned, 40 seconds to calculate a bid.

The performances of the agents were found to be in the following order: The agent with the longest optimization time per scenario (and thus the smallest number of scenarios) was clearly outperformed by the other agents. The agent with $t_{Scenario} = 10$s again was outperformed by the agent with the lowest optimization time per scenario. We conclude that it is more beneficial to have a higher number of scenarios that optimize over a smaller amount of time than the inverse. Thanks to the warmup initialization we may use a small optimization time for each scenario ($t_{Scenario}$=5s) to have a reasonably good solution for each scenario.

2.2.3 Choice of the horizon

Adding more potential future task may lead to a better prediction of the marginal cost, since there may be a relatively high number of tasks auctioned. However, the computational cost to find good solutions for more potential future tasks increases and might slow down the optimization process.

2.2.4 Setting, Observations

We compare different values for the horizon: $h \in \{2, 4, 8\}$. Auctions are launched with 15 Tasks and $t = 30$s to calculate each bid.
The agent with a horizon of 8 performed bad in direct comparison, probably because the optimization could not find a good solution in reasonable time. Surprisingly, the agents with horizon of 2 and 4 lead to similar performances. As we prefer an agent with a higher horizon, we choose to retain a horizon of 4.

2.2.5 Adaptation of the profit margin

Our agent adapts its profit margin according to the difference of the bid of the opponent and its own bid. If he won, it will try to maximize profit and increase the profit margin. If he lost, he will decrease profit margin and try to win next bid. The adaptation of the profit margin is based on the difference between his bid and opponent's one, multiplied by a weight a.

2.2.6 Setting, observations

We compare an agent with an adaptation ratio $a = 0.1$ against another agent with $a = 0.1$. Using the same settings as previously (15 Tasks, 40 seconds to make a bit).

The agent with an adaptation ratio $a = 0.1$ performs better in direct comparison with the other agent. We retain that solution also because a low adaptation ratio will lead to a more consistent bid when the agent competes against a dummy agent making random bids.

We selected the following parameters for our final main agent: $t_{Scenario} = 5$s, a=0.1 and $horizon = 4$.