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

Group №46: Yann Vonlanthen, Loïc Vandenberghe

November 24, 2019

1 Bidding strategy

1.1 Overall strategy

As we created many different auctioning agents we realized two major insights. First picking up the first tasks can be very advantageous, and second, an agents performance depends a lot on the opponent it is facing. This prompted us to distinguish two phases in an agents behavior that are described below.

The common mechanism between both phases is the underlying centralized planner using stochastic local search, which for each task to be auctioned tries to find the best possible plan in our current setting. Crucially, to aid the SLS algorithm we find a good initial solution by inserting the new task t at the best possible spot in our current plan. The cost of this new plan minus the cost of the old plan results in the *marginal cost* of accepting task t. The following phases and strategies will always compute a bid using the marginal cost of an agent as cornerstone.

1.2 Phase 1: Expected savings

In this phase the goal is to try to determine the *utility* of a given new task t, in order to know how much our agent can afford to bid at a deficit, i.e. below the marginal cost computed.

Intuitively if task t has a low weight but takes us through very common edges in the graph, there is a high chance that at the next auction we can offer more competitive prices than our opponents, as we can transport multiple tasks for some part of the way. In the following we consider the underlying probability distribution to compute the *expected savings* $S[t, p, n]$, a metric that will serve us to speculate about the utility of Task t, given current plan p and the auction number n.

$$S[t, p, n] = min_{v \in V} \left[\sum_{m \in t} \left(\mathbb{E}[S(m) \mid load(v, m)] \right) \right] * (1 - \mathbb{P}(\text{numberOfAuctions} \leq n))$$

$$\mathbb{E}[S(m)|load(v, m)] = len(m) * cost(v) * \sum_{t' \in T} \left(\mathbb{P}(t') * \mathbb{I}(m \in t') * \mathbb{I}(\mathbb{E}[weight(t')] + load(v, m) < capacity(v)) \right)$$

Where $\mathbb{P}(\text{numberOfAuctions} \leq n)$ is the probability that there are no more auctions following and thus no more savings to be made. This probability is defined to be distributed according to a Poisson distribution with the expected number of auctions being 10. In phase 1 we thus have:

$$bid = marginalCost - d * S[t, p, n]$$

Note that this metric is imprecise as an agent does not necessarily need to take the shortest path to deliver a task. Moreover for ease of implementation and performance reasons we compute savings only one step ahead. Nevertheless this gives us a good indication, much better than other methods we tried, such as running pageRank on the Graph vertices to determine their utility.

1.3 Phase 2: Multiplicative weight update method

In this phase we want to become profitable, and outperform our opponents. We do so by always bidding above our marginal cost, but as close as possible below our opponents marginal cost. Since our opponents strategy can't be predicted we found it hard to choose an expert that performs well against all agents we implemented. We thus chose to listen to many agents (called experts in the following) and bid a weighted average of their bid proposals. Each expert has its own strategy that might include approximating the opponents behavior, and a weight that is adjusted according to its ability to bid high but below the opponents bid.

More formally let $\mathbf{w}^t = (w_1, w_2, \ldots, w_n) \in [0,1]^n$ be a vector such that: $\sum_{i=1}^n w_i = 1$. Now let $\mathbf{b}^t = (b_1, b_2, \ldots, b_n) \in \mathbb{N}^n$ be a vector of the bids for each expert for task t.

The bid of our main agent for each task would be: $\mathbf{w}^t(\mathbf{b}^t)^T$.

Now the important part is how these weights are updated after we get the result of an auction. Let $\mathbf{r}^t \in \mathbb{N}^n$ be the rewards of each expert with the bid they did on task t. (i.e $r_i^t = b_i - marginalCost$ if $marginalCost \leq b_i \leq opponentBid$, $r_i^t = 0$ otherwise). The weights are updated as follows:

$$w_i^{t+1} = w_i^t * (1 + \lambda \frac{r_i^t}{max_{bid}(reward(bid))})$$

1.4 Experts

The very general setting described above allowed us to create experts very easily. Additionally we can create *Decorator*-experts that can add a specific behavior on top of one or multiple existing experts. Most of them were used for testing purposes, but in the following we describe some experts that we use in our best performing agent.

1.4.1 Ratio expert

$$bid = (marginalCost + tax) * ratio(t)$$

$$ratio(t+1) = \begin{cases} ratio(t) * 1.1 & \textit{if bid at time t was won} \\ ratio(t) * 0.8 & \textit{otherwise} \end{cases}$$

With $ratio(0) = 1$ and $tax = 2$ being an empirically chosen constant that avoids us to bid 0 when our marginal cost is 0.

1.4.2 Adaptive expert

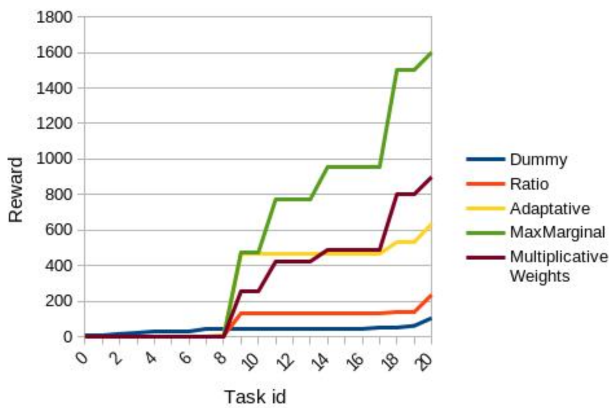
The adaptive expert tries to estimate the next bid of the opponent. It assumes that the opponent will always bid its marginal cost times a ratio (similarly to the ratio expert). It starts assuming the opponent ratio (r') is 1. The adaptive expert bids the maximum between the $ratioExpertBid$ and $r' * opponentMarginalCost * secureFactor$. The $secureFactor$ is chosen to be around 0.9 since the other factors represent the agent's approximation of the opponent bid, and we want to maximize our profit by bidding as close as possible to its bid.

After getting an auction result, we update $r' = r' * 0.5 + 0.5\frac{opponentBid}{opponentMarginalCost}$

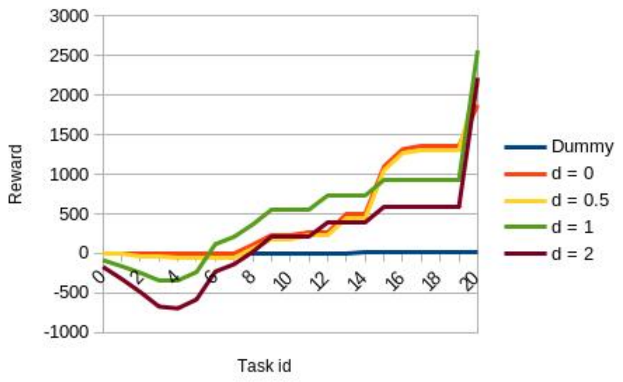
1.4.3 Max Marginal Expert

$$bid = max(marginalCost, marginalOpponentCost)$$

This expert fully trusts the estimated opponent marginal cost to be a (pretty tight) lower-bound on the opponents cost, but never takes a task at a deficit.



(a) Reward against a dummy agent



(b) Effect of discount factor against a dummy agent

2 Results

2.1 Experiment 1: Comparisons with baseline agent

2.1.1 Setting

In this setting we run a dummy agent that always bids its marginal cost plus a random value between 0 and 10. We used the England topology with 20 tasks, both company having 2 vehicles with the same capacity and cost. We show only the results were the dummy agent is advantaged with its home city (i.e the configuration were the dummy agent performs the best). The dummy agent will perform against 4 different strategies: the ratio expert, the adaptive expert, the max marginal expert, and the multiplicative weight agent that uses these 3 expert and a $\lambda = 1.0$.

2.1.2 Observations

We can see in figure 1a that multiplicative weight follows as expected the best of its experts. Against the dummy agent, the expert performing the best is ”MaxMarginal”. After task 10, our agent follows the strategies of experts ”Adaptative” and ”MaxMarginal” with greater interest than the expert ”Ratio”. Finally, since MaxMarginal gets clearly better results, our agent follows mainly the ”MaxMarginal” expert. In conclusion, our agent is not the one that would perform the best against any given opponent, however it will have some performance guarantees to approach the performance of the best of its experts against that opponent.

2.2 Experiment 2: Phase 1 Discount factor

2.2.1 Setting

We run our best agent described above with different discount factors, i.e. we test how aggressive an agent can be in Phase 1 while still making profits after the expected number of tasks. We use the *netherlands.xml* topology, with low timeouts (10 seconds for bidding and planning). Apart from the starting positions the setting is fair. The adversary implements the (very tight) strategy $bid = marginalCost + 1$.

2.2.2 Observations

Even though the experiment is probabilistic in nature Figure 1b confirms our intuition. Note the large increase in reward at the end due to the additional planning performed. We observe that the agent with discount factor 1 wins over all other agents after planning, as it picked up more tasks compared to agents with lower discount factors and didn't loose as much money in Phase 1 as the agent with $d = 2$. Finally note also that all agents were able to become profitable by the end of round 8, which validates our underlying probability distribution.