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

Group №4: Adrian Valente, Bruno Wicht

November 26, 2017

1 Bidding strategy

Our bidding strategy essentially tries to anticipate over future tasks in order to offer small bids at first (lower than the marginal cost) if we hope to make for this loss later. To do this let us recall that the marginal cost MC of a task t is defined as the cost of the best found solution with all previous tasks and t, minus the cost of the best found solution without task t.

We can decompose the marginal cost as follows:

$$MC = TC + OC$$

where the **transport cost** $TC = d(pickupCity, deliveryCity)$ is the inherent cost of task t, and the **overhead cost** OC is caused by the situation of the vehicles and the other tasks. This overhead can be positive (which will be the case for the first task), or negative if for example task t happens to be on the way of getting another task that we already had.

If the overhead cost is positive, we estimate the probability that a "good task" will come up in the future on the way to this pickup city. Thanks to this we will be able to bid under the marginal cost. Formally, let $\{c_i\}$ be the set of cities where our vehicles end up in the solution without t, and let A be the event that for some c_i and some future instant, a task from c_i to the new pickup city $pickup(t)$ will come up. The probability of A is given by:

$$P(A) = 1 - (1 - \sum_{c_i} P(task(c_i, pickup(t))))^{remainingTasks} \quad (1)$$

We know that if A happens, we will make up for our money, and if not we might loose the difference between our bid and the marginal cost. This gives us a lower bound on the expectation of the gain we will make:

$$E(gain) \geq (bid - TC)P(A) + (bid - MC)P(A^C)$$

If we want this expectation to be positive, we get:

$$bid \geq TC \cdot P(A) + MC \cdot (1 - P(A))$$

Which gives us a bid between the transport cost and the marginal cost. In order to make up for our debts, it suffices to then bid $Bid = TC$ when the marginal is lower than the transport cost.

The only catch is that we cannot compute (**??**) since we do not know the number of remaining tasks. So we will use the lower bound on the number of tasks given in the FAQ *ie* 20.

Finally we still want to exploit the history of bids of our adversaries in order to win as much money as possible. In order to do this, we multiply our *bid* computed as above by a *ratio*, wich will be updated by the formula:

$$ratio = \alpha \cdot \frac{bestBid}{bid} + (1 - \alpha) \cdot ratio$$

Where *bestBid* is the best bid made among our adversaries on the last round. Thus the algorithm is constantly trying to improve its ratio.

Finally let us add a few final notes:

- We use the ratio only in the case of a positive overhead. In the case of a negative one, we simply bid the transport cost.
- To protect against pathological cases, we force the ratio to stay in a hardcoded interval
- When the twientieth task is reached, we switch to a more simple regime where we simply bid $ratio * MC$ for all tasks.

1.1 A note on the optimization algorithm

The optimization algorithm that was chosen is **random restart local search**, which we found to be very robust under different timeout settings. Indeed, we made some attempts at using simulated annealing, but found that the hyperparameters of this algorithm were difficult to optimize, and very sensible to the timeout value.

2 Results

2.1 Experiment 1: Comparisons with dummy agents

2.1.1 Setting

We first tested our agent against the auction-random agent provided for this exercise, in the english topology, for a number of tasks varying between 20 and 40 and different seeds. This also enabled us to improve the agent.

2.1.2 Observations

In its final version, we were glad to observe that our agent was able to beat the random one in all situations, with a very significant margin. In fact, the random agent is seldom able to win a few bids, letting almost all of them to our agent. However, in our first tests, this was not always the case, partly due to some aberrant behaviour for the ratio. We have thus stabilized the ratio by limiting it to a specific range.

2.2 Experiment 2 : the usefulness of the distribution

2.2.1 Setting

In order to check for the usefulness of our exploitation of the task distribution, we made our agent compete the same agent (called agent-nostrat) except that it does distinguish between positive and negative overheads. More precisely, agent-nostrat uses a ratio, updated with the same rule, and always bids $ratio \cdot MC$.

2.2.2 Observations

We were happy to see that our agent always wins, although some races are tight. We had doubts about the behavior of our agent when the overhead is negative: indeed, while the marginal cost can be 0, our agent always bids the transport cost of the task. This strategy makes sense while the adversary has not won a single bid: indeed, he can certainly not have a marginal cost lower than the transport cost for his first task. However, after a while, many bids are lost to this strategy.

This observation pushed us to switch strategies after the twientieth task. Thanks to this, our agent is able to "rob" a lot of tasks at the beginning by bidding under the marginal cost, and at the end of the game when most tasks have a low marginal cost switch to a standard strategy which takes advantage of it. An additional positive effect of our strategy, is that it enables us to crush the ratio of our adversary at the beginning, thus reducing its subsequent gains.