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

Group №21: Dubravka Kutlešić, Nevena Drešević

November 22, 2020

1 Bidding strategy

In this exercise, we use the same representation of the solution as in the previous one. All the variables, models, and constraints remain the same, we only add additional ones. To create a bidding strategy for each task 3 factors (described in the following 3 subsections) and their relations are considered.

1.1 Feedback from the previous auctions

For each competitor, the approximations for its future bids are stored in the form of a matrix with dimensions $n \times n$, where n is the number of cities in the topology. An entry $[i, j]$ in the matrix represents how much money the bidder would bid for a task with the pick-up city i and the delivery city j. In the beginning, all entries are *null*, and the entries are updated after every bid.

For a task t, $t.p$ and $t.d$ are denoted as the pickup and delivery city of the task t, respectively. Depending on the competitor's result in the previous auction while bidding for task t, the entry $[t.p, t.d]$ is updated as following:

$$[t.p, t.d] = \begin{cases} bid\ that\ the\ competitor\ placed,\ if\ the\ competitor\ did\ not\ win\ in\ the\ auction \\ 0,\ if\ the\ competitor\ won\ the\ auction \\ 10\ \cdot maximal\ marginal\ cost\ for\ the\ agent,\ if\ the\ competitor\ placed\ null \end{cases}$$

For the competitors, cost per km of vehicles is approximated as the average of costs per km for the agent. That cost is denoted as $costKm$. The city i is ε-close to the city j iff $distance(i, j) < \varepsilon \cdot \max distance_{c_1,c_2 \in Cities}(c_1, c_2)$, where ε is config parameter. After the bid for task t, the entries $[c_p, t.d]$, $[t.p, c_d]$, where c_p is ε-close city to $t.p$ and c_d is ε-close city to $t.d$, might be updated. If $[t.p, t.d] + distance(c_p, t.p) \cdot costKm$ lowers the current entry for $[c_p, t.d]$ (or $[c_p, t.d] = null$), the update is done. Symmetrically, the same approach is done for $[t.p, c_d]$ entries. Here, we approximate that a competitor would bid similarly for the future tasks that are "similar" to task t. Also, the agent is aware that a competitor wins a task and that "similar" tasks will be payable to the competitor in the future.

1.2 Probability distribution

Assume that the agent already won tasks $T = \{t_1, t_2..t_n\}$ in the previous rounds and that owns vehicles with home cities from the set HC. Next assume that the task t is offered on the auction. The speculated probability is calculated as:

$$p = \max\{\max_{\widetilde{t} \in T} distribution(\widetilde{t}.d, t.p), \max_{\widetilde{t} \in T} distribution(t.d, \widetilde{t}.p), \max_{c \in HC} distribution(c, t.p)\}$$

where $distribution(x, y)$ denotes the conditional probability that in the town x appears the task with delivery city y. The probability p represents how likely it is to have a task t_f in future auctions that perfectly fits to the chains $\widetilde{t}.d \to t_f \to t.p$, $t.d \to t_f \to \widetilde{t}.p$ or $c \to t_f \to t.p$. If this probability is greater than threshold (config parameter), we declare the current task t as payable.

1.3 Plan estimation

When task t is currently on auction, before placing the bid, we estimate the plan that the agent would take if it wins task t. If it is possible to carry the task t (agent owns a vehicle v s.t. $v.capacity > t.weight$) we try all possible combinations of placing $t.p$ and $t.d$ task models on each position in task route for each agent vehicles without violating the constraints. From these generated possible plans, the one with the smallest cost is chosen as an estimated solution. The marginal cost of the task t is the difference between the cost of the current plan reduced by the estimated plan's cost that contains t. If marginal cost equals 0, adding task t to the current plan does not change the route of any vehicles.

1.4 Placing the bid

After computing strategies based on past, present and future events, the bid for currently auctioned tasks t is calculated in 3 steps. First, marginal cost is calculated based on the current plan estimation. Next, the approximation of what competitors would bid is calculated. Known approximations are extracted from entries $[t.p, t.d]$ (explained in 1.1) for each competitor. The best competitor has the lowest entry. Since the approximation might be unknown for some competitors, the belief for the approximation is calculated as the number of non-null values divided by the number of competitors. For the final approximation of the best competitor's bid, we linearly approximate the value between our marginal cost and the lowest extracted bid for competitors, based on the belief b: $(1 - b) \cdot marginal\ cost + b \cdot\ extracted\ bid$. For example, if b is 1 (we "know" approximations for all competitors), the final approximation is equal to the competitors' extracted bid. If b is 0 (we "know" nothing about the competitors), the best we can bid is our marginal cost. In this step, our bid is a maximum over our marginal cost and the approximation for the best competitor's bid reduced by 1. The bid might be refined in the final step. Finally, if in the previous step marginal cost is maximum (the best competitor would bit lower) and the offered task is payable (defined in 1.2), we bid below our marginal cost: $bid = (1 - probability\ discount) \cdot marginal\ cost$.

If the bid timeout (read from the configuration file) is not long enough to execute our 3 strategies, we use simpler estimations. In that case, feedbacks from previous auctions and task probability distribution is not used while estimating the plan containing task t is done by placing the t as the last task on the route for a vehicle that can carry it. The marginal cost for t is the estimated traveling distance for picking up and delivering task t from some city multiplied by the approximated cost per kilometer of agent vehicles. Then, the value of the bid is equal to the maximum of the marginal cost for t, and the maximum marginal cost encountered so far.

1.5 Calculating optimal plan for won tasks

Once the auction has finished, the current solution is optimized by running a stochastic local search algorithm implemented in the last exercise, with the following modifications.

Firstly, the initial solution now represents the current estimated solution that contains all the tasks won by the agent. Next, we generate neighbor solutions using the same two transformations as before - exchanging the order of any two task models that belong to the same vehicle or taking one task from a vehicle and giving it as the final task of some other vehicle. From the current solution, one random vehicle is picked ($v1$), on it we apply the first transformation by changing the position of one task to some randomly chosen position. This transformation executes for each of the tasks that belong to $v1$. For applying the second transformation for each vehicle ($v2$, where $v1! = v2$) we randomly choose one of its tasks and assign it to vehicle $v1$. Now the maximum number of generated neighbors is bounded by $2k \cdot n$, with k being the total number of won tasks and n the number of vehicles.

To decide whether to pick the best neighbor solution to be the one to continue with or to stay in the current one, we use the modification of simulated annealing. We accept the transition to the random best neighbor in the walk with probability $min(1, e^{-\beta(bestNeighbor.cost - currentSolution.cost)})$. Initial β is tuned to 0.01 and after every 50000 iterations it is multiplied by 2. Using this method, we avoid local optima.

2 Results

2.1 Experiment 1: Model parameters

2.1.1 Setting

In the experiment, the gained reward is examined comparing different values of model parameters ε, *probability threshold*, and *distribution discount*. Our agent has one competitor dummy agent given in the starting code for this exercise. The topology is England, rngSeed is 123456, the number of tasks is 50, the number of vehicles is 2, the timeouts for setup, bid, and plan are 1, 5, and 30 seconds respectively.

2.1.2 Observations

	ε	*probability threshold*	*distribution discount*	cost	reward
set 1	0.3	0.2	0.25	13 825	73 458
set 2	0	0.2	0.25	17 697	38 751
set 3	0.3	0.2	0.8	18 846	68 437
set 4	0.3	1	0.25	13 061	74 222

Table 1: Comparison of reward obtained for different values of model parameters

Compared to *set* 1, *set* 2 confirms that examining ε close cities significantly increases reward, while from *set* 3 we see that our distribution discount can't be too large. Even though the reward obtained with parameters of *set* 4 is greater than the one when using *set* 1, we chose *set* 1 as the best set of model parameters. This is because in *set* 4 we are not refining the bid at all, but because we are competing with one dummy agent, we can not take this for granted, but try to generalize our approach.

2.2 Experiment 2: Comparisons with dummy agents

2.2.1 Setting

This experiment examines the performance of our agent compared to the competitor dummy agent. From the previous experiment, we use the same timeout values and *set*1 model parameters.

2.2.2 Observations

	topology	tasks number	vehicle number	dummy rewards	agent rewards
set 5	England	50	2	5 024; 5 968	63 452; 70 104
set 6	England	20	2	0 126; 1 163	12 321; 9 081
set 7	Netherlands	100	4	14 973; 16 622	52 074; 55 222
set 8	Netherlands	40	4	1 017; 2 761	20 287; 10 037

Table 2: Performance of our agent when bidding against one dummy competitor agent

In the depicted table, the column vehicle represents how many vehicles each agent owns. The experiment is repeated twice in the tournament setup.

It can be observed that our agent is winning the tournament when competing in different topologies, with different numbers of tasks and vehicles.