Excercise 4
Implementing a centralized agent

Group №10: Alexis Jacq, Louis Faucon

November 7, 2017

1 Solution Representation

1.1 Variables

We define a solution as list of sequences of tasks to pickup/delivery, one sequence per vehicle, with N_T tasks in total. In order to encode the different tasks, we associate the i-th pickup task t_p^i with the integer i (from 0 to $N_T - 1$), and the corresponding delivery task t_d^i with the integer $i + N_T$ (from N_T to $2N_T - 1$).

For example, the sequence $(1;\ 5;\ 1 + N_T;\ 5 + N_T)$ means that the vehicle must pickup task 1, then pickup task 5, then deliver task 1, then deliver task 5.

1.2 Constraints

A solution respects constraints if both of the following conditions are verified:

- all vehicles can realize its sequence of tasks, *i.e.* all biggest sums of simultaneous carried tasks in sequences are smaller than corresponding vehicle's capacity.

- all tasks are picked up at any time i by one single vehicle and delivered at time $j > i$ by the same vehicle.

1.3 Objective function

We want to optimize the total cost $C_{\mathbf{tot}}(S)$ of a solution S:

$$C_{\mathbf{tot}}(S) = \sum_{v \in V} (cost\_per\_km(v) \sum_{t \in tasks(v)} dist\,[prev(v,t), city(t)])$$

where:

- $prev(v,t))$ is the previous city in which the vehicle is before the task t in the sequence s
- $cost\_per\_km$(s) is the cost per km of the vehicle to which is associated the sequence s.

2 Stochastic optimization

2.1 Initial solution

At the beginning, the vehicle with the largest capacity is assigned to pickup and deliver all the tasks one by one:

- sequence = $(0;\ 0 + N_T;\ \ldots\ ;\ N_T - 1;\ 2N_T - 1)$ for the largest vehicle
- sequence = $\emptyset$ for all other vehicles

This solution respects the constraints, otherwise one of the tasks has a weight larger than the largest capacity of available vehicles, and the problem has no solution.

2.2 Generating neighbours

At each step we generate a set of neighboring new solutions. This set originally contains the previous solution S_{old}. We randomly choose a task (t_p, t_d) to pickup and deliver which we remove from its vehicle v_1's sequence, and a second vehicle v_2. We add to the set of neighboring solutions all the possible insertions of t_p and t_d into v_2's sequence (such that v_2 can still realize its sequence given its capacity and such that t_p appears before t_d).

2.3 Stochastic optimization algorithm

With probability $p = 1/\sqrt{\tau}$ – where τ is a temperature parameter linearly increasing with time –, we randomly select a neighbour in the generated set, and with probability $1 - p$, we select the best neighboring solution (with the smaller cost). After 10000 iterations, we stop the algorithm.

3 Results

3.1 Experiment 1: Model parameters

Our only parameters are the initial temperature τ_0 and its increasing rate δ_τ. It controls how fast we trust a locally best solution rather than exploring random modifications. We cross-tested different values for each parameters: $\tau_0 \in \{1, 3, 5\}$, $\delta_\tau \in \{0, 0.1, 0.5, 1, 5\}$.

We set the random seed at 12345, the number of tasks N_T at 30 with constant weight $w = 3$, the number of vehicles N_V at 4 with capacities 3,6,9,12.

We observe that if τ_0 and δ_τ are low, the solution always changes but keep big costs, and if they are too high, the algorithm stops too quickly into local optima, which could have been better with further explorations. We obtained the best results with $\tau_0 = 1$ and $\delta_\tau = 0.5$. The best solution found uses several vehicles

3.2 Experiment 2: Different configurations

Random seed 33333, $N_T = 50$ with constant weight $w = 3$, $N_V = 8$ with capacities 3,3,6,6,9,9,12,12. In this larger setup, we observed the best results with an even smaller increasing rate $\delta_\tau = 0.1$.

3.3 Experiment 3: Edge case

Random seed 12345, $N_T = 30$ with constant weight $w = 3$, $N_V = 4$ with cost per km of 100,100,100,1. We observe that our algorithm finds converges to give all the tasks to the vehicle with lower cost per km, but this convergence takes many iterations.

3.4 Complexity

At each iteration our algorithm creates a number of neighbours that can be in the order of$O(N_T^2)$. The worst case complexity at each iteration does not depend on the number of vehicle or on other parameters of the system, however these parameter will influence the quality of the final solution.