Exercise 4
Implementing a centralized agent

Group №:44 Valentin Kindschi, Cyril van Schreven

December 2, 2019

1 Solution Representation

Our implementation of a centralized agent is inspired by the document "Finding the Optimal Delivery Plan"[1]. To allow vehicles to carry multiple tasks, we split the elementary unit of a task into two actions: the action of picking up and the action of delivering. The algorithms and constraints are updated accordingly.

1.1 Variables

We used four different map objects in our solution: *nextActionA* mapping action to action, *nextActionV* mapping action to vehicle, *time* mapping action to integer, *vehicle* mapping action to vehicle.

1.2 Constraints

The same constraints as in the above mentioned document are used. The constraint regarding capacity has been changed to verify the available capacity after each action. An extra constraint has been added to force a delivery action to be later than a pickup action

1.3 Objective function

The objective function is the sum of the cost of each vehicle. The cost is computed as the $costPerKm * totalDistance$.
We explored a second objective function. The aim is to accomplish all the tasks as quickly as possible. For this the objective function is the cost of the vehicle with the highest cost.

2 Stochastic optimization

2.1 Initial solution

For the initial solution, all the tasks are assigned to the vehicle with the largest carrying capacity. It delivers all the tasks sequentially without carrying multiple tasks. An error is triggered if a task is heavier than its total capacity.

[1] *"Finding the Optimal Delivery Plan:Model as a Constraint Satisfaction Problem"*. Radu Jurca, Nguyen Quang Huy and Michael Schumacher

2.2 Generating neighbours

To generate neighbours, first a random vehicle with a task to perform is chosen. Then, the function $changingVehicle()$ gives the first task of the chosen vehicle to all the other vehicles, which will create a new neighbour plan per vehicle. A second function $changingActionOrder()$ will affect the order of the task handled by the same chosen vehicle. All possible permutations will create a new plan. In the end, at most:

$$(\#Vehicles - 1) + \#Tasks_{RandomVehicle} \cdot (\#Tasks_{RandomVehicle} - 1)/2$$

neighbours solution are computed for each iteration from which all the solution that do not meet the constraints are removed.

2.3 Stochastic optimization algorithm

Our stochastic algorithm will select the solution that minimizes the objective function between all the generated neighbour solution. There is also a probability $probAddCurrent$ that the current best solution, i.e. the previous best plan, is added to the possible solutions. This means that the previous solution is not automatically reelected if its neighbours solution are worst, which increased the exploration and allow to avoid local minimas. We also added a probability *probPickRandom* that a random neighbour solution is chosen instead of the optimal one. This probability has an initial value which will decrease linearly with the iteration number to have more exploration in the beginning of the optimization. Finally, we added a *Dropout* parameter that was ignoring a proportion of the neighborood for the selection.

3 Results

3.1 Experiment 1: Model parameters

In the first experiment, the two parameters were the probability *probAddCurrent* to add the previous best solution to the pool of possible solutions and the objective function. The topology used was England.

3.1.1 Setting

Iterations	5000	**Number of Tasks**	30
$probAddCurrent$	Variable	**Number of Vehicles**	4
Dropout	Variable	**Vehicle's capacities**	9
$probPickRandom_0$	0.5	**Objective function**	Sum / Max

Table 1: Setting experiment 1

3.1.2 Observations

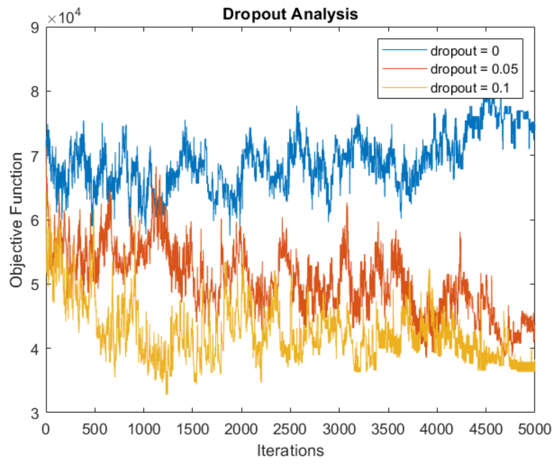

Figure 1: Dropout Analysis

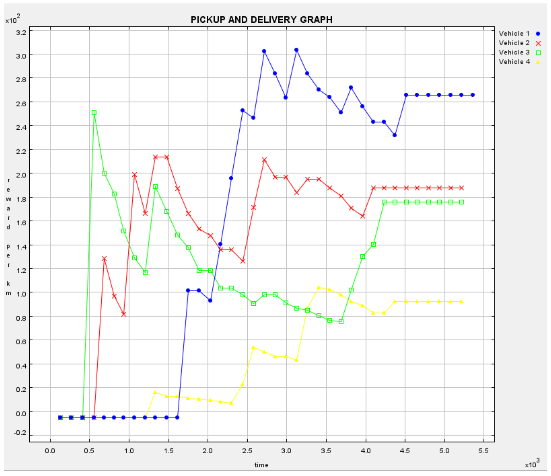

Figure 2: Max Objective Function Results

As we can see on Figure 1, the dropout decreases the objective function values between 0 and 0.1, after which its impact becomes negligible. We also tried to vary *probAddCurrent* between 0 and 1 and we did not notice any change in the objective function results. This is probably due to the fact that we add a lot of stochasticity in our algorithm and including the previous best solution does not really impact the results.

On Figure 2 one can see that using a Max objective function instead of using the sum over all the vehicles, the work is well distributed between and the tasks are all achieved a lot quicker. In fact, with the sum function a single agent is carrying most of the tasks which takes more time, but reduces the total cost between all the vehicles.

3.2 Experiment 2: Different configurations

3.2.1 Setting

In this experiment we try to observe the impact of varying the number of vehicles and the capacity. We kept the parameters as for the previous experiment and used a dropout of 0.1.

3.2.2 Observations

Varying the number of vehicles did not affect the results in a noticeable way. This highlights the fact that one single vehicle performs almost all the tasks.

Capacity	1	2	3	4
Cost	74’661	45’608	38’237	29’133

Table 2: Capacity Analysis

As expected with a higher capacity, the tasks can be delivered for a lesser cost. This shows that the algorithm is capable of properly arranging actions.