Excercise 4
Implementing a centralized agent

Group №17: Gervaise Lara, Khatib Karim

November 5, 2019

1 Solution Representation

1.1 Variables

We assign a list tasks to each vehicle through a hash map where keys are vehicles and values are linked lists of states. In each state, the vehicle either picks up a task or delivers it. This also allows to know the timing of the tasks.

1.2 Constraints

- Delivering a task must be after picking it up.
- The same task cannot be delivered again.
- At some point, all the tasks should be delivered.
- A vehicle can pick up multiple tasks simultaneously, but the total load cannot exceed the capacity of the vehicle.

1.3 Objective function

The objective function is the total cost of all the vehicles that we want to minimise :

$$C = \sum_{i=1}^{N_T} \Big(dist\big(t_i, nextTask(t_i)\big) + length\big(nextTask(t_i)\big) \Big) \cdot cost\big(vehicle(t_i)\big) + \sum_{k=1}^{N_V} \Big(dist\big(v_k, nextTask(v_k)\big) + length\big(nextTask(v_k)\big) \Big) \cdot cost\big(v_k\big);$$

2 Stochastic optimization

2.1 Initial solution

We find the vehicle with the largest capacity and assign it all the tasks. Thus, the vehicle with the largest capacity is ready to deliver all the tasks sequentially.

2.2 Generating neighbours

There are two ways to generate neighbours:
- Taking the first task from the tasks list of one vehicle and giving it to another one.
- Reversing two tasks in a vehicle task list.

Doing so, we generate a new plan set containing many neighbor plans.

2.3 Stochastic optimization algorithm

We do a specified number of iterations of the following algorithm :
First, we compute the neighbouring plans. From these *candidatesPlans* we choose one according to the probability p and to the randomly generated number $num \in [0:100]$.

- If $num <$ p, we choose the plan with the smallest cost from the *candidatesPlans*.
- If $num >$ p, we randomly choose a plan from the *candidatesPlans*. This avoid the algorithm being trapped in local minimum.

3 Results

3.1 Experiment 1: Model parameters

3.1.1 Setting

In our first set of experiments, we compare the performances and results of two different search algorithms - one that applies the *changeTaskOrder* method to all vehicles at every iteration of search for a neighboring plan and one that applies it to only one random vehicle. For the first one, we set the maximum number of iterations of our search algorithm to 150 as this version requires a big computation time. For the second one, we set the maximum number of iterations to 1000 to optimise the results as **increasing the number of iterations allows our algorithm to get a better performances**.

We also want to analyze the effect of the probability p of selecting a newly computed plan A instead of using a random one.

We use 3 vehicles on the England topology with 30 tasks.

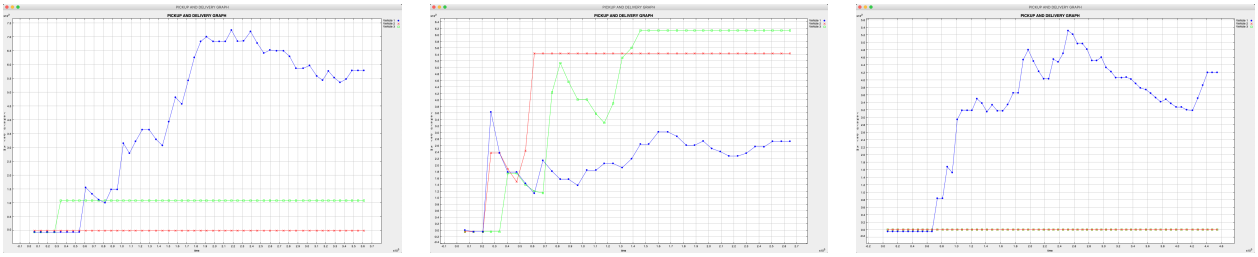
3.1.2 Observations

search algorithm	p	Cost (CHF)	computation time (s)	execution time (s)
Apply changeTaskOrder to one random vehicle **MaxIter = 1000**	0.2	18912	260.151	4.5
	0.3	18945	165.271	3.3
	0.4	18740	114.456	4.3
	0.5	20105	83.412	2.5
	0.8	15553	185.962	3.5
Apply changeTaskOrder to all vehicles **MaxIter = 150**	0.2	23364	260.776	4.7
	0.3	17791	266.707	4.0
	0.4	20858	258.708	4.6
	0.5	19862	251.652	4.4
	0.8	17047	224.911	4.1

Figure 1: Performance = computation time (s) ; execution time (s)

We can see that we obtain the best performance (lower cost) with p = 0.8 and with the algorithm that chooses a random vehicle to apply a change of tasks order.

We also displayed some graphs illustrating the reward per km for every vehicle using the different algoritms. We can observe that in the optimal case that we have found, all three vehicles are used, even



(a) p = 0.8, random vehicle (b) p = 0.5, random vehicle (c) p = 0.5, all vehicles

Figure 2: Results of Experiment 1

though the use of the blue one is predominant (vehicle 1). However, when using the algorithm that generates more different plans (all vehicles), we end up only using one vehicle all of the time.

3.2 Experiment 2: Different configurations

3.2.1 Setting

In our second experiment, we use the optimal parameters found above with the same topology but vary the nymber of tasks and vehicles: we evaluate the performance for 3, 4 and 5 vehicles for 20, 25 and 30 tasks.

3.2.2 Observations

	20 tasks	25 tasks	30 tasks
3 vehicles	13717	14618	15359
4 vehicles	11713	14996	17300
5 vehicles	14118	15208	18524

Figure 3: Performance

One can observe that we get a lower cost when there are less tasks to deliver as there are less kilometers to drive. We can also see that our algorithm performs better when there less agents.

From all our tests we can conclude that the optimal plan isn't fair. This is because more agents allow the company to deliver tasks faster but then each agent has to travel more to go from a delivery city to a pick up city and thus decrease their ratio reward/cost.

The complexity of our algorithm depends on O(maxIter*VehicleNumber^3*taskSetSize^2*planListSize).