Excercise 4
Implementing a centralized agent

Group №2 : Cosmin-Ionut Rusu, Sorin-Sebastian Mircea

November 5, 2019

1 Solution Representation

We represent a solution as a list of actions for each vehicle. Formally:

$$tasks : List < List < Task, Boolean >>$$

such that

$$tasks_i = List < (task : Task, pick\_up : Boolean) >$$

is the set of actions assigned to the vehicle i, ordered in the order they should be executed.

1.1 Variables

We're simply using the *tasks* nested list to represent each solution. We encapsulated this information in our CSP class. For each mutation of a solution, we deep clone the original object. Additionally, we are using a shared seeded *Random* object for reproducibility.

1.2 Constraints

With V as the number of vehicles and T the number of tasks, the following constraints are needed for our implementation:

1. $\sum_{v=0}^{V-1} tasks_i = 2 * T$ since each task has to be picked and dropped exactly once;
2. $\forall v \in [0, .., V)$, $\forall i, j \in [0, len(tasks_{v,i}))$ such that $tasks_{v,i} = (t, true)$ and $tasks_{v,j} = (t, false)$, we have that $i < j$. (i.e we must first pickup a task, and then deliver it at a later point in time);
3. For each vehicle, there is exactly one pickup action and one drop action of a particular task in its list of actions. (i.e there are no duplicates across same vehicle);
4. For each task, there is exactly one vehicle that both picks it up and drops it. (i.e there are no duplicates across the whole vehicles).

1.3 Objective function

Since we know the set of tasks and we need to pick all of them, we must minimize the cost to deliver all of them, which is defined as the sum of all the distance traveled by each vehicle weighted by their corresponding cost per km.

2 Stochastic optimization

2.1 Initial solution

We came up with three ideas to initialize a solution.

In the first idea, we could assign the tasks sequentially (pickup and then deliver immediately): first task to the first vehicle, the second task to the second vehicle and so on, restarting with the first vehicle again if needed, as long as the respective vehicle can handle the workload. This will give a fair initial distribution for each vehicle.

In our second idea, we could bulk load all the tasks to the first vehicle (assuming it can, sequentially, handle the load).

We've empirically chosen the third idea, which simply assign each task to a random vehicle that can handle the weight.

2.2 Generating neighbours

We generate a fixed number (100) of neighbours. With 50% probability we do each of the following mutations:

1. Swap two tasks from a vehicle
2. Move one task from one vehicle to another

We randomly choose all the tasks and the vehicles in each operation.

2.3 Stochastic optimization algorithm

1. In the first step, we generate our initial solution.
2. Then, on each iteration (as long as the time permits) we do the following:
   (a) Generate the neighbours of the current solution
   (b) Get the best neighbour
   (c) With probability p, assign the best neighbour to the current solution.

3 Results

3.1 Experiment 1: Four agents

All our next experiments will be on the *england* map. In the first experiment, we are using 4 agents that have the same characteristics, but are in different home cities.

3.1.1 Setting

We want to analyze how the tasks are being distributed and what is the best solution found.

3.1.2 Observations

The resulted best plan (solution) has a cost of 30355. The plan converged quite fast (after less than half the number of total iterations), suggesting that a local minima has been hit. Empirically, we've also seen that different initialization methods (described in the section 2.2) lead to different local minima.

Clearly, the solution space increases as more agents are added. Since we had a timeout constraint, the algorithm did not had a chance to explore the solution found in the next experiment (which we found to work the best on the same setup).

3.2 Experiment 2: One vehicle

Here, we are using the same map, but with exactly one vehicle. Turns out, this agent works better than the setup with 4 agents from the experiment 1.

3.2.1 Setting

The only difference from the previous experiment (1) is that we are using exactly one agent.

3.2.2 Observations

In this setup, the best plan achieved a cost of 14766. Interestingly, the solution did not converge that fast, suggesting that the algorithm was not stuck in a local minima.

3.3 Experiment 3: Two identical agents

The probability to pick the best neighbour is 0.6. We empirically saw this is the best one for the same setup. The best plan for this configuration was 21382.

3.3.1 Setting

In this experiment, we've just duplicated the agent from the 2nd experiment. We were surprised to find that this still performed worse than the one-agent setup.

3.3.2 Observations

We noticed however, that the algorithm gives more work to one of the agents, suggesting, again, that the algorithm learned that it's better to use only one agent, but was still trapped in a local minima. This demonstrates that the algorithm does not return a global minima, even in simple cases, if the solution space increases. One idea to overcome local minima is to repeat the same algorithm multiple times (including initialization). In this fashion, we might get a better starting point and have more chances to reach a global minima. This idea is used in Neural Networks as well.

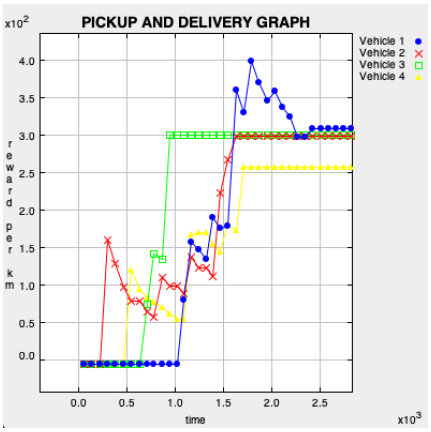

Figure 1: best cost 30355

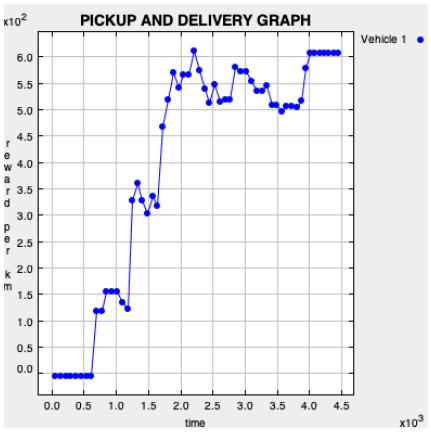

Figure 2: **best cost** 14766

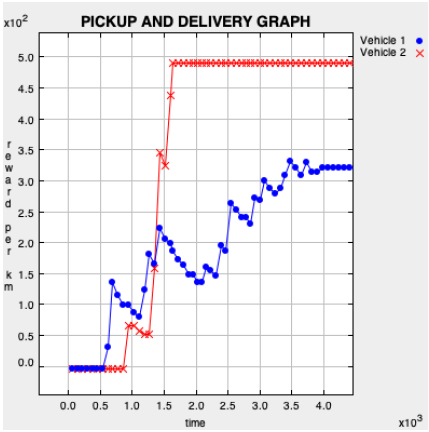

Figure 3: best cost 21382