Excercise 4
Implementing a centralized agent

Group №: 90 Kyle Gerard, Yann Bolliger

November 5, 2018

1 Solution Representation

1.1 Variables

Instead of representing a vehicle's journey as a sequence of tasks, we chose to represent it as a sequence of *pickup* and *delivery* actions. Each task $t \in \mathcal{T}$ has one action of both types.

$$\mathcal{P} = \{pickup(t) : t \in \mathcal{T}\},\ \mathcal{D} = \{delivery(t) : t \in \mathcal{T}\},\ \mathcal{A} = \mathcal{P} \cup \mathcal{D}$$

This accounts for the fact that a vehicle can carry multiple tasks at a time if there are two pickups in a row. The following variables define the first pickup of each vehicle (where $\mathcal{V}$ is the set vehicles).

$$\forall v \in \mathcal{V} :\ firstPickup(v) \in \mathcal{P} \cup \{\texttt{null}\}$$

If the variable is `null` this means the vehicle does not accomplish any actions. In that manner a vehicle goes from its starting point to the location of the first pickup and then it always goes to the associated city of the next task it has to perform. All journeys are made on the shortest possible path. All subsequent actions of a vehicle are defined by the next set of variables:

$$\forall a \in \mathcal{A} :\ nextAction(a) \in \mathcal{A} \cup \{\texttt{null}\}$$

where again the `null` signifies that a vehicle has no further actions to perform. We will define two other sets of variables which will clarify the former:

$$\forall a \in \mathcal{A} : vehicle(a) \in \mathcal{V};\ \ \forall a \in \mathcal{A} : time(a) \in \mathbb{N}$$

The *vehicle* variables define which vehicle carries out a certain action. This can be derived from the $firstPickup$ action at the start of each action chain defined by *nextAction*. (For example if $firstPickup(v) = a, nextAction(a) = b$ then $vehicle(a) = vehicle(b) = v$.)

The second variable can also be derived from the action chains. It simply gives the rank of each action in the chain (for example if $firstPickup(v) = a, nextAction(a) = b$ then $time(a) = 1$, $time(b) = 2$).

1.2 Constraints

As explained before, the action chain for each vehicle define the *time* and the *vehicle* for each action:

$$firstPickup(v) = a \Rightarrow time(a) = 1;\ \ nextAction(b) = c \Rightarrow time(c) = time(b) + 1$$

$$firstPickup(v) = a \Rightarrow vehicle(a) = v;\ \ nextAction(b) = c \Rightarrow vehicle(c) = vehicle(b)$$

Additionally, the same vehicle must *pickup* and *deliver* a task. It has to pickup the task before it delivers it, of course, and **each task must be picked up and delivered**:

$$\forall a \in \mathcal{A} : nextAction(a) \neq a,\ nextAction(a) = \texttt{null} \Rightarrow a \in \mathcal{D}$$

$$\forall t \in \mathcal{T} : vehicle(pickup(t)) = vehicle(delivery(t)) \in \mathcal{V}$$

$$\forall t \in \mathcal{T} : time(pickup(t)) < time(delivery(t))$$

$$\forall t \in \mathcal{T}\ \exists \{pickup(t), delivery(t)\} \subset \mathcal{A} \text{ and } \forall a \in \mathcal{A}\ \exists\, v \in \mathcal{V} : vehicle(a) = v$$

Last but not least, at each time τ a vehicle v can never carry more weight than its capacity.

$$carriedTasks(\tau, v) = \{t \in \mathcal{T} : vehicle(pickup(t)) = v \wedge time(pickup(t)) < \tau \wedge time(delivery(t)) > \tau\};$$

$$\forall \tau \in \mathbb{N} : \sum_{t\, \in\, carriedTasks(\tau,v)} weight(t) \leq capacity(v)$$

1.3 Objective function

The goal of the company is to maximise the reward. Because all tasks have to be delivered, all rewards will be earned and the overall reward is constant. Thus the objective function we want to minimise is the cost of the overall assignment $\mathcal{S}$. We define $dist(a, b)$ to be the shortest distance between the associated cities of actions $a, b \in \mathcal{A}$, $dist(a, \texttt{null}) = 0$, $start(v)$ is the initial postion and $cost(v)$ the cost per kilometre of vehicle v.

$$cost(\mathcal{S}) = \sum_{v\in\mathcal{V}} dist(start(v), firstPickup(v)) \cdot cost(v) + \sum_{a\in\mathcal{A}} dist(a, nextAction(a)) \cdot cost(vehicle(a))$$

2 Stochastic optimization

2.1 Initial solution

Our initial solution is already a valid, greedy solution. Therefore, our program returns a correct assignment at all times, even if there is no computation time at all.

The initial solution is computed by appending the consecutive *pickup* and *delivery* pair to the action chain of the vehicle that is the closest (and has enough capacity). This distance is calculated between the last position in the action chain of each vehicle and the *pickup* location. Therefore in the initial solution the vehicles don't carry multiple tasks at once.

2.2 Generating neighbours

We generate neighbors of a current assignment $\mathcal{S}$ by applying two stochastic operators. For both of them, we randomly choose a vehicle v. Then we remove both actions p, d for a random task t in the action chain of v.

The first set of neighbors is then generated by inserting p, d into the action chains of all other neighbors. This yields a lot of possibilities because the p, d can be inserted in many different ways into the new action chain as long as the capacity and the p before d constraint is respected.

The second set of neighbors is obtained from reordering the actions in v's chain without removing or inserting. Still the reordering has to respect the capacity and the p before d constraint.

2.3 Stochastic optimization algorithm

Local search blabla

3 Results

3.1 Experiment 1: Model parameters

3.1.1 Setting

3.1.2 Observations

3.2 Experiment 2: Different configurations

3.2.1 Setting

3.2.2 Observations