Exercise 4
Implementing a centralized agent

Group № 38: Andrej Janchevski 309143, Orazio Rillo 313423

November 3, 2020

1 Solution Representation

1.1 Variables

Let T denote the set of all tasks that need to be processed and let V be the set of all vehicles owned by the company. We denote the set of all possible actions that can be performed on the tasks by the vehicles as $A = (T \times \{pickup, deliver\}) \cup \{NULL\}$, where $NULL$ denotes the empty action.

In order to represent each of our solutions we use a collection of the following variables:

- $\forall t \in T,\ vehicle(t) \in V$ - the vehicle assigned to the task;
- $\forall t \in T,\ pickupTime(t) \in \{0, 1, \dots\}$ - the time instant at which the vehicle will pick up the task;
- $\forall t \in T,\ deliveryTime(t) \in \{0, 1, \dots\}$ - the time instant at which the vehicle will deliver the task;
- $\forall v \in V,\ nextAction(v) \in A$ - the first action that needs to be performed by the vehicle;
- $\forall t \in T,\ nextActionAfterPickup(t) \in A$ - the next action performed after picking up the task;
- $\forall t \in T,\ nextActionAfterDelivery(t) \in A$ - the next action performed after delivering the task;

1.2 Constraints

In order to accurately model the Pickup and Delivery Problem, our solutions need to satisfy the following constraints:

- **Tasks should be picked up only once**: $\forall t \in T,\ nextActionAfterPickup(t) \neq (t, pickup)$;
- **Tasks should be delivered only once**: $\forall t \in T,\ nextActionAfterDelivery(t) \neq (t, deliver)$;
- **Tasks cannot be picked up after they have been delivered**:
  $\forall t \in T,\ nextActionAfterDelivery(t) \neq (t, pickup)\ \wedge\ pickupTime(t) < deliveryTime(t)$;
- **A task t is the first task to pick up for a vehicle v iff**:
  $nextAction(v) = (t, pickup)\ \wedge\ pickupTime(t) = 0\ \wedge\ vehicle(t) = v$;
- **A task t is the last one to be delivered iff**: $nextActionAfterDelivery(t) = NULL$;
- **The vehicle assigned to the task cannot change during transport**:

$$\forall t_i, t_j \in T, a \in \{pickup, deliver\} \begin{cases} nextActionAfterPickup(t_i) = (t_j, a) \Rightarrow vehicle(t_i) = vehicle(t_j) \\ nextActionAfterDelivery(t_i) = (t_j, a) \Rightarrow vehicle(t_i) = vehicle(t_j); \end{cases}$$

- **Actions should have a correct time ordering**: $\forall t_i, t_j \in T$,
  $nextActionAfterPickup(t_i) = (t_j, pickup) \Rightarrow pickupTime(t_j) = pickupTime(t_i) + 1$
  $nextActionAfterPickup(t_i) = (t_j, deliver) \Rightarrow deliveryTime(t_j) = pickupTime(t_i) + 1$
  $nextActionAfterDelivery(t_i) = (t_j, pickup) \Rightarrow pickupTime(t_j) = deliveryTime(t_i) + 1$
  $nextActionAfterDelivery(t_i) = (t_j, deliver) \Rightarrow deliveryTime(t_j) = deliveryTime(t_i) + 1$

- **At each time step, every vehicle cannot carry more tasks than its capacity**:
  $$\forall t_i, t_j \in T \begin{cases} currentLoad(vehicle(t_i), pickupTime(t_i)) + weight(t_j) > capacity(vehicle(t_i)) \\ \quad \wedge\ nextActionAfterPickup(t_i) = (t_j, pickup) \Rightarrow vehicle(t_j) \neq vehicle(t_i) \\ currentLoad(vehicle(t_i), deliveryTime(t_i)) + weight(t_j) > capacity(vehicle(t_i)) \\ \quad \wedge\ nextActionAfterDelivery(t_i) = (t_j, pickup) \Rightarrow vehicle(t_j) \neq vehicle(t_i) \end{cases}$$

- **Every task should be transported eventually**: Let S be the set of all entries in the three *nextAction*-type vectors. Then S should contain $|V|$ times the element $NULL$ and $\forall t \in T$, once the element $(t, pickup)$ and once the element $(t, delivery)$, implicitly satisfied by the implementation. $S = \bigcup_{v \in V} nextAction(v) \cup \bigcup_{t \in T} nextActionAfterPickup(t) \cup \bigcup_{t \in T} nextActionAfterDelivery(t)$.

1.3 Objective function

As an objective function to minimize we chose to use the total travel cost of all vehicles:
$cost(T, V) = \sum_{v \in V} costPerKm(v) \cdot dist(homeCity(v), pickupCity(nextAction(v)))$
$+ \sum_{t \in T} costPerKm(vehicle(t)) \cdot dist(pickupCity(t), nextActionAfterPickup(t))$
$+ \sum_{t \in T} costPerKm(vehicle(t)) \cdot dist(deliveryCity(t), nextActionAfterDelivery(t))$

2 Stochastic optimization

2.1 Initial solution

Three different methods for generating a starting point for the algorithm were considered. (1) The first is a greedy weight-based assignment, where the heaviest tasks are assigned to the vehicle with greatest capacity etc. (2) The second one is a topology-based search approach: all tasks are assigned to their closest vehicle. Finally, (3) the third one consists in a greedy cost-based assignment, where the lightest tasks are assigned to the vehicle with cheapest cost per km etc.

2.2 Generating neighbours

Sets of neighbors are generated by considering all input argument combinations for each of these three operators:

- $\forall t \in T, v \in V \setminus \{vehicle(t)\}$, $moveTaskToVehicle(t, v)$ transfers the duty of transporting t from $vehicle(t)$ to v, in particular t is added as a new final task to be delivered by v;

- $\forall t \in T, i \in \{0, \ldots, deliveryTime(t) - 1\}$, $changePickupTime(t, i)$ moves the pickup of t to time i;

- $\forall t \in T, i \in \{pickupTime(t) + 1, \ldots, \}$, $changeDeliveryTime(t, i)$ moves the delivery of t to time i;

2.3 Stochastic optimization algorithm

Our implementation of the Stochastic Local Search algorithm respects the general structure proposed in the given document, however we introduce a few modifications.

Iterations employ the ε-greedy approach to select a random neighbor, with $\varepsilon = 1 - p$, where p is a parameter of the algorithm. The best optimal solution discovered during execution is stored and continuously updated. To define the stopping criterion we consider both a maximum number of iterations set to 10000 by default and the given timeout value.

3 Results

3.1 Experiment 1: Model parameters

3.1.1 Setting

For this experiment we wished to analyse the influence of our two model parameters (p and the choice of the initial solution) on some statistics of the algorithm execution.

In the first sub-experiment the settings used are the default ones, the number of tasks is 15 and the weight distribution of the tasks is *Unif([5, 15])*.

To evaluate the influence of the initial solution on the optimal solution research, we computed the optimal cost we got for each of the configurations described in Table 2 and the empirical standard deviation on the number of tasks assigned to each vehicle. This is intended to model the "fairness" of the task assignment across vehicles (the less the more fair). The topology used is England in all cases.

p	$cost(T, V)$
0.30	18286
0.50	17461
0.65	16497
0.80	16088
0.90	15714
0.95	16274
1.00	18989

Table 1: p table

3.1.2 Observations

The results from the experiments on the parameter p are shown in Table 1. We observed that its value must not be too close to the border values 0 and 1. The optimal discovered setting was $p = 0.9$. This is expected, as it represents a good balance between exploration and exploitation.

The results from the second sub-experiment are shown in Table 2. We expected that some initial solutions would have led to better optimal costs in specific configurations. However we observed that the impact of this choice is extremely tiny (yet still present), as in most cases the stochasticity of the algorithm leads to optimal solutions, possibly with a different structure.

$\|T\|$	$\|V\|$	Initial solution	Weights	Capacities	Cost per km	SD	$cost(T, V)$
20	4	1	$Unif([2, 20])$	$[30, 20, 15, 10]$	$[5, 5, 5, 5]$	5.8401	28875.6
20	4	2	$Unif([2, 20])$	$[30, 20, 15, 10]$	$[5, 5, 5, 5]$	6.4703	28055.2
20	4	3	$Unif([2, 20])$	$[30, 20, 15, 10]$	$[5, 5, 5, 5]$	5.8290	28082.8
20	4	1	$Unif([2, 20])$	$[30, 20, 15, 10]$	$[9, 7, 5, 2]$	4.8508	38324.72
20	4	2	$Unif([2, 20])$	$[30, 20, 15, 10]$	$[9, 7, 5, 2]$	5.7344	38212.44
20	4	3	$Unif([2, 20])$	$[30, 20, 15, 10]$	$[9, 7, 5, 2]$	4.6564	38291.42
30	5	1	$Const(3)$	$[20, 25, 30, 35, 40]$	$[4, 5, 6, 7, 8]$	9.4	17262.2
30	5	2	$Const(3)$	$[20, 25, 30, 35, 40]$	$[4, 5, 6, 7, 8]$	7.234	17.456.2
30	5	3	$Const(3)$	$[20, 25, 30, 35, 40]$	$[4, 5, 6, 7, 8]$	9.86006	17262.76

Table 2: Initial solution experiments

3.2 Experiment 2: Different configurations

3.2.1 Setting

For this experiment we wished to analyse the scaling effect of the given number of tasks and vehicles on the execution time and on the standard deviation of the task assignment.

In order to facilitate a relevant comparison we used the same configuration for all runs: $Unif([5, 15])$ task weight distribution, all other settings are kept as the default.

$\|T\|$	$\|V\|$	SD	Execution time (sec)
25	3	2.73	13.339
50	3	5.22	57.002
100	3	6.66	299.54
25	4	2.52	9.813
25	5	3.61	6.328
25	10	2.51	5.214

Table 3: Scaling effects of the number of tasks and vehicles

3.2.2 Observations

The results from the second experiment are presented in Table 3. By increasing the number of tasks by a factor of 2, we observed an increase by a factor of 5 in the execution time of the algorithm. As such, we predict that in general the time complexity of the procedure is directly proportional to the number of tasks. In addition, a greater number of tasks seems to negatively affect the fairness of task assignment, as we observed its standard deviation also to increase.

Interestingly, we observed that the number of vehicles seemed to influence none of the two analyzed quantities.