Excercise 4
Implementing a centralized agent

Group №: Liangwei CHEN, Siyuan LI

November 5, 2019

1 Solution Representation

1.1 Variables

In total we have three types of variables, namely Next, Time and Vehicles.

- $Next : V \cup Pickup \cup Delivery \mapsto Pickup \cup Delivery \cup NULL$
  These variables capture the immediate next action of current state. For instance, if $Next(v) = Pickup(t1)$, the immediate action when v begins its job is to pick up t1. Similarly if $Next(x) = NULL$, then no action happens after x. From the definition one may observe that we have $|V|+2|T|$ $Next$ variables in total.
- $Time : V \cup Pickup \cup Delivery \mapsto \mathbb{Z}$
  These variables capture the relative order of actions. $Time(x) = t$ means that x is the t^{th} action executed by its carrying vehicle. For convenience we set $Time(v) = 0$ for all $v \in V$.
- $Vehicle : Pickup \cup Delivery \mapsto V$
  These variables represent the carrying vehicles of actions. $Vehicle(x) = v$ indicates that x is carried out by v.

1.2 Constraints

1. $Next(x) \neq x$
2. $Next(x) = y$ indicates $Time(x) + 1 = Time(y)$ if $y \neq NULL$
3. $\forall x, y \in Pickup \cup Delivery, \exists n > 0\ s.t.\ Next^n(x) = y \iff Vehicle(x) = Vehicle(y)$ and $Time(x) < Time(y)$
4. $\forall t \in TaskSet, Time(Pickup(t)) < Time((Delivery(t))$ and $Vehicle(Pickup(t)) = Vehicle(Delivery(t))$
5. All tasks must be delivered: $\sum_x Next(x) = \{NULL\} * |V| \cup Pickup \cup Delivery$
6. No vehicle is overloaded: $\forall v \in V$, $\forall n\ s.t.\ \exists t \in TaskSet$ with $Time(Pickup(t)) = n$, we have $\sum_{x \in TaskSet\ s.t.\ Time(Pickup(s)) \leq n \leq Time(Delivery(s))} Weight(s) \leq Capacity(v)$

The forth constraint ensures that the pickup and delivery actions of certain task must occur at the same vehicle and the former must take place before the latter. The third constraint ensure that there must exist a Next sequence from the pickup action of a certain task to its delivery action. The last constraint is sufficient and necessary since overloading may happen if and only if it happens at some pickup action. Thus by checking the validity at each pickup time, the validity of the whole plan is revealed.

1.3 Objective function

The cost function can be described as the following algorithm.

Algorithm 1 getCost ($Next$, V)

```
1: $cost \leftarrow 0$
2: for $v \in V$ do
3:     $current \leftarrow v$
4:     while $Next(current) \neq NULL$ do
5:         $cost \leftarrow cost + dist(current.city,\ Next(current).city) * v.costPerKm$
6:         $current \leftarrow Next(current)$
7: return $cost$
```

$$\text{where } x.city = \begin{cases} x.baseCity(), & \text{if } x \in V \\ x.pickUpCity(), & \text{if } x \in Pickup \\ x.deliveryCity(), & \text{if } x \in Delivery \end{cases}$$

2 Stochastic optimization

2.1 Initial solution

The initial solution is generated by assigning all tasks to the vehicle with greatest capacity in the way that the vehicle does not pick up a new task until it finishes deliver current task. If there exists a task not fitting in to any vehicle, then the problem is unsolvable. If there are multiple candidates with greatest capacity, randomly select one as the initial holder.

2.2 Generating neighbours

We have proposed and implemented three ways to generate neighbours.

- Change Vehicle: Randomly pick a vehicle with non-empty load. Let t represented the first task picked up by it. Select another vehicle v s.t. $v.capacity() >= t.weight()$. Then instruct v to pickup and deliver t at the beginning. Repeat this procedure for all feasible v.

- Postpone Pickup: Postpone the pickup action of a task to any time before its delivery. Specifically, let $< x_1, ..., x_n, p(t), y_1, ..., y_m, d(t), z_1, ..., z_l >$ represents a $Next$ sequence of a vehicle. Then any $< x_1, ..., x_n, y_1, ..., y_i, p(t), y_{i+1}, ..., y_m, d(t), ... >$ is a feasible execution sequence and thus leads to a neighbour of current plan to be considered in local search. In terms of feasibility, it is obvious that constraints 1-5 hold after postponing a pickup action. Constraint 6 also holds since postponing a pickup action does not increase any total weight at any pickup time.

- Postpone Delivery: Postpone the delivery action of a task till reaching a pickup action such that postponing the delivery after it will lead to overloading. Let $< x_1, ..., x_n, d(t), y_1, ..., y_m >$ be the $Next$ sequence of a vehicle, then $< x_1, ..., x_n, y_1, ...y_i, d(t), y_{i+1}, ..., y_m >$ is a feasible sequence iff $\forall 1 \leq j \leq i$, total weight at $Time(y_j)$ does not overload the vehicle. From the argument of constraint 6, one may observe that this condition holds iff it holds at every pickup time from $Time(y_1$ to $Time(y_i)$. Thus by checking feasibility at every pickup location after $d(t)$ is sufficient to generate a feasible postpone delivery sequence and eventually a neighbour variable assignment.

2.3 Stochastic optimization algorithm

The algorithms simply decide the next variable assignment by flipping a biased coin. With probability $1 - p$, it sticks with current assignment. Meanwhile with probability p, it chooses the best among neighbours as the next assignment.

3 Results

3.1 Experiment 1: Model parameters

3.1.1 Setting

In the following experiment we demonstrate the results influence of decision threshold p. The lower p becomes, the greater probability current plan will be insisted in the next round. The experiment has been conducted with England topology, 30 tasks with default distribution, 3 vehicles with 30 units of capacity each and runs for 5000 rounds.

3.1.2 Observations

p	0	0.35	0.5	0.7	1.0
cost	62147	18029	19514	21423	22417

From the experiments one may see that, in general it is not optimal to always move the best neighbour. Above all the best neighbour even may not be better than current one. Besides, due to the randomness in selecting neighbour, it is highly possible that the optimal neighbour selected in current round may not outperform the neighbour selected in the next round if one sticks with current plan in the next round. The results indicate that $p = 0.35$ is a good decision boundary for stepping forward.

3.2 Experiment 2: Different configurations

3.2.1 Setting

For simplicity we have insisted on the England topology in the following experiments. The number of tasks and vehicles have been set as vehicle to illustrate the insight of centralized planning algorithm. The task distribution and capacity of each vehicle have been set to default. Furthermore, p has been set to 0.35 according to the result retrieved above.

3.2.2 Observations

#Vehicles	1	2	3	1	2	3
tasks	60	60	60	30	30	30
cost	37090	38415	36120	21012	19407	17512

From the results one may observe that in general the larger number of vehicles, the lower the cost. This phenomenon appears in not only the case with few tasks but also the case with more tasks. By digging into the the insight of the plan, we found that in order to achieve better plan, some vehicles are just set as spared while some are loaded with the majority of tasks. This may result from the biased topology in the sense that some cities are close to the majority while some are far from majority. Lastly, our algorithm is exponential costly with respect to the number of vehicles and size of task set due to its neighbour generation scheme.