Excercise 4
Implementing a centralized agent

Group 48 : Alexandru Mocanu, Ivan Yurov

November 5, 2019

1 Solution Representation

1.1 Variables

Let there be T tasks and V vehicles. The actions that can be performed with a task are PICKUP and DELIVER. Based on the single task in the vehicle scenario, we create our new algorithm which uses the following variables:

- nextTask(v) - associates the first task to be picked-up by vehicle v, or NULL, if the vehicle has no task to take care of
- nextTask(t,act) - points to the next task to be performed by the vehicle which performs action act with task t
- space(v) - specifies the capacity of vehicle v
- space(t,act) - specifies the space in the vehicle after carrying out action act with task t
- time(t,act) - the order index of the (task, action) pair in the list of tasks of a vehicle
- vehicle(t) - the id of the vehicle that handles task t

Therefore, the total number of variables that we have is $V + 2T + V + 2T + 2T + T = 2V + 7T$.

1.2 Constraints

The constraints for this scenario are:

- nextTask(t,DELIVER) != (t,act)
- nextTask(t,act) != (t',DELIVER), if vehicle(t') != vehicle(t)
- nextTask(v) != (t,DELIVER)
- nextTask(v) != (t,PICKUP) if load(t) ¿ space(v)
- nextTask(t,act) != (t',PICKUP) if load(t') ¿ space(t,act)
- nextTask(v) = (t,PICKUP) implies time(t,PICKUP) = 1
- nextTask(t,act) = (t',act') implies time(t',act') = time(t,act) + 1
- nextTask(v) = (t,PICKUP) implies vehicle(t) = v
- nextTask(t,act) = (t',act') implies vehicle(t') = vehicle(t)
- nextTask(t,act) = NULL implies space(t,act) = space(vehicle(t))
- space(t,act) ¿= 0 for all task-action pairs
- all tasks must be delivered, so the set of values of the variables in the nextTask arrays must be equal to the set of T tasks plus V null values

1.3 Objective function

The function that we optimize is the total distance travelled by all the vehicles controlled by the centralized agent.

2 Stochastic optimization

2.1 Initial solution

To generate the initial solution, we first pick the vehicle that has the largest capacity. Then, we take all the tasks t and add (t,PICKUP), (t,DELIVER) task-action pairs one after the other in this vehicle. If one of the tasks is to heavy to carry, then no other vehicle can carry it and there is no solution for the problem.

2.2 Generating neighbours

To generate the neighbours of a state (i.e. variable value-mapping), we first pick uniformly at random one vehicle that carries tasks and we perform one of the following two moves:

- Moving one of the tasks in another vehicle at random. We select the PICKUP and DELIVER entries of a task and move them together to the end of the task list of another vehicle. The task can be moved to the other vehicle if that vehicle has enough capacity to carry it.
- Exchange two task-action pairs, (t,act) and (t',act') with $time(t, act) < time(t', act')$, in the vehicle. We can perform such an exchange if the following conditions are met: if act' = DELIVER, then $time(t, act) > time(t', PICKUP)$; if act = PICKUP, then $time(t', act') < time(t, DELIVER)$; after the exchange, when we update the space variables, they remain positive.

We include all such possible moves in the list of neighbours and then choose the move that brings the best cost improvement. If there are multiple moves that provide the same improvement, we choose one of them at random.

2.3 Stochastic optimization algorithm

The stochastic optimization algorithm receives the list of vehicles and the list of tasks as inputs and performs the following steps:

- We initialize the plan with the initial solution described above.
- We iterate for a fixed number of steps and do the following
  - We generate a list of possible moves that bring us to neighbours of the current state.
  - We choose one of the best moves.
  - We perform the move chosen with some probability p.
  - If the plan that we get provides the best cost, we store it as the best plan.
- We return the best plan found.

3 Results

3.1 Experiment 1: Model parameters

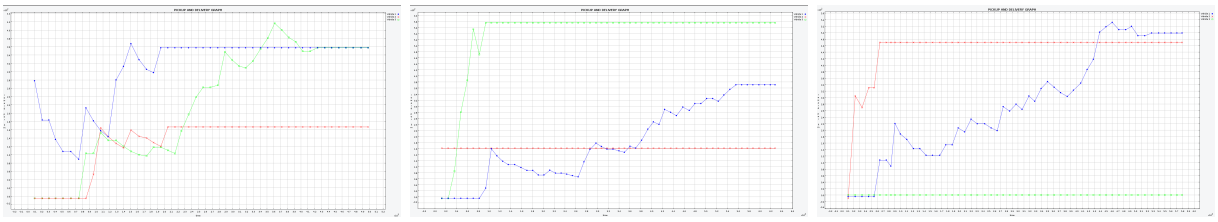
We run the model for 3 agents with different vehicle capacities and cost-per-km settings.

3.1.1 Setting

We use the England topology. We have 30 tasks to be delivered. We use 3 agents positioned at Newcastle, Cardiff and Plymouth with three different setups: all agents have capacity 30 and cost-per-km 5; all agents have capacity 20 and cost-per-km 5, except for one that has capacity 30; all agents have capacity 30 and cost-per-km 10, except for one that has cost-per-km 5.

3.1.2 Observations

Figure 1 shows the performance of the agents in the 3 different setups. We see that in all cases, one of the agents performs more tasks than the others. However, this effect is accentuated when one of the agents has more capacity and even more so when one of the agents has a smaller cost-per-km.



(a) Same settings (b) More capacity (c) Smaller cost

Figure 1: Evolution of 3 agents in 3 different setups

3.2 Experiment 2: Different configurations

We first analyze how changing the number of vehicles affects the cost of the best plan that we find.

3.2.1 Setting

We use the England topology with 1-4 agents. All agents have capacity 30 and cost-per-km 5 and they are positioned in Newcastle, Cardiff, Plymouth and Norwich. There are 30 tasks that need to be delivered.

3.2.2 Observations

We perform 10 runs with 1-4 agents. Table 1 presents the mean and standard deviation of the cost, as well as the mean and standard deviation of the maximum number of tasks carried by an agent for each of the 4 settings.

1 agent	2 agents	max tasks	3 agents	max tasks	4 agents	max tasks
16.3k ± 1.8k	20.4k ± 3.4k	22.2 ± 3.5	26.9k ± 4.5k	18.1 ± 3.9	28.7k ± 3.7k	17.4 ± 3.2

Table 1: Cost and maximum number of tasks carried by an agent in terms of number of agents

We can see that the more vehicles we have, the worse the performance is. That is because for more agents that need to be coordinated, we need more time to find a good plan for them. For a fixed number of steps for all settings, it is easier to perform the optimization for a smaller number of agents. We see that the tasks are always concentrated on one or two agents every time, so some agents will have to do more work than others. In some cases, when the tasks were more uniformly distributed, the agents as a whole were actually performing worse.