Excercise 4
Implementing a centralized agent

Group №45:ERBACHER, PIQUET

November 6, 2018

1 Solution Representation

1.1 Variables

We have 3 variables represented in one Hashtable in our problem:

-Action:Is the action performed by the vehicle , it has 2 parameters, the task and a if the vehicle has to pick it up or deliver it.(class: Task and a boolean) for N task on the topology , it makes $2N$ actions. These actions are executed sequentially by each vehicle.

-Time to specify the order of actions.

-Vehicle that do the sequence of actions.

1.2 Constraints

Constraints are what make sure that the generated Array is still a solution.Some of the constraints are implicit because of implementation, other are explicit. These are the list of explicit constraints:

-The 2 actions associated to the same task are in the same vehicle

-For the same task, the action 'pick-up' is always before the action 'deliver'.

-The maximum capacity of the vehicle stay superior or equal to the sum of the weights of the tasks carried at the same time.

Vehicle 0	Vehicle 1	Vehicle 2	Vehicle 3
Task 1(P)	Task 0(P)	Task 4(P)	Task 7(P)
Task 5(P)	Task 3(P)	Task 4(D)	Task 7(D)
Task 1(D)	Task 3(D)		Task 6(D)
Task 2(P)	Task 0(D)		Task 6(D)
Task 2(D)			
Task 5(D)			

Figure 1: HashTable 'nextTask'

The implicit constraints:

-For an Action in the array at index i, the next action's index is i+1.

-Indexes of actions in the array are their time parameter.(actions are done in sequentially from index 0 to 2*m (with m number of tasks attributed to the vehicle)

-First action's index of vehicle is always 0

1.3 Objective function

In our case all the tasks have to be delivered and all the vehicles have the same speed and cost/km. We just need to minimize the sum of the total distance of all vehicles.

2 Stochastic optimization

2.1 Initial solution

Our initial solution is always the same, each vehicle picks the two corresponding actions for one task from the actions' list. Example: 1st task is given to the 1st vehicle, 2nd task to the 2nd vehicle, until the last vehicle, then, in the same order, all the vehicles picks a 2nd task, and so on, until all tasks are attributed. With this initial solution all the vehicle start with almost the same number of tasks. This is a solution because, if $time$(pickup task i) $= t$ then $time$(deliver task i) $= t + 1$, vehicles never exceed their capacities.

2.2 Generating neighbours

To generate a neighbour solution,we make elementary transformation on the array. There are 2 possible transformations: The transformation 'swap' that switch 2 actions in the same vehicle. (the constraints make sure that action with the same task are not switching order.) The transformation 'Give' that takes the 2 actions associated to one task and move them at the end of an other vehicle(in the right order). Each time we want to move to a neighbor in the solution space, we randomly choose one of these two elementary transformation.

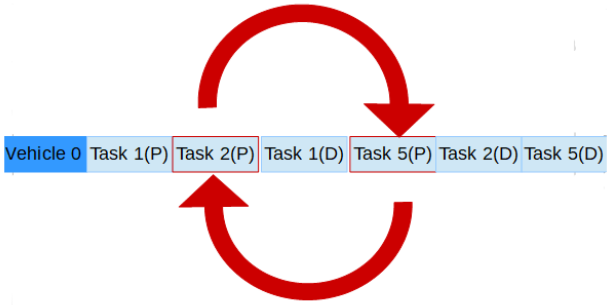


Figure 2: Swap transformation

2.3 Stochastic optimization algorithm

We are working on the solution's space, from an initial solution we choose a random neighbour solution with a random transformation (that respect the constraints), if the cost is lower, this is the new starting point, and then we repeat until the best solution doesn't evolve anymore. Because transformations are chosen randomly, we miss other better possible path. When the solution does not evolve after many iteration(can't find a better neighbour), we restart to the initial solution and look if there are better path that could converge to a better solution. We return the best overall solution. Instead of restarting from the same initial solution we could also pick a random initial solution every time we converged to a minimum. This is an efficient algorithm to find better solutions than the initial one, but it does not lead the optimal solution. The algorithm is described bellow:

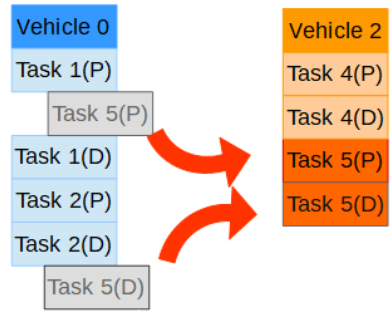


Figure 3: give transformation

```
Bestplan = initial solution
Loop for dedicated time :
    Plan = Initial solution
        Loop until plan don't change after many iteration :
            Choose random transformation to do on plan:
                If swap : 1 random vehicle, 2 random tasks
                If give :  2 random vehicles, 1 random task
            Compute new plan = transformation(plan)
            If cost(new plan)< cost(plan) :
                plan = new plan
        If cost( plan)< cost(Bestplan) :
            Bestplan = plan
Return Bestplan
```

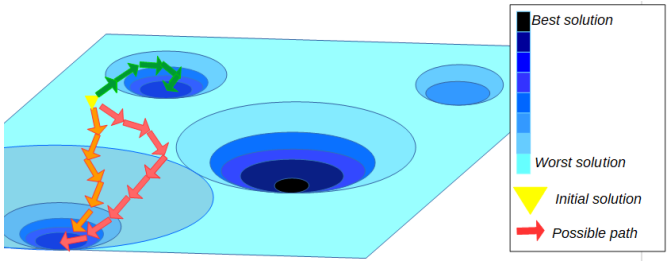


Figure 4: Stochastic optimization

3 Results

3.1 Experiment 1: Model parameters

3.1.1 Setting

In our solution we run with 4 vehicles. We search a total of 500 000 plans. we restart to the initial solution if the 10 000 first computed neighbors are not better. We compare the solution with the naive agent. Naive agent is only one vehicle with all the tasks done one after the other.

3.1.2 Observations

When the number of task is large, the improvement seems more constant. When the number of task is low, it seems to really depend of the topology. The number of possible solution increase exponentially with the number of tasks. For a small number of tasks we can find the optimal solution. We ran 10 times for 5 tasks we always have the same results. But for 30 tasks we never had the same results because there are too many solution to explore.

For 30 tasks:

Topology/Agent	Naive	Centralized
England	13746.2	9020.0
France	41312.0	28366.0
Netherlands	6026.9	4151.8

For 5 tasks:

Topology/Agent	Naive	Centralized
England	2477.4	1697.1
France	5387.0	4590.0
Netherlands	974.6	566.0

Improvement cost between centralized and naive

Topology/Number task	5 tasks	30 tasks
England	31%	34%
France	15%	31%
Netherlands	41%	31%

3.2 Experiment 2: Number of vehicles

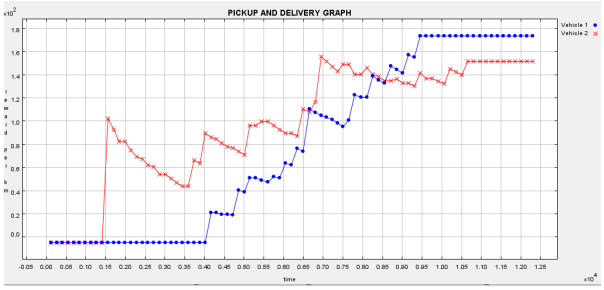
3.2.1 Setting

We ran the simulation for England topology with 30 tasks to deliver. The first time with 2 vehicles, then with 8 vehicles.

3.2.2 Observations

For 8 vehicles: cost: 9780.2 We can see that some vehicles don't have many task assigned while other have many. They finish to deliver all task in $time = 7.5 * 10^3$

For 2 vehicles: cost=9749.4 The two vehicles have almost the same number of task assigned.But they deliver all task in $time = 10.5 * 10^3$. Because the number of vehicles increase, the number of possibilities also increase. Because the search is stochastic, adding more and more vehicles doesn't mean better result, because the complexity is increasing exponentially with the number of vehicles.The Graph on the left shows for 2 vehicles, on the right for 8 vehicles:



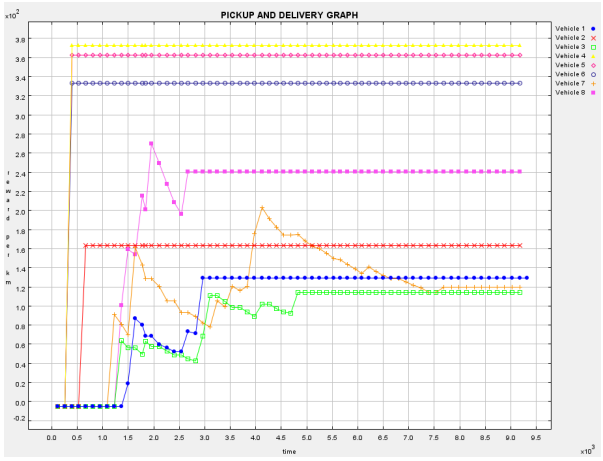


Figure 5: 30 tasks, 8 vehicles