Implementing a Centralized Agent

Smail Ait Bouhsain, Maxime Gautier

November 6th, 2018

1 Solution Representation

1.1 Variables

Our solution representation is called a Strategy and it contains several variables that allow us to create a list of plans once we find the best strategy. It is created based on the principle that the vehicles and tasks are given in an ordered list and so an integer serving as id can be attributed to each. We also use the fact that our actions can only be to pick up a task or deliver one so we have twice as many actions as tasks. A Strategy contains :

nextAction This is an array of Actions which is ordered as such : First all actions of pick up, then all actions of delivery then all the vehicles. Its size is therefore twice the number of tasks plus the number of vehicles. Each action is the next action of the one where we are in the array, so for example the action in nextAction[0] is the action executed after picking up the first task.

vehicle This array of integers contains the number of the vehicles which carry a certain task, so for example if the first task is carried by the second vehicle then vehicle[0]=1. Its size is the number of tasks since the same vehicle will pick up and deliver the same task.

time This array of integers represents the time of each action, its size is the number of actions. This first task of each vehicle is picked up at time 0.

load This is a 2D array of integers that represent the carried weight of each vehicle at any given time.

We convert our strategy into a list of plans for each vehicle at the end of our stochastic optimization algorithm.

1.2 Constraints

A task must always be picked up before it is delivered. All tasks must be delivered and a task cannot be delivered twice. The total weight of tasks picked up cannot exceed the vehicle's capacity. The first action of a vehicle is always to pick up a task, the same vehicle must pick up and deliver the same task. Time must be linear : the time of the next action must be the time of the previous action + 1.

1.3 Objective function

The function we optimize to find the best strategy is the total cost of all vehicles and it is calculated as follows :

$$C = \sum_{i=1}^{2\times N_t} distance(cityOf(a_i), cityOf(nextAction(a_i))) \times cost(vehicle(a_i)) + \sum_{k=1}^{N_v} distance(homeCity(v_k), cityOf(nextAction(v_k))) \times cost(v_k)$$

With N_t as the number of tasks, N_v as the number of vehicles, a_i as the action i and v_k as the vehicle k.

Of course the city in the first sum depends on whether the action and next action are to pick up or deliver a task so we made several cases with the right equations. The second sum represent the first action of each vehicle and it will always be the distance to a pickup city since a vehicle's first action is always to pick up a task.

2 Stochastic optimization

2.1 Initial solution

Our initial solution consists in putting the first task in the first vehicle's plan, the second task in the second etc until all vehicles have a task then add the next task to the first vehicle's plan and the next one to the next vehicle's plan until all tasks are distributed.

2.2 Generating neighbours

We generate neighbors in two different manners. First we change the vehicle carrying tasks by taking the first task of a random vehicle and putting it as the first task of another vehicle. We do this for all the tasks of a vehicle and for all vehicles which generates a lot of potential neighbors, some of which are excluded as they do not respect our constraints.

Secondly we change the order of the actions of one random vehicle to generate neighbors with all the possible permutations, many of which are once again excluded if they do not fit in our constraints.

2.3 Stochastic optimization algorithm

Our local choice algorithm finds the strategy with the lowest total cost out of all the neighbors and returns it with a probability of p. It returns the current strategy with a probability of 1-p. This probability's purpose is to avoid being stuck in a local minima.

Our stochastic optimization algorithm generates neighbors and finds a best local strategy for a chosen number of iterations. This does not give us the optimal strategy but it finds an efficient one in a reasonable amount of time which is what we are looking for. Of course, a higher number of iterations usually takes more time to compute without always finding a more optimal solution.

3 Results

3.1 Experiment 1: Model parameters

3.1.1 Setting

The only changeable parameters we have implemented is the number of iterations and the probability p in the stochastic optimization algorithm. We varied p from 0.1 to 0.99 and the number of iterations from 1000 to 100 000. The configuration we used is as follows : England topology, 30 tasks with a seed of 12345 and 4 vehicles.

3.1.2 Observations

Since this is a scholastic algorithm, running the exact same experiment 10 times gives 10 different results so we had to run a lot of simulations to draw real conclusions. What we found follows what we expected : extreme values of p (0.1 or 0.9) usually give less good results than values between 0.3 and 0.5. Our total cost varies from 25 000 to 40 000. We also found that increasing the number of iterations rarely gave better results and that we could even find plan costs around 25 000 with only 1000 iterations.

3.2 Experiment 2: Different configurations

3.2.1 Setting

The topology is England with a seed of 12345, p = 0,4 and we run with 15000 iterations.

3.2.2 Observations

	20 tasks		30 tasks		40 tasks	
	cost	time [s]	cost	time [s]	cost	time [s]
2 vehicles	38 691	3.409	55 970	10.689	81 209	21.039
3 vehicles	37 022	1.577	48 757	4.934	65 193	16.634
4 vehicles	26 648	2.153	27 300	14.930	50 591	19.299
5 vehicles	26 829	2.805	33 515	4.623	45 524	9.095

Figure 1: Results of Experiment 2

We ran several experiments with different parameters to see how the number of tasks and vehicles influenced the results. AS the results are partly random, some might be completely off but we ran each experiment several times to be sure. We can see that when there are 40 tasks, a higher number of vehicles is required to lower the cost and computation time. Basically, for each number of tasks, there is an ideal number of vehicles to minimize the cost. This also explains why some vehicles often do more work than others as we can see in the following experiment :

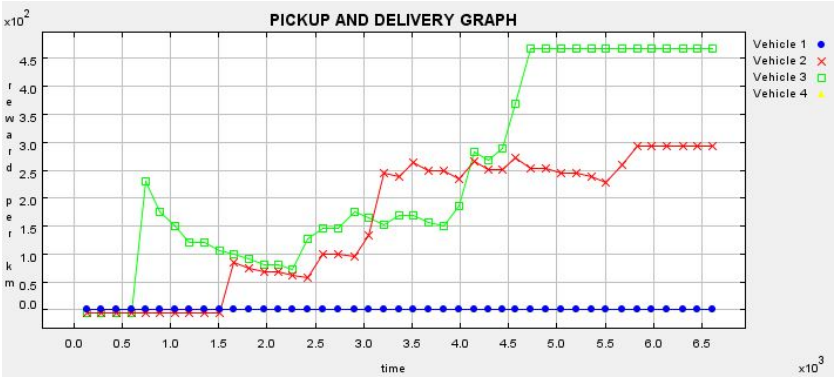

Figure 2: Simulation with 30 tasks and 4 vehicles

One would think the complexity of the algorithm would grow with the number of tasks and vehicles but because of the way our algorithm is written, some combinations compute much faster and give better results than expected.