Excercise 4
Implementing a centralized agent

Group №87 : El Abridi Ali, Kalim Tariq

November 5, 2019

1 Solution Representation

1.1 Variables

Following up on our pickup and delivery problem, we have implemented a centralized agent that focuses on coordinating the actions between the different vehicles of a company. In order to achieve that, we focused on designing the adequate representation of the solution space for the Stochastic Local Search algorithm using the following variables:

- ***TaskTypeTuple***: a tuple containing the Logist.Task and the type of the action being performed on that Task, i.e., *PickUp* or *Delivery.*
- ***Vehicle Plan***: The representation of a plan for a single vehicle which includes a list of nextTask of type TaskTypeTuple which represent the different task performed by the vehicle sequentially in time, which can be later on transformed into a Logist.Plan type. This representation with TaskTypeTuple allows a vehicle to carry more than a single task at a time.
- ***Solution***: The solution variable represents a potential candidate solution to the pickup and delivery problem, which includes a list of VehiclePlan for vehicle; hence, a list of Logist.Plan

1.2 Constraints

- ***No two vehicle can have the same task***: The following constraint is maintain implicitly in the implementation of the solution, by making sure that for each unique solution no unique task is present in two different vehicles.
- ***No Delivery before PickUp***: The following is implemented in the form of a CheckTimelineContrainst which ensures that for every VehiclePlan of a vehicle, the later cannot deliver a task before picking it up.
- ***Vehicle Capacity***: In every solution being generated, the vehicle cannot exceed the maximum weight capacity of the vehicle, otherwise, the potential solution is discarded and not added to the potential neighbours.

1.3 Objective function

The objective function of the SLS algorithm is to minimize the cost of all the vehicles for a certain candidate solution, which is the the sum of travelled distance * costPerKm for that specific vehicle.

2 Stochastic optimization

We implemented the stochastic local algorithm following the guideline of the course's paper.

2.1 Initial solution

We have experimented with different potential initial solutions in order to help the stochastic local search algorithm find the best solution rapidly.

- ***Largest Vehicle***: The selectInitialSolutionLargestVehicle assigns all the tasks available to the vehicle that has the highest capacity without overlapping the pickup and delivery of the different tasks.
- ***Random Vehicle***: The selectInitialSolutionRandomVehicle assigns every tasks randomly to a vehicle that has the capacity to carry it.
- ***Closest Vehicle***: the selectInitialSolutionClosestVehicle assigns every tasks to the vehicle that has the shortest distance between its homeCity and PickUp location of the task.

2.2 Generating neighbours

In order to generate potential other solutions (neighbours) we have applied two different methods as described in the paper.

- ***Changing Vehicle***: This method chooses a random vehicle carries at least one task, and assign its first task to another vehicle, generating a set of potential solutions.
- ***Changing Task Order***: This method chooses a random vehicle that carries at least two tasks, and generate a set of solution in which all combination of sets are swapped to produce a potential solution. In addition to that, since we assume that a vehiclePlan is described by a list of nextTask contains TaskTypeTuple, the method enables the generation of potential solutions in which a vehicle can carry more than a single task at a time.

2.3 Stochastic optimization algorithm

The stochastic optimization algorithm for constraint satisfaction problem is implemented as described in the course's paper by choosing an initial solution (closest vehicle), and repeating the process until the algorithm has reached a certain number of max iterations or has timed out by performing the following two methods:

- ***ChooseNeighbours***: This method uses the two methods described earlier for generating all potential neighbours from a specific solution, and select the on that has the minimum cost according to our objective function (minimize the total cost for all the vehicle), in case of a tie in the objective function, it chooses a solution at random.
- ***Local Choice***: This methods selection with probability 1 - p the old solution or with probability p the the new solution candidate that gives the best improvement of the objective function. p has been set to 0.35, it helps the algorithm converges faster while avoiding being trapped in a local minima.

3 Results

3.1 Experiment 1: Model parameters

In this experiment, we experiment with different values of the probability of choosing the new solution, and the different initialization methods.

	p = 0.3	p = 0.4	p = 0.5
selectInitialSolutionLargestVehicle	31250	32000	33440
selectInitialSolutionRandomVehicle	31850	31940	33840
selectInitialSolutionClosestVehicle	17900	19080	19730

Table 1: Total cost depending on the initlization and probability.

3.1.1 Observations

We can clearly see that selectInitialSolutionClosestVehicle methods and the probability 0.35 minimizes the most the total cost of all the vehicles. This might be due to the fact of using a smarter allocation of task to vehicle depending on the distance which lower the total cost, and p = 0.3 seems to produce the best results with it as seen in table 1.

3.2 Experiment 2: Different configurations

In this experiment, we are demonstrating the use of centralized agents for multiple numbers of tasks (30) and a number of vehicle (4)

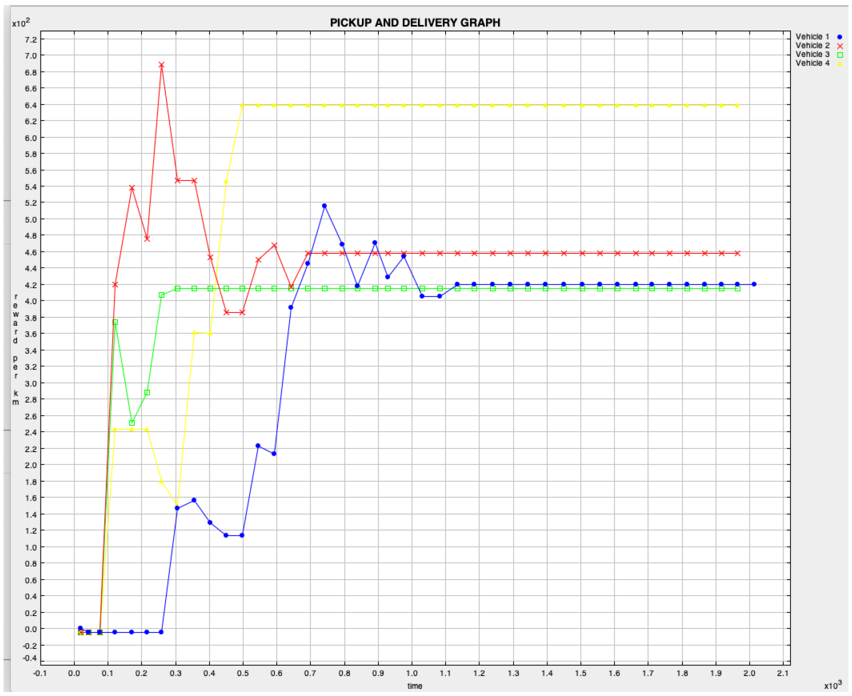

Figure 1: 30 tasks - 4 vehicles

3.2.1 Observations

We can clearly see from the figure 1 that a centralized agent is capable of coordinating in a more efficient manner the load and the travelling distances between the different vehicles in order to maximize the reward. However, as seen in the figure, vehicle has been carrying most of the tasks compared to the other vehicles.