Excercise 4
Implementing a centralized agent

Group №: Student 1, Student 2

6 novembre 2018

1 Solution Representation

1.1 Variables

We organized our program with different classes that provide a level of abstraction in order to perform the different mutations. Here is the implementation of our program in the Figure 1 :

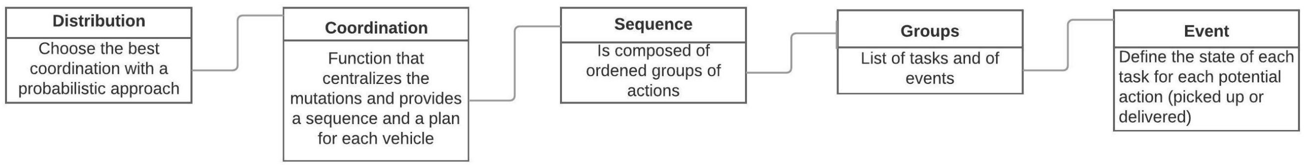
Distribution
Choose the best coordination with a probabilistic approach
Coordination
Function that centralizes the mutations and provides a sequence and a plan for each vehicle
Sequence
Is composed of ordened groups of actions
Groups
List of tasks and of events
Event
Define the state of each task for each potential action (picked up or delivered)

Figure 1 – Finishing the outer contour

1.2 Constraints

The constrains in our representation are :
— When performing the tasks in a group, a vehicle is never empty and always carry a minimum of one task.
— The pickup action of a task is always before the delivery of the task. Hence, the event pick up is always before the event delivery.
— The total number of groups in a Coordination is equal to the number of tasks that need to be executed.

1.3 Objective function

Our objective function is to minimize the total cost of a coordination. It is calculated using the distance that each vehicle performs during and between all the Events of a Sequence and the cost per km of a vehicle.

2 Stochastic optimization

2.1 Initial solution

We generate our initial solution by creating a 1 sequence for each vehicle and assigning randomly the tasks to each vehicle. In each sequence, we then create a group for each task assigned and place the tasks randomly in the groups. We then randomly order the events in each group of tasks.

2.2 Generating neighbours

We generate the neighbours by mutating a Coordination object. The mutation function has 6 different possible mutations :
— Switching sequences between 2 vehicles.
— Removing a Group of tasks from a Sequence and placing it in another sequence.
— Switching the order of 2 Groups of tasks in a Sequence.
— Removing a task in a Group and placing it in another Group of the same Sequence.

— Switching the order of 2 Events in a Group of tasks.

Each of these possible mutations occur with a certain probability when creating a neighbour. This probability is controlled by a general parameter γ.

2.3 Stochastic optimization algorithm

Our algorithm initialize a random Coordination of vehicles and creates a number N of neighbours from this initial Coordination. It then compares all the neighbours with the original one and selects the best one of them with a probability p. Else it goes selects a random Coordination from the neighbours. This allows for a stochastic movement in the search space. We store the best solution that we generate at the end.
This algorithm is enabled when the type of the algorithm is set to **"TYPE1"** in the section algorithm of *agent.xml*.

2.4 Steady State Genetic algorithm

We wanted to implement an algorithm that could explore the search space better than SLS. So we implemented a Steady state Genetic algorithm that works as follow :

— We initialize a population of Coordination randomly across the search space.
— We select randomly an individual IND1 to mutate according to a roulette selection. We use a reverse roulette selection to select the individual IND2 to replace.
— We mutate IND1 and if the resultant child is better than the parent, we replace IND2 with the mutant.
— This will make the average cost to decrease across generation.
— After a certain number of iteration we generate the plans for the best Coordination.

This algorithm is enabled when the type of the algorithm is set to **"TYPE2"** in the section algorithm of *agent.xml*.

3 Results

3.1 Experiment 1 : Model parameters

3.1.1 Setting

We are currently doing our tests on the topology **england** with **15 tasks** and **4 vehicles**. The tasks are first and foremost distributed randomly to each vehicles and then, the algorithm start to reorganize the tasks for each vehicles.

The parameters that we are analyzing are the **total cost of our simulation**. We didn't considered the rewards for each tasks there because the final reward will remain constant during the simulation. Hence, we were trying to reduce the total cost of our system. We were also trying to reduce the **computation time our program** is taking to provide us the plans for each vehicles by reducing the loops of our system.

3.1.2 Observations

The results we observe are the following. We performed 3 experiments for each algorithm. The number of tasks is set to 15 and the number of vehicles is 4. The average results for each set of experiments are in the table below :

Type of algorithm	Time to compute	Cost for the initialization	Cost for the last iteration	Number of calculations
Stochastic Local Search	1246 milliseconds	36812.33	25125.16	nbNeighbours*nbIterations = 20*10000 = 200000
Steady state GA	1616 milliseconds	31173.33	16019.83	populationSize*nbGeneration = 80*100 = 8000

Table 1 – Comparing Stockastic local search with Genetic Algorithm

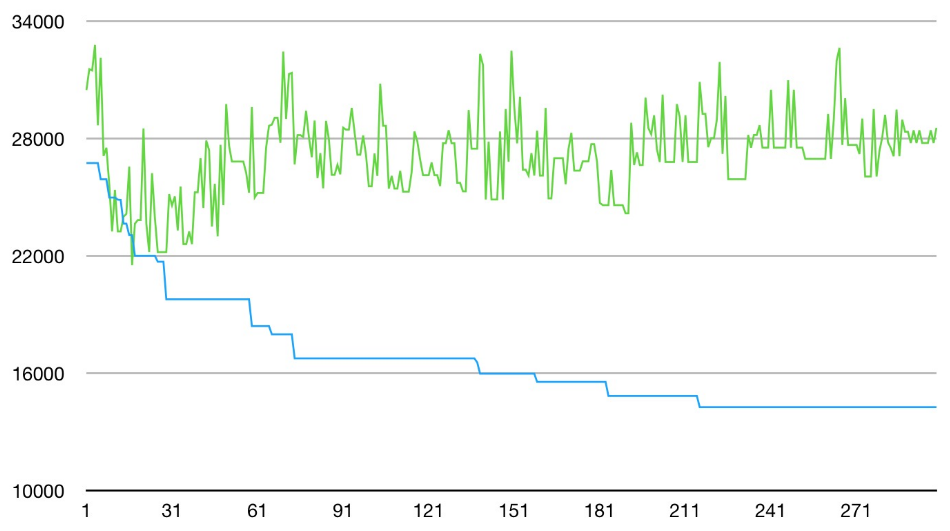

FIGURE 2 – Example comparing the behavior of SLS and SSGA

We can clearly see from the results that the genetic algorithm performs way better. This is due to the SLS getting stuck in local minimas. The genetic algorithm avoids better getting stuck by covering initially a much larger fraction of the search space.

For SLS we have 2 model parameters : mutationRate and probLimit, controlling respectively the amount of mutation for each new mutant and the probability to follow the gradient at each iteration. We also perform 3 experiments for each combination of these parameters.

mutationRate\probLimit	0.01	0.3	0.6	0.9
0.01	24150.16	23726.33	24311.83	28225.33
0.5	20720.5	24887	24194	27304
1.0	19137.67	23084.67	25099.67	30238

TABLE 2 – Influence of mutation rate and stochastic probability

3.2 Experiment 2 : Different configurations

3.2.1 Setting

We are currently doing our tests on the topologys **france**, with **10 tasks**, **15 tasks**, **25 tasks** and **2 vehicles**, **4 vehicles**, **6 vehicles**. We are keeping the same number of calculation for **eachs of the algorithms** and we will only **give the cost for the last iteration** (all of those test were computed in less than 1 minute).

3.2.2 Observations

	10 Tasks	15 Tasks	25 Tasks
2 Vehicles	23175.0	66210.0	120030.0
4 Vehicles	30075.0	57445.0	110230.0
6 Vehicles	22705.0	54650.0	107875.0

Table 3 - Cost for the Steady State Genetic Algorithm

	10 Tasks	15 Tasks	25 Tasks
2 Vehicles	19945.0	44435.0	83895.0
4 Vehicles	21030.0	44170.0	88355.0
6 Vehicles	20285.0	41565.0	89405.0

Table 4 -Cost for the Stochastic Algorithm