Excercise 4
Implementing a centralized agent

Group №23: Jean-Thomas Furrer, Emily Hentgen

October 29, 2017

1   Solution Representation

1.1   Variables

We use two variables for representing a plan resulting from the Stochastic Local Search (SLS) algorithm:

Map<Vehicle, TaskAction> *vehicleToFirstTaskAction*: maps a vehicle to the first task action it has to perform (necessarily a pick up, may be `null` if the vehicle picks up no task at all)

Map<TaskAction, TaskAction> *taskActionToTaskAction*: maps a task action to the next task action performed by the same vehicle; used jointly with *vehicleToFirstTaskAction* in order to know which vehicle handles which set of tasks

1.2   Constraints

We enforce X constraints:

– The load of vehicle must not exceed its capacity.

– The delivery of a task cannot happen before the pickup of this same task.

1.3   Objective function

The objective function we optimize is the total travel cost resulting from all pickups and deliveries of all vehicles.

2   Stochastic optimization

2.1   Initial solution

One possible initial solution consists of assigning all tasks to the vehicle with the largest capacity, in any order. This allows to check the problem is actually solvable: if the weight of the heaviest task exceeds the capacity of the largest vehicle, then there is no solution to this pick up and delivery problem.

In **vehicleToFirstTaskAction**, we add an entry mapping the largest vehicle to a any task pickup, and map all the other vehicles to the `null` value. In **taskActionToTaskAction**, we begin by mapping the first task pickup to its corresponding task delivery. We then map this task delivery to any the pickup of any task in the set of remaining tasks, map the latter task pickup to its corresponding task delivery and so on. Each time a task is selected, we remove it from the set of remaining tasks: the mapping **taskActionToTaskAction** is complete once the set of remaining tasks is empty.

2.2   Generating neighbours

We use almost the same two transformations as in the Pick Up and Delivery Problem where each vehicle can carry only one task at a time: a neighbour plan can either be generated by swapping the order of two tasks for the same vehicle, or by giving the first task of one vehicle to another.

We firstly generate a potential neighbour plan without enforcing the constraints. It is only once it has been generated that we check whether the constraints are respected: if this is the case, we add it to the set of neighbouring plans, otherwise, we discard it.

2.3 Stochastic optimization algorithm

3 Results

3.1 Experiment 1: Model parameters

3.1.1 Setting

3.1.2 Observations

3.2 Experiment 2: Different configurations

3.2.1 Setting

3.2.2 Observations