# Exercise 4
# Implementing a centralized agent

Group 39 : Hugo Bonnome, Pedro Amorim

November 6, 2016

## 1 Solution Representation

Unless noted otherwise, values that are not defined explicitely here use the definitions provided in the reference paper.

### 1.1 Variables

The following variables are used to define the CSP:

- $P$ - a set of plans, one for each vehicle.

$$P = \{p_1, p_2, \cdots, p_{N_V}\}$$

- $p_i$ - a doubly linked list representing the sequence of actions in a particular plan i.e. the deliveries and pickups that the vehicle should execute.

$$p_i = (u_1 \rightarrow u_2 \rightarrow d_2 \rightarrow \cdots | u_i \in U \wedge d_i \in D)$$

- $U$ - the set of pickups associated to each task.

$$U = \{u_1, u_2, \cdots, u_{N_T}\}$$

- $D$ - the set of deliveries associated to each task.

$$D = \{d_1, d_2, \cdots, d_{N_T}\}$$

### 1.2 Constraints

1. All tasks must be delivered i.e. $p_1 \cup p_2 \cup \cdots \cup p_{N_V} = U \cup D$
2. Max load of a given plan must be under the carrying capacity of the vehicle i.e. $maxLoad(p_i) \leq maxCapacity(v_i) \ \forall_{p_i \in P}$
3. A plan can only deliver a task it has previously picked up i.e. $u_i \in predecessors(d_i) \ \forall_{u_i, d_i \in p_i}$
4. A given action can only appear once in the set of all the plans.
5. If there is a pickup action in a plan then there is a corresponding delivery action in the same plan and vice-versa. i.e. $u_i \in p_i \leftrightarrow d_i \in p_i \ \forall_{p_i \in P,\ u_i \in p_i,\ d_i \in p_i}$

### 1.3 Objective function

Let us first define the total cost of a vehicle $v$ with a given plan.

$$c_v = \sum_{a_i \in p_v} (dist(a_i)) \cdot cost(v)$$

The total cost of the company is then defined by the sum of the costs of each vehicle in the company i.e.

$$\sum_{v_i \in V} c_{v_i}$$

## 2 Stochastic optimization

### 2.1 Initial solution

The initial solution is generated simply by giving all the tasks to the biggest vehicle.

### 2.2 Generating neighbours

Two operators are used to generate neighbours:

- *Changing vehicle*: give a pair of delivery and pickup tasks to another vehicle by appending them to the plan of the other vehicle. Weight constraints should not be violated by this change.
- *Changing task order*: change the order of two tasks in the plan of a vehicle while making sure that the order does not violate constraint 3; namely the fact that a delivery task cannot come before the associated pickup task.

### 2.3 Stochastic optimization algorithm

The algorithms works by first generating an initial solution that satisfies all the constraints. Then locally similar solutions are generated and the best one among them is chosen. This is repeated until a given number of iterations is reached. Since it uses an hill climbing heuristic, the algorithm is susceptible to getting stuck in a local minima and thus there is no guarantee of finding an optimal solution. To reduce the chances of getting stuck in a local minima, local improvements are only chosen with a certain probability.

## 3 Results

### 3.1 Experiment 1: Parameter $p$

#### 3.1.1 Setting

- Topology: England.
- Task distribution: uniform probability, constant reward and constant weight.
- Task number: 30.
- Vehicles : 4 vehicles with all the same characteristics.

- $p \in \{0, 0.8, 1\}$
- Number of iterations: 20000.

#### 3.1.2 Observations

- With $p = 0$ the total cost of the plan is equal to: 58353.0 CHF. This can be explained by the fact that the solution is chosen in a completely random manner as long as it satisfies the constraints. It is thus far from being efficient.
- With $p = 0.8$ the total cost of the plan is equal to: 18518.0 CHF. This is a much better result and it can easily explained by the fact that the algorithm can now do an hill-ascent on the range of solutions.
- With $p = 1$ the total cost of the plan is equal to: 28943.6 CHF. The higher cost can be explained by the fact that the algorithm will now almost always converge to a local minimum since the random changes in the solution no longer occur.

### 3.2 Experiment 2: Different vehicle numbers

#### 3.2.1 Setting

Same setting as experiment 1; the difference lies in the number of vehicles $N_V \in \{1, 4, 10\}$ and $p = 0.8$

#### 3.2.2 Observations

- With $N_V = 1$ the total cost is equal to 28945.6 CHF. We get the same result as in experiment 1 when $p = 1$. This can be understood by the fact that the algorithm computes the best order of delivery when only one vehicle is available.
- With $N_V = 4$ the total cost is equal to 18518.5 CHF. In this setting the vehicles share the task set and as such can be more efficient than a lone vehicle. This does not mean however that the tasks are distributed equally since the only metric of performance is the total cost.
- With $N_V = 10$ the total cost is equal to 30237.5 CHF. We can see that having more vehicles does not mean having a lower cost. In this setting many vehicles do not even act for the whole duration of the simulation. This is understandable by the fact that what is being optimized here is the total cost and not the speed of delivery of all tasks.

The time complexity of the algorithm grows linearly with the number of vehicles present in the simulation and the number of tasks. This comes from the fact that the *generateNeighbours* function has two operations:

1. Changing the order of tasks within the plan of a given vehicle. This operation grows linearly with the number of tasks.
2. Giving a specific task to another vehicle. This grows linearly with the number of vehicles to consider.