Excercise 4
Implementing a centralized agent

Group №: 272257, 272709

November 3, 2020

1 Solution Representation

We formalize the pickup and delivery problem as a *constraint optimization problem*. The *COP* is formally a tuple $< X, D, C, f >$ where X is a set of variables describing a plan, D the domain of the variables $x_i \in X$, C a set of constraints that a plan needs to fulfill and f a cost function that we are trying to minimize. A plan P is given by an assignment of the variables in X (where the constraints in C are satisfied). We call N_v the number of vehicles and N_t the number of tasks.

1.1 Variables

Any plan can be described by a given assignment of the 4 following variables (in out implementation, a plan is represented by the `plan` class):

1. **nextTask_v** : the $nextTask\_v$ variable is an array of size N_v where each element $nextTask\_v[i]$ represents the first task that the ith vehicle will perform.

   if $nextTask\_v[i] = j$ it means that the ith vehicle will start it's route by delivering the jth task

   if $nextTask\_v[i] = NULL$ it means that the ith vehicle has no task to deliver in the given plan.

2. **nextTask_t** : the $nextTask\_t$ variable is an array of size $2N_t$ where each element $nextTask\_t[i]$ represents the next task that the vehicle that the vehicle will pickup (in this array the pickup and delivery are represented as two distinct array element to allow for multiple tasks to be picked up by the agent)

   if $nextTask\_t[i] = j$ it means that the vehicle that delivered the ith task will deliver the jth task next.

   if $nextTask\_t[i] = NULL$ it means that the vehicle that delivered the ith has no task to deliver next.

3. **time** : the $time$ variable is an array of size $2N_t$ where each element $time[i]$ represents the position of the ith task in the plan of the vehicle delivering it in the plan (so if it is the first task delivered by some vehicle we would have $time[i] = 1$, again it is of size $2N_t$ to represent pickup an delivery as distinct actions)

4. **vehicle** : the $vehicle$ variable is an array of size $2N_t$ where each elements $vehicle[i]$ describes which vehicle will delivered the ith task

1.2 Cost function

Since we are looking for an optimal solution of our Constrained Optimization Problem (in the sense that it minimizes a cost function) we need to define our cost function.

Given :

$$\text{Distance between two cities } c_1 \text{ and } c_2\text{:} \qquad dist(c_1, c_2)$$
$$\text{Cost per kilometer for a given vehicle:} \qquad C_{km}(v)$$

We define the cost function $cost(P)$ as :

$$cost(P) = \sum_{v \in [1...N_v]} \overbrace{\left[C_{km}(v) \cdot \sum_{path} dist(c_1, c_2) \right]}^{\text{where } c_1 \text{ and } c_2 \text{ are two cites on vehicle } v\text{'s path}}$$

1.3 Constraints

Not all possible variable assignments correspond to a valid plan, a valid plan is a plan that satisfies all constraints $c \in C$, the constraints are the following:

$$\text{The next task after a given task } t \text{ cannot be itself} \qquad nextTask\_t[t] \neq t$$
$$\text{The time variable must be coherent} \qquad nextTask\_v[i] = j \Rightarrow time(j) = 1$$
$$nextTask\_t[i] = j \Rightarrow time(j) = time(i) + 1$$
$$\text{All tasks must be delivered} \qquad \#\ NULL \text{ values in } nextTask\_t$$
$$= \#\ non\text{-}NULL \text{ values in } nextTask\_v$$
$$\text{From the definition of vehicle} \qquad nextTask\_v(k) = j \Rightarrow vehicle(j) = k$$
$$\text{From the definitions of } nextTask\_t \text{ and vehicle} \qquad nextTask\_t[i] = j \Rightarrow vehicle[j] = vehicle[i]$$
$$\text{From the definitions of } nextTask\_v \text{ and vehicle} \qquad nextTask\_v[i] = j \Rightarrow vehicle[j] = vehicle[i]$$
$$\text{A vehicle cannot carry more than it's capacity} \qquad load(i) > capacity(k) \Rightarrow vehicle(i) \neq k$$

2 Stochastic optimization

Because of the high computational complexity of the problem we search for a solution using a *Stochastic Local Search method (SLS)* that can be described by the following algorithm :

2.1 Stochastic optimization algorithm

Algorithm 1: Stochastic Local Search for Constrained Optimization Problem

---
$S \leftarrow \text{GenerateInitialSolution}(< X, D, C, f >)$
repeat
    $A_{old} \leftarrow A$
    $N \leftarrow Succ(A_{old}, < X, D, C, f >)$
    $A \leftarrow Choice(N, f)$
until ***stable_solution*** $\vee$ ***timeout***;
return A

This algorithm is implemented in the `plan` function of the `CentralizedAssignement` class.

2.2 Initial solution

We initialize our *COP* with a naïve solution to the pickup and delivery problem where we a

2.3 Generating neighbors

The set *succ* of neighbors is generated (implementation in the `neighbours` function of class `Solution`) by generating every possible exchange of two tasks in the order of pickup/delivery of tasks and every possible change of assignment of vehicle per task. So each *succ* is a set of size $N_t! + N_v \cdot N_t$.

2.4 Choosing neighbors

In order to minimize to cost we choose the neighbor that will replace the current plan in the next iteration according to the following law (let $0 < p < 1$):

Algorithm 2: Choice(N,f)

With probability p:
return *plan $\in N$ such that f is minimal*
else
return *plan $\in N$ where plan is randomly selected*

---

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