

# Pickup and Delivery Reactive Agent: Implementation of MDP

Christophe Marciot & Titouan Renard

October 3, 2020

## 1 Problem definition

### 1.1 Markov Decision Process

In a MDP current state is know with certainty, but the reward of transition is not. A MDP is defined by :

$$\begin{array}{ll} \text{Where } s \text{ denotes a state and } a \text{ an action} & \\ \text{A reward function:} & \overbrace{R(s, a)} \rightarrow \mathbb{R} \\ \text{Where } s' \text{ denotes the state the action leads to} & \\ \text{A probabilistic state transition table:} & \overbrace{T(s, a, s')} = p(s'|s, a) \end{array}$$

The goal of the process is to find a policy  $\pi$  such that *the average reward is maximized*.

### 1.2 The Pickup and Delivery Problem

Agents exist in a static environment (a model of Switzerland's road network) described by a graph. Nodes of the graph are called *cities* and it's (weighted) edges are called *roads*.

The pickup and delivery problem is described by a series of tasks spread over the topology, the transportation tasks are described by:

1. Pickup city (and it's position)
2. Delivery city (and it's position)
3. Reward in CHF

### 1.3 Definitions

#### 1.3.1 Existing tables

The dataset usable for learning is described by two probability tables :

1.  $P_{table}(i, j)$  : the probability of a task for city  $j$  to be present in city  $i$
2.  $R_{table}(i, j)$  : the average reward given when a task is transported from city  $i$  to city  $j$

#### 1.3.2 State

It doesn't seem to make sense for the state to be anything other than **the city the agent is when it has no task**, this is because :

1. An agent can not have more than one task
2. When an agent has a task it cannot interrupt it
3. There is no difference between an agent in a city with no task because it succeeded or because it failed it's last task, it still has to make a decision about how to get another task

The set  $S$  containing all states is exactly the set of all cities.

#### 1.3.3 Action

An action consists in the agent either:

1. going to another city with no tax
2. taking a task in the city it's in (which amounts to moving to another city)

And always result in the agent being in a city (different or not from it's starting point, the agent can loop between two city if it maximizes reward) without a task, in another words in a (new) state.

The set  $A$  containing all actions is the set of all pairs of two cities on the network.

#### 1.3.4 Reward

Given a state  $s$  the agent is in and for an action  $a$ , the reward function for a single action is defined as follows :

$$R_{action}(s, a) = \begin{cases} 0 & \text{if the agent moves to a city without taking a contract} \\ R_{table}(i(a), j(a)) & \text{if the agent moves to a city after taking a contract} \end{cases}$$

#### 1.3.5 Reward

Where  $i(a)$  is the starting city of the action  $a$  and  $j(a)$  it's ending city, note that  $i(a) = s$ .

### 1.3.6 Discount factor

We have, in order to ensure the convergence of  $V(s)$  that :

$$\gamma \in [0, 1[$$

We want our system to optimize for time, not for number of actions take. Because an action may take more or less time. So let's define our  $\gamma$  as a function of an action :

$$\gamma(a) = \frac{\min_{\alpha \in \text{actions}}(t(\alpha))}{t(a)}$$

### 1.3.7 Probability of transition $p(s'|s, a)$

## 2 Solving the MDP

We denote *the value of a state  $s$*  as  $V(s)$ . This value represents "*the potential rewards from this state onwards*". In order to ensure  $V(s_i) < \infty \forall i$  (and make the problem solvable) we introduce a *discount factor*  $\gamma \in [0...1[$ .

$$V(s_i) = R(s_i) + \gamma \cdot V(T(s_i), a(s_i))$$