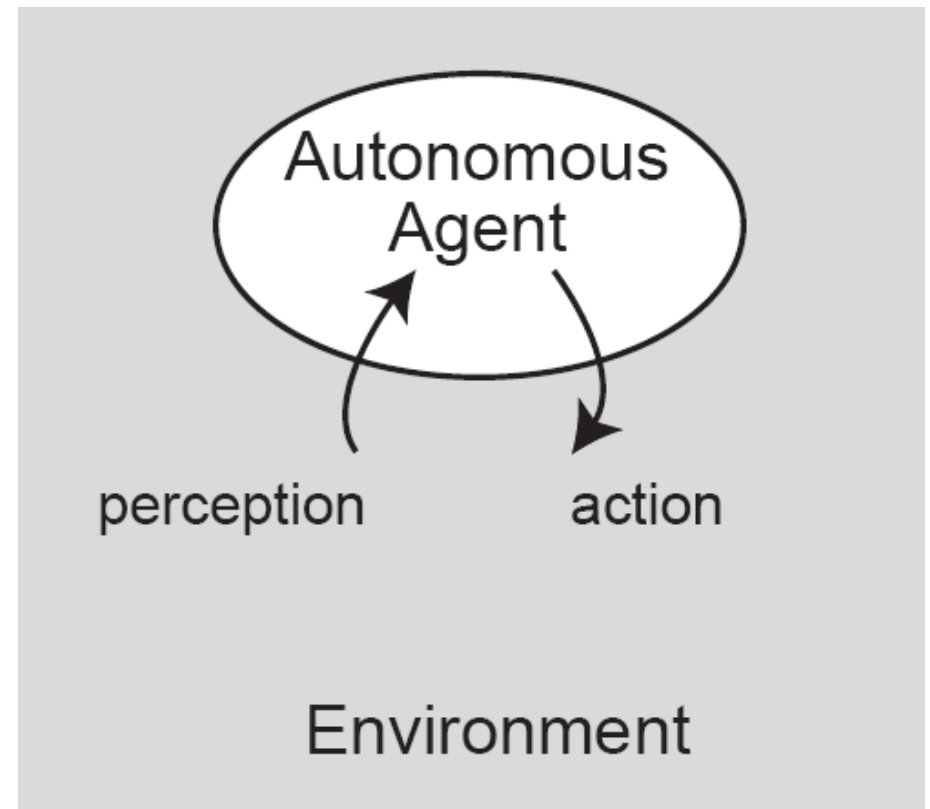


# The PDP with Reactive Agents

Intelligent Agents  
Course 2017/2018

# Reactive Agents

- Simple behaviours that react to stimuli from the environment
- Robot-like:
  - « Stimuli »  
from sensor  
« information »



# Our Reactive Agent Architecture

- Vehicles as reactive agents
- Percepts from the environment :
  - Route?
  - City?
  - Task?
- Actions the agent can take :
  - Move in a direction according to the topology
  - Pickup a task
  - Deliver a task

# Three Steps

1. Learn offline the actions to take in order to optimally search and deliver tasks; => MAIN PART OF THE EXERCIZE
2. Using the created table (the strategy), travel through the network;
3. As soon as a task has been picked up, deliver it on the shortest path (given)

# Assumptions (1)

- The vehicle starts from its home city
- When the vehicle arrives in a city, it finds out whether a task is available or not in that city. The vehicle sees at most one task!
- If task, agent can decide to:
  - Deliver it
  - Give it up and continue to a neighbour city

# Assumptions (2)

- There exists a probability distribution of the tasks
- An vehicle can transport only one task
- Must deliver it, and on the shortest path (given)

# Applying reinforcement learning

## (Markov Decision Processes - MDP)

- Goal:
  - learn optimal strategy to move in the network and deliver tasks
  - i.e. react optimally on the basis of a probability distribution of the tasks in the network
- MDP: offline, before agent travels!

# Existing two tables

- $P(i,j)$ :
  - probabilities  $p_{ij}$  that in city  $i$ , there is a task for city  $j$
- $AR(i,j)$ :
  - average reward  $r_{ij}$  for a task to be transported from city  $i$  to city  $j$
- Beforehand, tasks  $t_{ij}$  have been created with the probability  $p_{ij}$  and a variation reward around  $r_{ij}$



# Reinforcement Learning

- Define on paper (and in your report!):
  - A state representation of the world,
  - the actions for the transitions between those states,
  - with the corresponding rewards,
  - and the probability of the transition.

# Algorithm

- Compute  $V(S)$  by value iteration:

initialize  $V(S)$  arbitrarily

loop until good enough

  loop for  $s \in S$

    loop for  $a \in A$

$$Q(s, a) \leftarrow R(s, a) + \gamma \sum_{s' \in S} T(s, a, s') V(s')$$

$$V(s) \leftarrow \max_a Q(s, a)$$

# Data Structures (1)

- $V(S)$ :
  - vector indicating the best action to take from a specific state (with its corresponding accumulated value)
- $R(s,a)$ :
  - table that indicates the rewards for taking action  $a$  being in state  $s$
- $T(s,a,s')$ 
  - $= p(s'|s,a)$ , i.e. probability to arrive in state  $s'$  given that you are in state  $s$  and that you take action  $a$
- Discount Factor: between 0 and 1

# Data Structures (2)

<b><math>Q(s, a)</math></b>	<b>State s1</b>	<b>State s2</b>	<b>...</b>
<b>Action a1</b>	$Q(s1, a1)$	$Q(s1, a1)$	...
<b>Action a2</b>	$Q(s1, a2)$	$Q(s2, a2)$	...
<b>...</b>	...	...	...

- Initialize to 0

# To Do

- Implement a simple reactive agent
- Define:
  - state representation  $s$
  - possible actions  $a$ ,
  - reward table  $R(s,a)$
  - probability transition table  $T(s,a,s')$
- Implement the offline reinforcement learning algorithm
- Run simulations, analyze the performance of your agent and test limit cases