

# Lesson 5: Markov Decision Processes

Emmanuel Hyon<sup>1</sup>

<sup>1</sup>Université Paris Nanterre, France  
LIP6, Sorbonne Université, CNRS, France

RESCOM Summer School, June 2019

# Outline

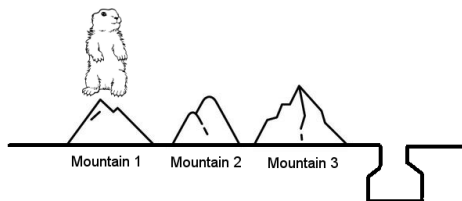
- 1 Introduction
- 2 Markov Processes
- 3 Elements of Markov Decision Processes
- 4 Resolution of Markov Decision Processes
- 5 Related models (extension of MDP)
- 6 Bibliography

# Outline

- 1 Introduction
  - Stochastic Systems
  - Controlled stochastic systems

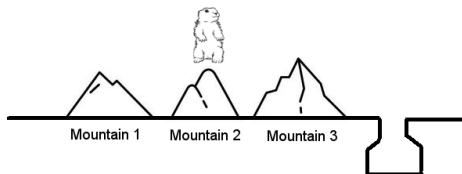
# Introductory Example

You are on vacation and every morning you observe a groundhog and you record its position.



# Introductory Example

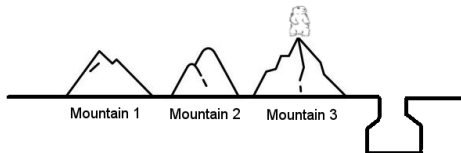
You are on vacation and every morning you observe a groundhog and you record its position.



The groundhog moves during the night.

# Introductory Example

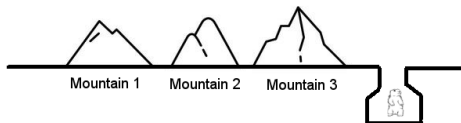
You are on vacation and every morning you observe a groundhog and you record its position.



The mountain on which the groundhog changes.

# Introductory Example

You are on vacation and every morning you observe a groundhog and you record its position.



The behaviour of the groundhog seems you random.

# Introductory Example (ctd.) Dynamical system

We introduce  $x_k$  which is the record of the position at day  $k$ .

The sequence of the daily records is called a *path* or an *episode*.  
It can take several values :

- (M1, M2, M3, M3, M2, M3, M2, M1, M1,...)
- (M1, M2, M2, M1, M1, M2, M2, M2, M2,...)
- (H, H, H, H, H, H, H, H, H, H, H,.....)
- (H, M3, M2, M3, H, H, H, M3, M2, M1,...)

The system you observe is a *dynamical system*:

the “*state*” of the system evolves during time, the state of the next day depends on a transition function:

$$x_{k+1} = f_k(x_k, x_{k-1}, \dots, x_1).$$



# Introductory Example (ctd.)

## Stochastic dynamical system

When the transition from a state at day  $k$  to another state at day  $k + 1$  follows a stochastic transition function it is a *stochastic system* :

$$x_{k+1} = f_k(x_k, x_{k-1}, \dots, x_1, \varepsilon(\omega))$$

with  $\varepsilon(\omega)$  is the hazard.

-The transition can depend on the states :

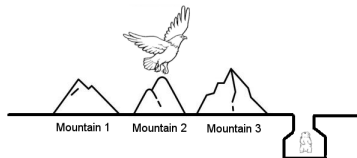
e.g. if  $k$  is a day corresponding of a winter day there few chances that the state will be different of  $H$  (home).

-The transition can depend on event that are not considered in the states:

e.g. the hazard is the presence of an eagle in the sky.

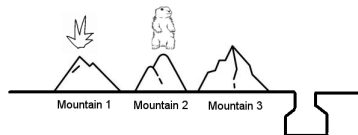
## A second example

You have now the ability to place on a mountain an eagle or herbs.



## A second example

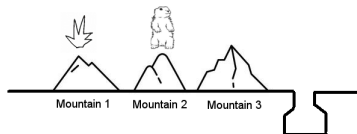
You have now the ability to place on a mountain an eagle or herbs.



You change the behaviour of the groundhog.  
You control the behaviour of the system.

## A second example

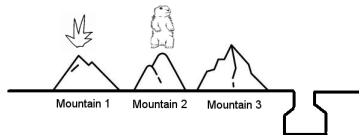
You have now the ability to place on a mountain an eagle or herbs.



You have a *controlled stochastic system*.

## A second example

You have now the ability to place on a mountain an eagle or herbs.



The (stochastic) transition function depends on states and actions:

$$x_{k+1} = f_k(x_k, x_{k-1}, \dots, x_1, a_k, \dots, a_{k-1}, \varepsilon(\omega)).$$

# Controlled stochastic system (ctd)

Two axis for studies of controlled stochastic system

- Keeping some features of the system during its life,  
e.g Let the groundhog alive.  
⇒ mainly in *automatic* field.
- Coupled with the evolution of the system it exists rewards (or costs),  
e.g. you receive a reward every time the groundhog is on mountain  $M1$ .  
⇒ mainly in *Stochastic optimisation* field.

We are interested here by the **stochastic optimisation aspects**.

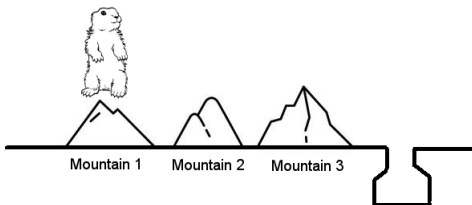
# Outline

## 2 Markov Processes

- Markov Chain
- Markov Reward Processes
- Markov Decision Processes: an example

## Example 1 and Markov properties

Consider again the first example :

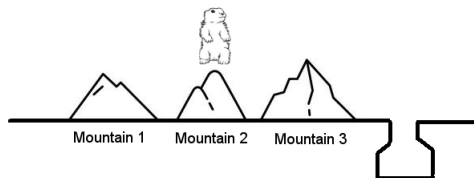


a) We assume that we know the transition probabilities (*i.e.* the stochastic transition function is known).



## Example 1 and Markov properties

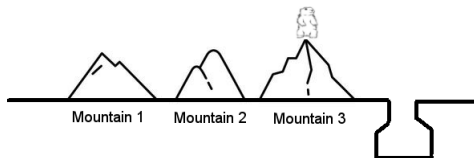
Consider again the first example :



b) We assume that the transition probabilities only depend on the current state.

## Example 1 and Markov properties

Consider again the first example :



c) We also assume that we know the initial place.

# Markov Chain

We describe the behaviour of the groundhog by a Markov Chain that is a memoryless random process with Markov property.

- Let  $x_k$  be the position of the groundhog at day  $k$ .
- The stochastic transition function depends only on the hazard and on the current state :

$$x_{k+1} = f(x_k, \varepsilon_{k+1}(\omega))$$

## Definition

A (finite) *Markov Chain* is a tuple  $\mathcal{X}, \mathcal{P}$  :

- $\mathcal{X}$  is the state space : a (finite) set of states.
- $\mathcal{P}$  is the state to state transition probability (given by a transition matrix):

$$\mathcal{P}_{x,x'} = \mathbb{P}[x_{k+1} = x' \mid x_k = x].$$

## Example 1 (ctd.)

Assume transition probabilities on the Example 1 such that:

- If the groundhog stays on the first mountain, then it can
  - remain on the first mountain with probability 0.25;
  - leave to go on the second mountain with probability 0.5;
  - leave to go on the third mountain with probability 0.25.
- If the groundhog stays on the second mountain, then it can
  - remain on the second mountain with probability 0.2;
  - leave to go on the first mountain with probability 0.4;
  - leave to go on the third mountain with probability 0.4.
- If the groundhog stays on the third mountain, then it can
  - remain on the third mountain with probability 0.3;
  - leave to go on the first mountain with probability 0.4
  - leave to go on the second mountain with probability 0.3.

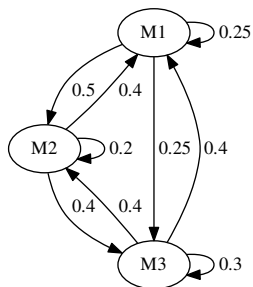
## Example 1 (ctd.): Model

The *state space* is  $\{M1, M2, M3\}$ , where  $Mi$  is the mountain  $i$ .

The transition matrix is

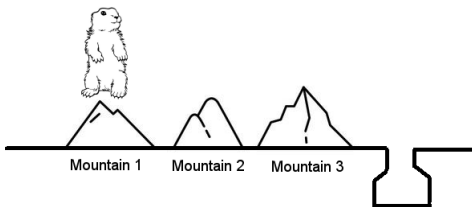
$$\mathcal{P} = \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.4 & 0.2 & 0.4 \\ 0.4 & 0.3 & 0.3 \end{bmatrix}$$

The transition graph is



## Example 1 with reward

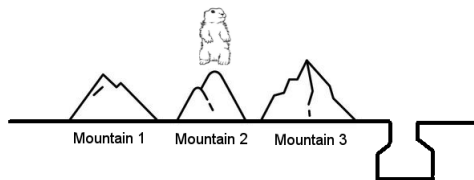
Consider again the first example :



In state  $M1$  we are *very happy* we receive a satisfaction of 10,

## Example 1 with reward

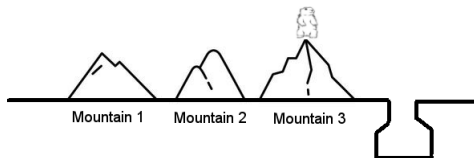
Consider again the first example :



In state  $M2$  we are *happy* we receive a satisfaction of 1,

## Example 1 with reward

Consider again the first example :

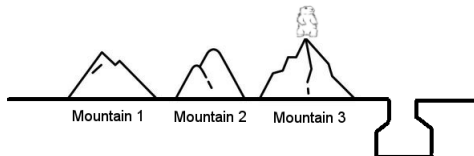


In state  $M3$  we are *unhappy* we receive a satisfaction of 0.1,



## Example 1 with reward

Consider again the first example :



Satisfaction is the reward of a state.

# Markov Reward Process (MRP)

- Let  $r_k$  be the reward collected at day  $k$ .
- The reward is given by a function  $r_k = \mathbb{E}(\mathcal{R}(x_{k+1}, x_k, \varepsilon_{k+1}))$

## Definition

A *Markov Reward process* is a tuple  $\mathcal{X}, \mathcal{P}, \mathcal{R}$  :

- $\mathcal{X}$  is the state space : a set of states.
- $\mathcal{P}$  is the state to state transition probability:  
 $\mathcal{P}_{x,x'} = \mathbb{P}[x_{k+1} = x' \mid x_k = x]$ .
- $\mathcal{R}_k$  is a reward function  $\mathcal{R}_k(x_{k+1}, x_k, \varepsilon)$ .

# Collected rewards

We are interested by the return of a MRP: the sum of the rewards collected along a path :

Define  $V_k(x)$  the sum of collected rewards from state  $x$  at day  $k$  :

$$V_k(x) = \sum_{i=k}^{i=N} r_i .$$

# Collected rewards

We are interested by the return of a MRP: the sum of the rewards collected along a path :

Define  $V_k(x)$  the sum of collected rewards from state  $x$  at day  $k$  :

$$V_k(x) = \sum_{i=k}^{i=N} r_i.$$

Path:

(M1, M2, M3, M3, M3, M3)

(M1, M2, M3, M2, M1, M2)

(M1, M2, M1, M1, M2, M1)

Return: ( $V_1(M1)$  with  $N = 6$ )

$10 + 1 + 0.1 + 0.1 + 0.1 + 0.1 = 11.4$

$10 + 1 + 0.1 + 1 + 10 + 1 = 23.1$

$10 + 1 + 10 + 10 + 1 + 10 = 42$

## Expected collected rewards

- It is more interesting to use the expectation of the sum of the collected rewards.
- This takes better into account the stochastic feature of the system (a path occurs with a given probability).
- We evaluate the expected cost but we experience a trajectory cost.

## Expected collected rewards

- It is more interesting to use the expectation of the sum of the collected rewards.
- This takes better into account the stochastic feature of the system (a path occurs with a given probability).
- We evaluate the expected cost but we experience a trajectory cost.

Path:	Return
(M1, M2, M3, M3, M3, M3)	11.4 with probability 0.0054 $0.5 \times 0.4 \times 0.3 \times 0.3 \times 0.3 = 0.0054$
(M1, M2, M3, M2, M1, M2)	23.1 with probability 0.012 $0.5 \times 0.4 \times 0.3 \times 0.4 \times 0.5 = 0.012$
(M1, M2, M1, M1, M2, M1)	42 with probability 0.01 $0.5 \times 0.4 \times 0.3 \times 0.3 \times 0.3 = 0.01$
Expected reward : 0.00996	

# Objectives

But actually it exists more realistic objectives:

## Discounted reward

Let  $\theta \in [0, 1)$  be the discount factor. The *finite discounted reward* is:

$$V_N^\theta(x) = \mathbb{E} \left[ \sum_{k=0}^{k=N} \theta^k r_k \mid x_0 = x \right]$$

Economic relevance: interest rate can be taken into account.

Practical relevance: future rewards have different weights according to  $\theta$ .

Mathematical relevance: convergence of the sum.

# Objectives

But actually it exists more realistic objectives:

## Average reward

In an infinite horizon problem, the average reward is defined by:

$$\rho(x) = \lim_{N \rightarrow \infty} \frac{1}{N} \mathbb{E} \left[ \sum_{k=0}^{k=N} r_k \mid x_0 = x \right]$$

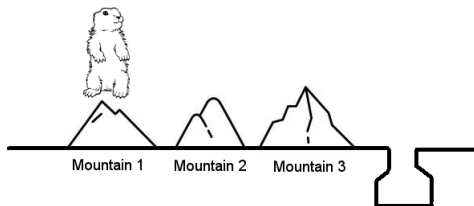
Practical relevance: same cost in average for each day.

Mathematical relevance: convergence.



## Example 1 with rewards and controls

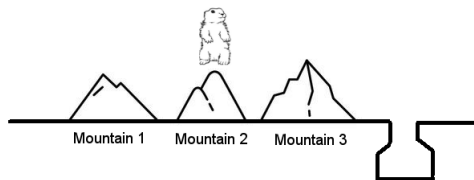
Consider *again* the first example :



You can put food on one mountain each day.

## Example 1 with rewards and controls

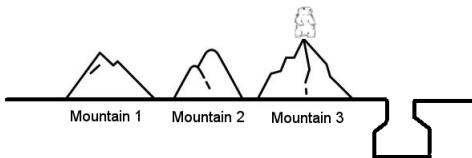
Consider *again* the first example :



The food modifies the behaviour of the groundhog.

## Example 1 with rewards and controls

Consider *again* the first example :



But putting food on a mountain has a cost:  
 $M1$  costs 5,  $M2$  costs 1 and  $M3$  costs 0.5.

# Example 1 with with rewards and controls :

a *dynamic control* problem

- You have a set of possible actions in each state.
- When you perform an action the stochastic transition function is modified (also the evolution of the system).
- When you perform an action the reward is modified.

We denote by  $r_k(x_k, a_k)$  the reward at day  $k$  when action  $a_k$  is triggered in state  $x_k$ .

The transition function depends on state, action and hazard:

$$x_{k+1} = f(x_k, a_k, \varepsilon_{k+1}).$$

The transition probability is now expressed with respect to the state and action:

$$\mathbb{P}(x_{k+1} = y | x_k = x, a_k = a) = p(y|x, a)$$

# Markov Decision Process

## Definition

A *Markov Decision Process* is a tuple  $\mathcal{X}, \mathcal{A}, (\mathcal{P})_a, \mathcal{R}$  :

- $\mathcal{X}$  is the state space : a set of states.
- $\mathcal{A}$  is the action space : a set of actions.
- $(\mathcal{P})_a$  is a collection of transition probabilities :  
 $(\mathcal{P}_a)_{x,x'} = \mathbb{P}[x_{k+1} = x' \mid x_k = x, a_k = a]$ .
- $\mathcal{R}_k$  is a reward function  $\mathcal{R}_k(x_{k+1}, x_k, a_k, \varepsilon)$

The Markov Decision Process (MDP) is a tool to solve :  
*Dynamic control* problems aka *Optimal Control* problems aka  
*Stochastic multistage* problems.

## Example 1: Temporal behaviour

It is important to express with precision the temporal decomposition of events and their effects in a time slot.

It is important to detail carefully when rewards are received and when action is triggered.

### Temporal decomposition of a day (method 1)

- ① Breakfast
- ② Decide where to place the food
- ③ Put the food on the chosen mountain
- ④ Go to sleep (*cost due to the action*)
- ⑤ Wake up
- ⑥ Observe the mountain and receive satisfaction (*cost due to nature*)

## Example 1: Temporal behaviour

It is important to express with precision the temporal decomposition of events and their effects in a time slot.

### Temporal decomposition of a day (method 2)

- ① Wake up
- ② Observe the mountain and receive satisfaction (*cost due to nature*)
- ③ Breakfast
- ④ Decide where to place the food
- ⑤ Put the food on the chosen mountain
- ⑥ Go to sleep (*cost due to the action*)

If we choose to let begin the slot at step 5 (wake), then satisfaction is not (directly) related with the action taken during the day.

## Example 1 expressed with MDP (ctd.)

State space :  $\{1, 2, 3\}$ , where  $i$  means mountain  $Mi$ .

Action space :  $\{0, 1, 2, 3\}$ , where  $i$  means put the food on mountain  $Mi$  (and 0 means that nothing is done)

Transition probabilities:

$$\begin{aligned} \mathcal{P}_0 &= \begin{bmatrix} 0.25 & 0.5 & 0.25 \\ 0.4 & 0.2 & 0.4 \\ 0.4 & 0.3 & 0.3 \end{bmatrix}, \mathcal{P}_1 = \begin{bmatrix} 0.5 & 0.25 & 0.25 \\ 0.6 & 0.2 & 0.2 \\ 0.5 & 0.4 & 0.1 \end{bmatrix}; \\ \mathcal{P}_2 &= \begin{bmatrix} 0.25 & 0.55 & 0.2 \\ 0.2 & 0.4 & 0.4 \\ 0.3 & 0.4 & 0.3 \end{bmatrix}, \mathcal{P}_3 = \begin{bmatrix} 0.1 & 0.2 & 0.7 \\ 0.0 & 0.1 & 0.9 \\ 0 & 0 & 1 \end{bmatrix} \end{aligned}$$



# Example 1 expressed with MDP (rewards)

Express the rewards now for state 1:

$$r(1|1,0) = 10, \quad r(2|1,0) = 1, \quad r(3|1,0) = 0.1$$

$$r(1|1,1) = 5, \quad r(2|1,1) = -4, \quad r(3|1,1) = -4.9$$

$$r(1|1,2) = 9, \quad r(2|1,2) = 0, \quad r(3|1,2) = -0.9$$

$$r(1|1,3) = 9.5, \quad r(2|1,3) = 0.5, \quad r(3|1,3) = -0.4$$

The expected reward  $r(x, a)$  is

$$r(x, a) = \sum_{y \in \mathcal{X}} R(y|x, a) \cdot p(y|x, a)$$

and then we have :

$$r(1,0) = 0.25 \cdot r(1|1,0) + 0.5 \cdot r(2|1,0) + 0.25 \cdot r(3|1,0) = 3.025$$

$$r(1,1) = 0.25 \cdot r(1|1,1) + 0.5 \cdot r(2|1,1) + 0.25 \cdot r(3|1,1) = -1.975$$

$$r(1,2) = 0.25 \cdot r(1|1,2) + 0.55 \cdot r(2|1,2) + 0.2 \cdot r(3|1,2) = 2.07$$

$$r(1,3) = 0.1 \cdot r(1|1,3) + 0.2 \cdot r(2|1,3) + 0.7 \cdot r(3|1,3) = 0.77$$

## Example 1 expressed with MDP (rewards ctd.)

We can count the rewards for the other states in the same way, we get

$$r(2,0) = 10 \cdot 0.4 + 1 \cdot 0.2 + 0.1 \cdot 0.4 = 4.24$$

$$r(2,1) = (10 - 5) \cdot 0.6 + (1 - 5) \cdot 0.4 + (0.1 - 5) \cdot 0.2 = 1.22$$

$$r(2,2) = 9 \cdot 0.2 - 0.9 \cdot 0.4 = 1.45$$

$$r(2,3) = 0.5 \cdot 0.1 - 0.4 \cdot 0.9 = -0.31$$

$$r(3,0) = 4.33$$

$$r(3,1) = 0.41$$

$$r(3,2) = 2.43$$

$$r(3,3) = -0.4$$

## Solving dynamic control problem

We want to optimise the discounted cost, we have

$$V^\theta = \mathbb{E} \left[ \sum_{k=0}^{k=7} \theta^k r(\mathbf{x}_k, \mathbf{a}_k) \right] \quad (1)$$

How optimise our problem ?

## Solving dynamic control problem

We want to optimise the discounted cost, we have

$$V^\theta = \mathbb{E} \left[ \sum_{k=0}^{k=7} \theta^k r(\mathbf{x}_k, \mathbf{a}_k) \right] \quad (1)$$

How optimise our problem ?

What can I do ?

# Solving dynamic control problem

We want to optimise the discounted cost, we have

$$V^\theta = \mathbb{E} \left[ \sum_{k=0}^{k=7} \theta^k r(\mathbf{x}_k, \mathbf{a}_k) \right] \quad (1)$$

How optimise our problem ?

What can I do ?

$\Rightarrow$  Choose the action to perform each day.

## Solving dynamic control problem

We want to optimise the discounted cost, we have

$$V^\theta = \mathbb{E} \left[ \sum_{k=0}^{k=7} \theta^k r(\mathbf{x}_k, \mathbf{a}_k) \right] \quad (1)$$

How optimise our problem ?

What can I do ?

$\Rightarrow$  Choose the action to perform each day.

What should I do ?

# Solving dynamic control problem

We want to optimise the discounted cost, we have

$$V^\theta = \mathbb{E} \left[ \sum_{k=0}^{k=7} \theta^k r(\mathbf{x}_k, \mathbf{a}_k) \right] \quad (1)$$

How optimise our problem ?

What can I do ?

⇒ Choose the action to perform each day.

What should I do ?

⇒ Determine the most profitable action (for the objective)

# Decision tree

- The set of possible paths including decisions and their possible consequences can be represented by a *decision tree*.
- A branch in the tree is a path in the system
- The costs of a trajectory in the system are the costs of a tree path.

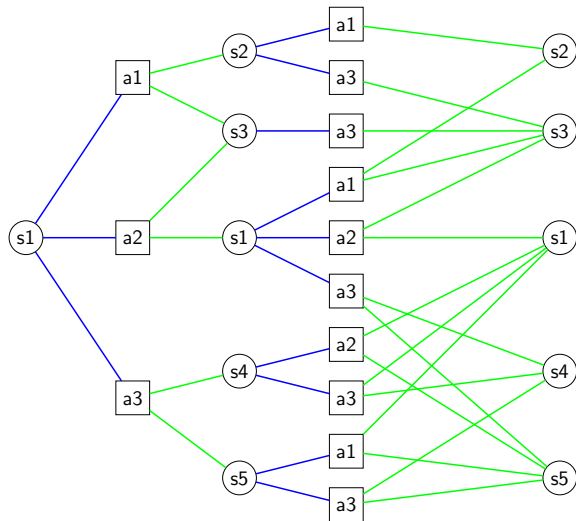
## An illustration

We assume an example with 5 states ( $s_1, s_2, s_3, s_4, s_5$ ) and three actions ( $a_1, a_2, a_3$ ).

It can be represented by a decision tree that depends on the transition probabilities and whose edges are valued by the rewards.



# Decision tree associated with the illustration



# Computation of optimal action

Let us detail now (with the help of the previous decision tree) some difficulties with the determination of optimal actions:

- ① An action has a long term effect.
  - Ex. 1 Once you are in state  $s_3$  you remain in it.
  - Ex. 2 Once you played  $a_1$  in state  $s_1$  then you have no possibility to reach states  $(s_1, s_4, s_5)$ .
- ② The benefit of an action depends on the following actions and their rewards.
  - Ex. 3 Assume that  $r(s_1, a_1) = 3$ ,  $r(s_1, a_2) = 2$ ,  $r(s_1, a_3) = 1$  and  $r(s_2, a_2) = r(s_2, a_3) = r(s_3, a_3) = 0$ .  
In  $s_1$  the action  $a_1$  will give a good immediate reward but no rewards in the future.
  - Ex. 4 Assume moreover that  $r(s_4, \cdot) = 3.5$  and  $r(s_5, \cdot) = 4$   
Action  $a_3$  in  $s_1$  is better than action  $a_2$  since the rewards in  $s_4$  or  $s_5$  are larger.

# Computation of optimal action (ctd.)

So the question is how evaluate branches ? and how evaluate sub branches ?

- Monte Carlo Simulations ?

But how do we choose the action ?

- Random choice ?
- The action that gives the best immediate reward ?
- We explore a part of the branch ?

The *Markov Decision Process*

can represent a dynamic control problem.

is a tool to solve the computation of optimal action using  
**dynamic programming**.

# Outline

## 3 Elements of Markov Decision Processes

- Model
- Objective function
- Decision Rule and Policy
- Value function

# Mathematical model of MDP

## Definition

A *Markov Decision Process* is a tuple  $\mathcal{X}, \mathcal{A}, (\mathcal{P})_a, \mathcal{R}$  :

- $\mathcal{X}$  is the state space : a set of states.
- $\mathcal{A}$  is the action space : a set of actions.
- $(\mathcal{P}_k)_a$  is a collection of transition probabilities :  
 $(\mathcal{P}_k)_a(x, x') = \mathbb{P}_k [x_{k+1} = x' \mid x_k = x, a_k = a]$ .
- $\mathcal{R}_k$  is a reward function  $r(x_k, a_k)$

When transition probabilities  $(\mathcal{P}_k)_a$  are the same for any  $k$  and when the rewards are the same for any  $k$  then the MDP is *stationary* (or homogeneous).

We only study here stationary MDP.

# State Space

The state space is the (countable) set of states it represents the set of values in which our system evolves.

It can be

- the *physical state* of the system.
  - e.g. the number of customers in a queue.
  - e.g. the number of objects that are connected to me.
- the *physical state* of the system and an additional information.
  - e.g. the mountain where the groundhog is *and* the presence of an eagle in the sky.
  - e.g. the number of customers in a queue *and* an information about the transmission rate.

Powell [1] [2] proposes a “definition” :

**State** represents the “physical” state and all additional information required for the decision.

## Action Space

The **Action Space** represents the set of actions available at the controller in a given state.

However this set can be different depending on the state. Hence you can have two different sets  $\mathcal{A}_x$  and  $\mathcal{A}_{x'}$  for different states  $x$  and  $x'$ .

Usually you represent the action space by a set  $\mathcal{A}$  that contains all possible actions ( $\mathcal{A} = \cup_x \mathcal{A}_x$ ) considering that there are impossible actions in given states.

To model impossible action, two possibilities

- 1 You consider only the actions of  $\mathcal{A}_x$  in  $x$ .
- 2 You consider all the actions in each state but you penalise impossible actions

## Transition function

Your transition function “has” the [Markov properties](#).

All the past is resumed in the present state.

$$x_{k+1} = f(x_k, a_k, \varepsilon_{k+1}).$$

Thus

$$\begin{aligned} \mathbb{P}(x_{k+1} = y \mid x_k = x, a_k = a, x_{k-1}, a_{k-1}, \dots, x_0, a_0) = \\ \mathbb{P}(x_{k+1} = y \mid x_k = x, a_k = a) \end{aligned}$$

also

$$\mathbb{P}(x_{k+1} = y \mid x_k = x, a_k = a) = p(y|x, a).$$



# Rewards

We consider only rewards which depend on the state and the action :  $r(x, a)$ .

Reward can be bounded or unbounded.

# Computer Representation

## Matrices

*How:* Storing a set of transition probabilities matrices in an array.

*When:* a complete description of the system is storable and available.

*Advantages:* Efficient especially coupled with sparse matrices

*Disadvantages* Large MDP can not be represented.

## Hash tables

*How:* Storing transition probabilities in an hash table. Key is the tuple  $(x_k, x_{k+1}, a_k)$ .

*When:* Simple system in which a complete description is available.

*Advantages:* Fast to program. Efficient.

*Disadvantages* Large and complex MDP can not be represented.

## List and function

*How:* Probability transitions are described by a function and not all the states are studied.

*When:* A compact form is desired.

*Advantages* Adapted for very large systems.

*Disadvantages* Require many computations and recalculations.

# Objectives

The different objectives that can be solved are :

## Finite Horizon

The gain on a finite horizon  $N$  is the expected sum of the reward on a path of length  $N$ :

$$V_N = \mathbb{E} \left( \sum_{i=0}^{N-1} r(x_i, a_i) \right)$$

where  $r(x_i, a_i)$  is the reward at step  $i$ .

# Objectives II

The different objectives that can be solved are :

## discounted finite Horizon

The gain on a discounted finite horizon  $N$  is the expected sum of the weighted rewards on a path of length  $N$ :

$$V_N^\theta = \mathbb{E} \left( \sum_{i=0}^{N-1} \theta^i r(x_i, a_i) \right)$$

where  $r(x_i, a_i)$  is the reward at step  $i$ ,  
where  $\theta \in [0, 1)$  is the discount factor.

# Objectives III

The different objectives that can be solved are :

## discounted infinite Horizon

The gain on a discounted infinite horizon is the expected sum of the rewards on an infinite path :

$$V^\theta = \mathbb{E} \left( \sum_{i=0}^{\infty} \theta^i r(x_i, a_i) \right)$$

where  $r(x_i, a_i)$  is the reward at step  $i$ ,  
where  $\theta \in [0, 1)$  is the discount factor.

# Objectives IV

The different objectives that can be solved are :

## total reward

The gain on a total reward is the expected sum of the rewards on an infinite path :

$$V = \mathbb{E} \left( \sum_{i=0}^{\infty} r(x_i, a_i) \right)$$

where  $r(x_i, a_i)$  is the reward at step  $i$ .

The sum of the total rewards may not converge.

This kind of models is often applied with a set of absorbing states.

# Objectives V

The different objectives that can be solved are :

## average

The gain on an average cost model is the limit of the Cesaro mean of the expected sum of the rewards on a path :

$$\rho = \lim_{N \rightarrow \infty} \frac{1}{N} \mathbb{E} \left( \sum_{i=0}^{N-1} r(x_i, a_i) \right)$$

where  $r(x_i, a_i)$  is the reward at step  $i$ .

This kind of objective has a gain which depends on the intrinsic Markov Chain.

# Decision rule

## Definition

We define a **decision rule** as the mapping that allows to select an action to perform.

Its classification depends on the information set that is used.

We denote by  $\pi_k$  the decision rule that is applied at step  $k$ .

A decision  $\pi_k$  can be :

<i>History dependent</i>	Depends on the <u>complete</u> past (states, actions)
<i>State dependent</i> (or <i>Markov</i> )	Depends on the <u>current</u> state

Thus a **Decision rule** is a mapping  $\pi$   
from the history to an action  $\pi : \mathcal{H} \mapsto A$   
from the state to an action  $\pi : S \mapsto A$



## Decision rule (ctd.)

Another classification is made according to the manner the decision is returned

<i>Deterministic</i>	The rule defines a single action
<i>Random</i>	The rule defines a probability on an action.

Thus a **decision rule** can define

a *single action*  $\pi : \mathcal{H} \mapsto a$

a *probability* with which the action is chosen  $\pi : \mathcal{H} \mapsto \mathbb{P}(a)$

## Decision rule (ctd.)

Thus we have four classes of decision rules.

- ① Random History dependent,
- ② Deterministic History dependent,
- ③ Random Markov,
- ④ Deterministic Markov,

Random history dependent is the more general.

Deterministic Markov is the more specific and is included in Random History dependent.

# Policy

Usually the terms *policy* and *decision rule* are considered as synonym.

However, I prefer make a distinction between them for the sake of clarity

## Definition

A **Policy**  $\Pi$  is a sequence of decision rules

$$\Pi = (\pi_1, \pi_2, \dots, \pi_n, \dots).$$

## Definition

A decision rule can be *stationary*: the same rule is applied in each slot

$$\Pi = (\pi, \pi, \dots, \pi).$$

# Value function

How to evaluate the reward of a policy ?

For this we define **value function**.

## Definition

A value function is defined for a fixed policy and a fixed initial state  $x \in \mathcal{X}$  and returns the total gain (for the criteria considered) collected following the policy  $\Pi$  from the initial state.

For a fixed  $\Pi$  we have a function (or a vector) such that

$$V^{\Pi} : \mathcal{X} \mapsto \mathbb{R}.$$

## Back to the groundhog

The value function have a value for each of the state  $M1$ ,  $M2$ ,  $M3$ .

## Value function (ctd.)

There exists a different value function expression for each of the objective.

### Value Function for finite horizon problem

Let  $N$  be the horizon. The finite horizon value function is denoted by  $V_N^\Pi(x)$  with

$$V_N^\Pi(x) = \mathbb{E}^\Pi \left( \sum_{k=0}^{N-1} r(x_k^\Pi, \pi_k(x_k^\Pi)) \mid x_0 = x \right).$$

### Value Function for discounted cost

The discounted cost value function is  $V_\theta^\Pi$  such that  $\forall x \in \mathcal{X}$  :

$$V_\theta^\Pi(x) = \mathbb{E}^\Pi \left( \sum_{k=0}^{\infty} \theta^k r(x_k^\Pi, \pi(x_k^\Pi)) \mid x_0 = x \right).$$

## Back to example 1

I will define a (Markov deterministic stationary) policy such that the decision rule is

$$\pi(M1) = 0$$

$$\pi(M2) = 1$$

$$\pi(M3) = 2$$

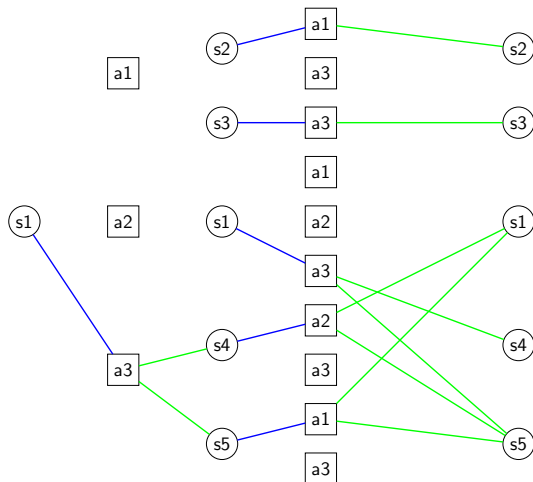
I can compute the value function for this policy (the MDP is now a Markov Reward Process) either formally or by Monte Carlo.

But to find the best policy, I have to compare different reward processes !!!

## Decision tree associated with a decision rule

Take the model of the illustration before. We assume that

$$\pi(s_1) = a_3, \pi(s_2) = a_1, \pi(s_3) = a_3, \pi(s_4) = a_2, \pi(s_5) = a_1.$$



# Main goal of MDP

## Partial Order on value function

We denote by  $\mathcal{V}$  the space of functions from  $\mathcal{X} \rightarrow \mathbb{R}$  (related with the vector space  $\mathbb{R}^{|\mathcal{X}|}$ )

The space  $\mathcal{V}$  has a partial order :

$$\forall U, V \in \mathcal{V} \quad U \leq V \Leftrightarrow U(x) \leq V(x), \forall x \in \mathcal{X}.$$

## Goal of a MDP

The goal of a Markov Decision Process is to characterise, as well as to search and compute (if they exist): the optimal policy  $\Pi^* \in \Pi^{HR}$  ( $\Pi^{HR}$  is the set of history dependent random policies) such that

$$\forall \Pi \in \Pi^{HR}, \forall x \in \mathcal{X} : V^\Pi(x) \leq V^{\Pi^*}(x),$$

(equivalently said  $\Pi^* \in \arg \max_{\Pi \in \Pi^{HR}} V^\Pi$ ) and the optimal value  $V^{\Pi^*}$ .



# Outline

- 4 Resolution of Markov Decision Processes
  - Bellman Equation
  - Resolution Algorithms

## Equivalence of policy

If we should consider all the history dependent random policies, then the computation will suffer from **curse of dimensional**.

Fortunately, there are results that claim we can search in the *Markov Deterministic policy* set.

### Theorem

*Let  $\Pi \in \Pi^{HR}$  be an history dependent random policy. Then for each initial state  $x$  it exists a Markov Random policy  $\Pi'$  such that*

- $V_N^\Pi(x) = V_N^{\Pi'}(x)$  *Finite horizon*
- $V_\theta^\Pi(x) = V_\theta^{\Pi'}(x)$  *Discounted cost*
- $V^\Pi(x) = V^{\Pi'}(x)$  *Total reward*
- $\rho^\Pi(x) = \rho^{\Pi'}(x)$  *average reward*

*For each objective, the two value functions have the same value.*

## Equivalence of policy

If we should consider all the history dependent random policies, then the computation will suffer from **curse of dimensional**.

Fortunately, there are results that claim we can search in the *Markov Deterministic policy* set.

Then the passing of the equivalence from Markov Random policy to Markov Deterministic policy is a case by case study (for each objective).

Proofs and validity conditions can be found in [1].

# Dynamic Programming and framework of proof

## Bellman formulation of DP principle

If the shortest path from town A to town B goes through town C then the shortest path from C to B is the portion between C and B of the shortest path between A and B.

# Dynamic Programming and framework of proof

## Bellman formulation of DP principle

If the shortest path from town A to town B goes through town C then the shortest path from C to B is the portion between C and B of the shortest path between A and B.

Then to obtain the shortest path from A to B:

- ① you consider all the successors of A
- ② for each successor (say  $y$ )
  - you compute the length of the path from A to B by adding:
    - the length from A to  $y$
    - the shortest path from  $y$  to B.
- ③ you take the successor such that the path from A to B is the smallest.

# Dynamic Programming and framework of proof

## Bellman formulation of DP principle

If the shortest path from town A to town B goes through town C then the shortest path from C to B is the portion between C and B of the shortest path between A and B.

## Framework of proof

- 1 Express the *Bellman Equation*.  
This means decompose your cost in two parts:  
the immediate cost due to action and the future cost due to the effects of your action.
- 2 Prove its validity:  
This means to show that using the Bellman Equation will lead to the optimal policy.
- 3 Solve the equation numerically:  
This means apply well known algorithms.

# Bellman Equation

Let us compute the Bellman equation for the finite horizon criterion.

$$\begin{aligned}
 V_N^*(x) &= \max_{a, \dots, a \in \mathcal{A}^N} \mathbb{E} \left( \sum_{k=0}^{N-1} r(x_k, a_k) \mid x_0 = x \right), \\
 &= \max_{a, \dots, a \in \mathcal{A}^N} \left( \mathbb{E} \left( r(x, a) \mid x_0 = x \right) + \mathbb{E} \left( \sum_{k=1}^{N-1} r(x_k, a_k) \mid x_0 = x \right) \right), \\
 &= \max_{a, \dots, a \in \mathcal{A}^N} \left( r(x, a) \right. \\
 &\quad \left. + \mathbb{E} \left( \sum_{y \in \mathcal{X}} \sum_{k=1}^{N-1} r(x_k, a_k) \mathbb{1}_{\{x_1=y, a_0=a\}} \mid x_0 = x \right) \right),
 \end{aligned}$$

# Bellman Equation

Let us compute the Bellman equation for the finite horizon criterion.

$$\begin{aligned} V_N^*(x) &= \max_{a, \dots, a \in \mathcal{A}^N} \left( r(x, a) \right. \\ &\quad \left. + \sum_{y \in \mathcal{X}} \mathbb{E} \left( \sum_{k=1}^{N-1} r(x_k, a_k) \middle| x_1 = y, a_0 = 0, x_0 = x \right) \times \right. \\ &\quad \left. \mathbb{P}(x_1 = y | x_0 = x, a_0 = a) \right) \\ &= \max_{a, \dots, a \in \mathcal{A}^N} \left( r(x, a) + \sum_{y \in \mathcal{X}} p(y|x, a) \mathbb{E} \left( \sum_{k=1}^{N-1} r(x_k, a_k) \middle| x_1 = y \right) \right) \end{aligned}$$



# Bellman Equation

Let us compute the Bellman equation for the finite horizon criterion.

$$\begin{aligned} V_N^*(x) &= \max_{a \in \mathcal{A}} \left( r(x, a) + \sum_{y \in \mathcal{X}} p(y|x, a) \times \right. \\ &\quad \left. \max_{a_1, \dots, a_{N-1} \in \mathcal{A}^{N-1}} \left( \mathbb{E} \left( \sum_{k=1}^{N-1} r(x_k, a_k) \mid x_1 = y \right) \right) \right) \\ &= \max_{a \in \mathcal{A}} \left( r(x, a) + \sum_{y \in \mathcal{X}} p(y|x, a) V_{N-1}^*(y) \right). \end{aligned}$$

# Bellman Equation (ctd.)

The Bellman equation in finite horizon case is :

$$V_N^*(x) = \max_{a \in \mathcal{A}} \left( r(x, a) + \sum_{y \in \mathcal{X}} V_{N-1}^*(y) p(y|x, a) \right). \quad (2)$$

Sketch of proof:

To prove its validity we have several things to show.

- ① Show that the value function  $V_N$  that results from Eq. (2) is the optimal (i.e.  $V_N^*$ ).
- ② Show that the value function  $V_{N-1}$  used is the optimal (i.e.  $V_{N-1}^*$ )

# Bellman Equation (proof of Bellman function)

## Proof of Bellman Function in other fields

The checking of the validity of the Bellman Equation is not restricted to MDP !

e.g. a graph course that presents the Bellman-Ford algorithm will check the conditions of convergence.

## Many works have already done the job

- In a large number of MDP models the validity are proved and the assumptions that should be verified are detailed.
- The Puterman's book presents some of them with detailed proofs.
- When you work with MDP you have to use the assumptions related to your model.  
e.g. bounded rewards or unbounded rewards models require different sets of assumptions

# Resolution algorithm

It exists three main ways to compute both optimal value function and optimal policy (both are computed together).

- 1 Value Iteration (VI)
- 2 Policy Iteration (PI)
- 3 Linear Programming

Note that the computation of the decision rule is computed off-line and then applied by the controller.

You can compute the decision rule *on-line* but you are in the field of *on-line MDP*.

# Value Iteration for the finite horizon

Backward Algorithm: resolution of finite horizon criterion

Initialise  $V_0 = r_N$

**for**  $k = N - 1$  to 0 do

**for**  $x \in \mathcal{X}$  do

$$V_k^*(s) = \max_{a \in \mathcal{A}} \left[ r_k(x, a) + \sum_{y \in \mathcal{X}} \mathbb{P}_k(y|x, a) V_{k+1}^*(y) \right]$$

$$\pi_k(x) \in \arg \max_{a \in \mathcal{A}} \left[ r_k(x, a) + \sum_{y \in \mathcal{X}} \mathbb{P}_k(y|x, a) V_{k+1}^*(y) \right]$$

**end for**

**end for**

**return** the two sequences  $V_k^*$  and  $\pi_k$

# Value Iteration for discounted infinite horizon

value iteration algorithm:  $\theta$ -discounted cost criterion

Initialise  $V_0$

$k \leftarrow 0$

**repeat**

**for**  $x \in \mathcal{X}$  **do**

$$V_{k+1}(x) = \max_{a \in \mathcal{A}} \left( r(x, a) + \theta \sum_{y \in \mathcal{S}} \mathbb{P}(y|x, a) V_k(y) \right)$$

$k \leftarrow k + 1$

**until**  $\|V_{k+1} - V_k\| \leq \epsilon$

**for**  $x \in \mathcal{X}$  **do**

$$\pi(x) \in \arg \max_{a \in \mathcal{A}} \left\{ r(x, a) + \theta \sum_{y \in \mathcal{X}} p(y|x, a) V_{k+1}(y) \right\}$$

**return**  $V_{k+1}, \pi$

# Policy Iteration for infinite horizon discounted cost

Policy iteration algorithm:  $\theta$ -discounted cost criterion

Initialise  $\pi_0 \in \Pi^{MD}$

$k \leftarrow 0$

**repeat**

  Computation phase): solve

$$V_k(x) = r(x, \pi_k(x)) + \theta \sum_{y \in \mathcal{X}} p(y|(x, \pi_k)) V_k(y), \quad \forall x \in \mathcal{X}$$

  improvement phase): improve

**for**  $x \in \mathcal{X}$  **do**

$$\pi_{k+1}(x) \in \arg \max_{a \in \mathcal{A}} \{r(x, a) + \theta \sum_{y \in \mathcal{X}} p(y|(s, a)) V_k(y)\}$$

$k \leftarrow k + 1$

**until**  $\pi_k = \pi_{k+1}$

**return**  $V_k, \pi_{k+1}$

# Outline

- 5 Related models (extension of MDP)
  - Approximated MDP
  - Reinforcement Learning Models
  - Stochastic Games



# Approximated MDP

Consider *again* the first example :

with some modifications

- you observe a large number of positions on each of the mountains,
- the cardinal of your state space increases exponentially,

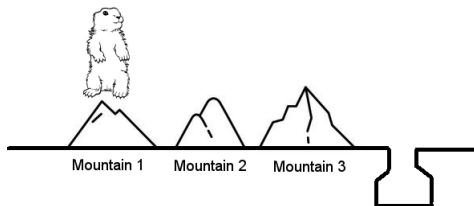
thus the MDP can not be stored in memory.

You can not solve completely the MDP and you should approximate it.

You use *approximate dynamic programming* or *factored MDP* but only approximated values or partial exploration are obtained.

# Reinforcement Learning

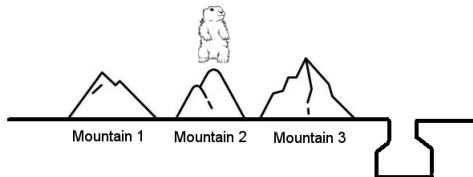
Consider *again* the first example :



Assumption: You can put food on one mountain each day.

# Reinforcement Learning

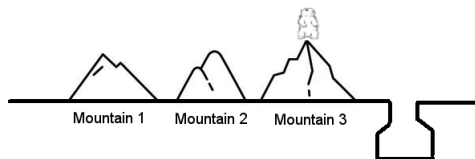
Consider *again* the first example :



Assumption: You know neither transition probabilities nor rewards.

# Reinforcement Learning

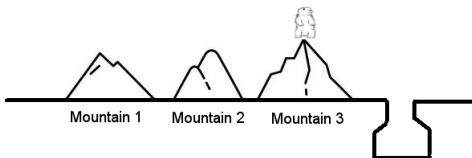
Consider *again* the first example :



You are in a *Reinforcement learning* case

# Reinforcement Learning

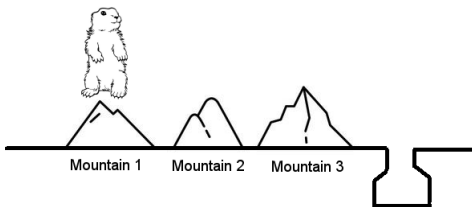
Consider *again* the first example :



You will learn probabilities and rewards (and “value function”) to determine the best action.

# Stochastic games

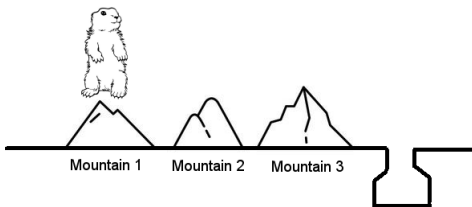
Consider *again* the first example :



You have a neighbour who can also put food.

# Stochastic games

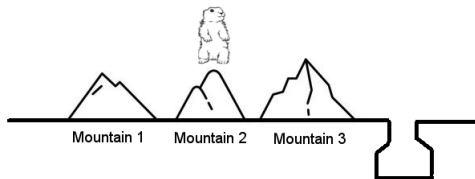
Consider *again* the first example :



But he has not the same rewards than you.

# Stochastic games

Consider *again* the first example :



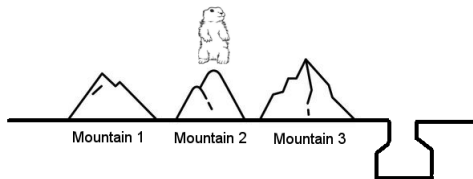
Since there are two people who act, then probabilities and rewards depend on two actions:

$$p(y|x, a_1, a_2) \text{ and } r_1(x, a_1, a_2) \text{ and } r_2(x, a_1, a_2).$$



# Stochastic games

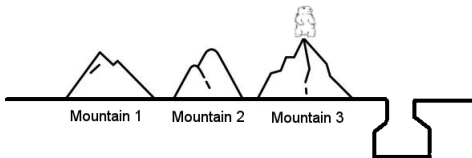
Consider *again* the first example :



The model is now a *stochastic game*.

# Stochastic games

Consider *again* the first example :



Same basis (due to Shapley) but computation of maxima is replaced by Nash Equilibrium equation.

# Outline

## 6 Bibliography

# Bibliography

References that present the main concepts of MDP



F. Garcia.

*Processus Décisionnels de Markov*

Chapter 1 in *Processus Décisionnels de Markov en Intelligence Artificielle*

*Hermes, 2008.*



W.B. Powell

Introduction to Markov Decision Processes

Chapter 3 in *Approximate Dynamic Programming, Solving the Curses of Dimensionality*

*Wiley, 2011.*

# Bibliography

Complete book (with detailed proofs)



[M. Puterman.](#)

*Markov Decision Processes Discrete Stochastic Dynamic Programming.*

[Wiley, 2005.](#)

Article to locate MDP in the stochastic optimization field.



[W. B. Powell.](#)

Clearing the Jungle of Stochastic Optimization

[Informs TutORials, 2014.](#)