Intelligence Artificielle
Apprentissage par renforcement

Emmanuel ADAM

Université Polytechnique des Hauts-De-France

UPHF/INSA HdF

Processus de Markov Décisionnels

Markov Decision Process

un MDP est un graphe composé :

Objectif : trouver la **politique**/stratégie $\pi^* = \{a_0, \ldots, a_n\}$ qui maximise l'obtention des récompenses à partir d'un état initial

Processus de Markov Décisionnels

Markov Decision Process

un MDP est un graphe composé :

- d'états ($\mathbf{S} = \{s_i\}$), d'actions reliant ces états ($\mathbf{A} = \{a_i\}$),

Objectif : trouver la **politique**/stratégie $\pi^* = \{a_0, \ldots, a_n\}$ qui maximise l'obtention des récompenses à partir d'un état initial

Processus de Markov Décisionnels

Markov Decision Process

un MDP est un graphe composé :

- d'états ($\mathbf{S} = \{s_i\}$), d'actions reliant ces états ($\mathbf{A} = \{a_i\}$),
- d'une fonction de transition **P(s, s', a)** indiquant la probabilité d'atteindre un état s' à partir d'un état s en prenant l'action a

Objectif : trouver la **politique**/stratégie $\pi^* = \{a_0, \ldots, a_n\}$ qui maximise l'obtention des récompenses à partir d'un état initial

Processus de Markov Décisionnels

Markov Decision Process

un MDP est un graphe composé :

- d'états ($\mathbf{S} = \{s_i\}$), d'actions reliant ces états ($\mathbf{A} = \{a_i\}$),
- d'une fonction de transition **P(s, s', a)** indiquant la probabilité d'atteindre un état s' à partir d'un état s en prenant l'action a
- d'une **fonction de récompense** associant une récompense ou une pénalité à un état : $r(s_t) \rightarrow \Re$,

Objectif : trouver la **politique**/stratégie $\pi^* = \{a_0, \ldots, a_n\}$ qui maximise l'obtention des récompenses à partir d'un état initial

Apprentissage par renforcement : Utilité

Fonction d'utilité

L'objectif pour le système est de trouver les situations les plus utiles

Remarque, souvent on prendra $u(s) = r(s)$

Apprentissage par renforcement : Utilité

Fonction d'utilité

L'objectif pour le système est de trouver les situations les plus utiles

- On note **u(s)** : $S \rightarrow \Re$ la fonction qui évalue l'utilité d'un état

Remarque, souvent on prendra $u(s) = r(s)$

Apprentissage par renforcement : Utilité

Fonction d'utilité

L'objectif pour le système est de trouver les situations les plus utiles

- On note **u(s)** : $S \rightarrow \Re$ la fonction qui évalue l'utilité d'un état
- Il existe un relation d'ordre $\leq$ réflexive, transitive et totale

Remarque, souvent on prendra $u(s) = r(s)$

Apprentissage par renforcement : Utilité

Fonction d'utilité

L'objectif pour le système est de trouver les situations les plus utiles

- On note **u(s)** : $S \rightarrow \Re$ la fonction qui évalue l'utilité d'un état
- Il existe un relation d'ordre $\leq$ réflexive, transitive et totale
  - **reflexive** : $\forall s, u(s) \leq u(s)$

Remarque, souvent on prendra $u(s) = r(s)$

Apprentissage par renforcement : Utilité

Fonction d'utilité

L'objectif pour le système est de trouver les situations les plus utiles

- On note **u(s)** : $S \rightarrow \Re$ la fonction qui évalue l'utilité d'un état
- Il existe un relation d'ordre $\leq$ réflexive, transitive et totale
  - **reflexive** : $\forall s, u(s) \leq u(s)$
  - **transitive** : si $u(x) \leq u(y)$ et $u(y) \leq u(z)$ alors $u(x) \leq u(z)$

Remarque, souvent on prendra $u(s) = r(s)$

Apprentissage par renforcement : Utilité

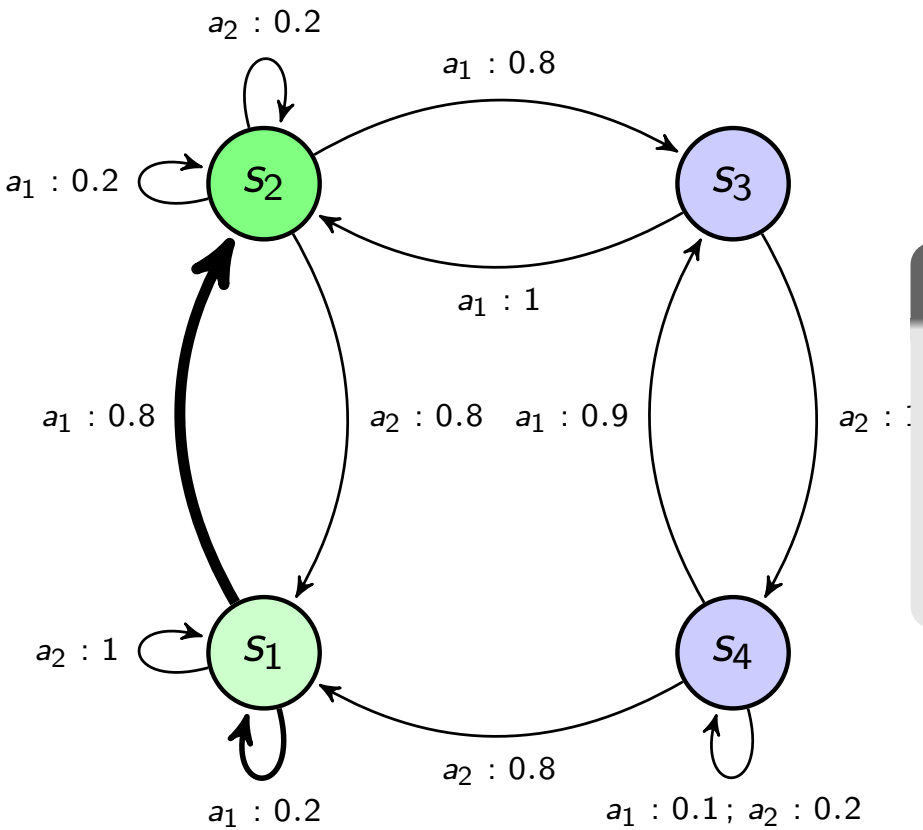
Fonction d'utilité

L'objectif pour le système est de trouver les situations les plus utiles

- On note **u(s)** : $S \rightarrow \Re$ la fonction qui évalue l'utilité d'un état
- Il existe un relation d'ordre $\leq$ réflexive, transitive et totale
  - **reflexive** : $\forall s, u(s) \leq u(s)$
  - **transitive** : si $u(x) \leq u(y)$ et $u(y) \leq u(z)$ alors $u(x) \leq u(z)$
  - **totale** : $\forall s, \forall t, u(s) \leq u(t)$ ou $u(t) \leq u(s)$

Remarque, souvent on prendra $u(s) = r(s)$

MDP : Un exemple simple (inspiré de [Fundamentals of MultiAgent Systems, JM Vidal, 2010])



Exemple de MDP

- à partir de l'état s_1, en effectuant l'action a_1, il y a 80% de chance d'atteindre s_2 et 20% de rester sur s_1

MDP : Un exemple simple (inspiré de [Fundamentals of MultiAgent Systems, JM Vidal, 2010])

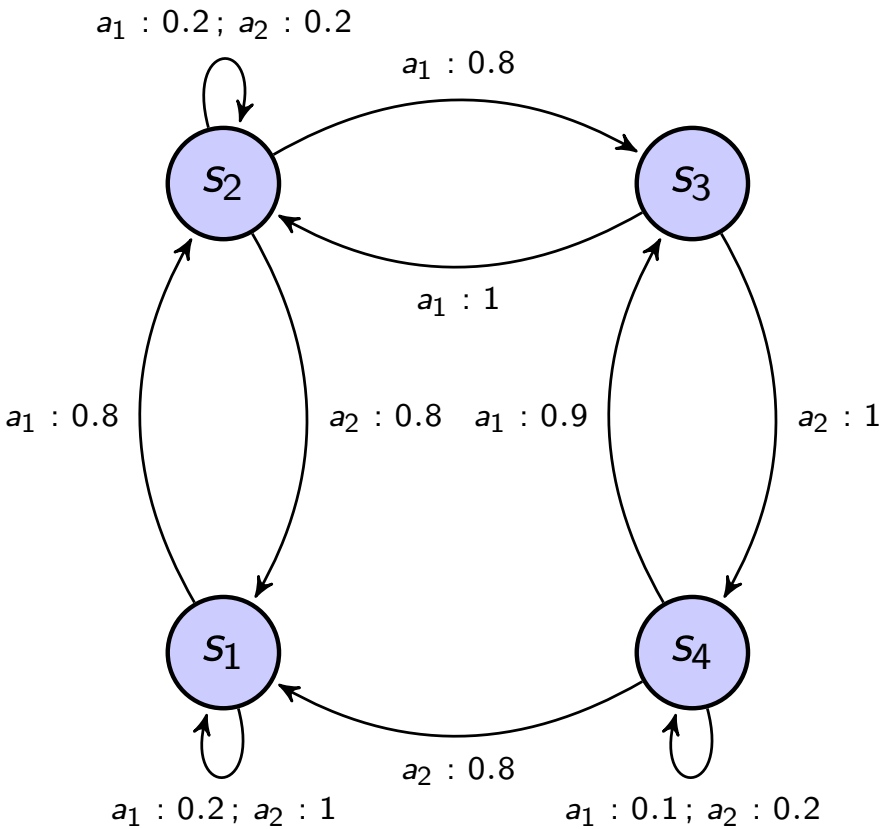
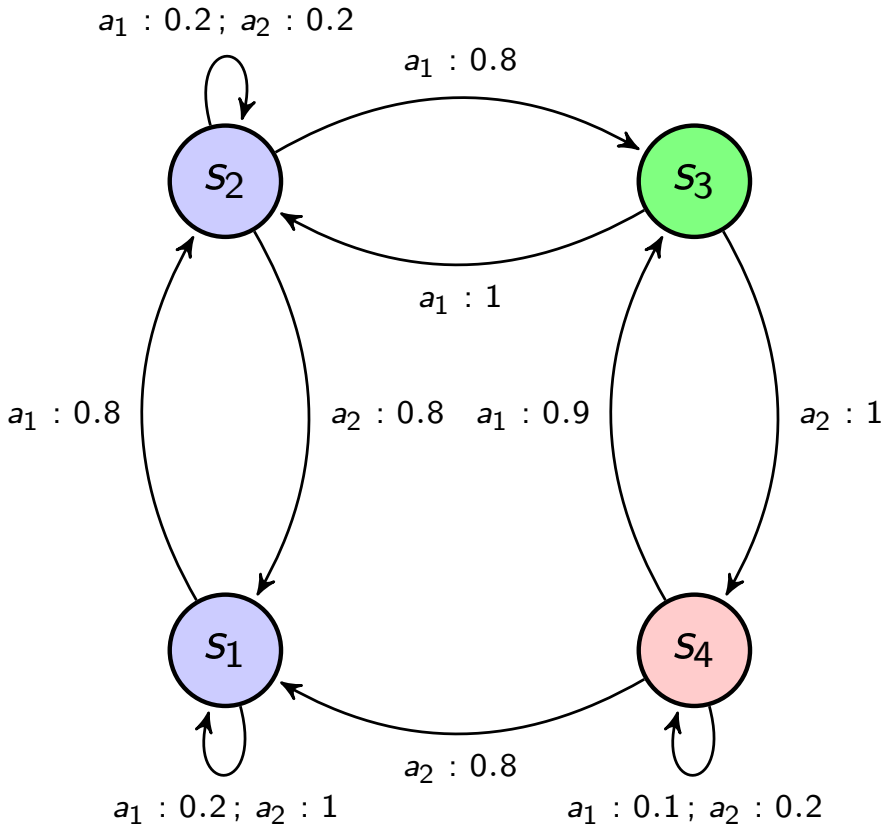


TABLE – $P(s, s', a)$

depart	arrivée	action	proba
s_1	s_1	a_1	0.2
s_1	s_2	a_1	0.8
s_1	s_1	a_2	1
s_2	s_2	a_1	0.2
s_2	s_3	a_1	0.8
s_2	s_2	a_2	0.2
s_2	s_1	a_2	0.8
s_3	s_2	a_1	1
s_3	s_4	a_2	1
s_4	s_4	a_1	0.1
s_4	s_3	a_1	0.9
s_4	s_4	a_2	0.2
s_4	s_1	a_2	0.8

MDP : Un exemple simple (inspiré de [Fundamentals of MultiAgent Systems, JM Vidal, 2010])

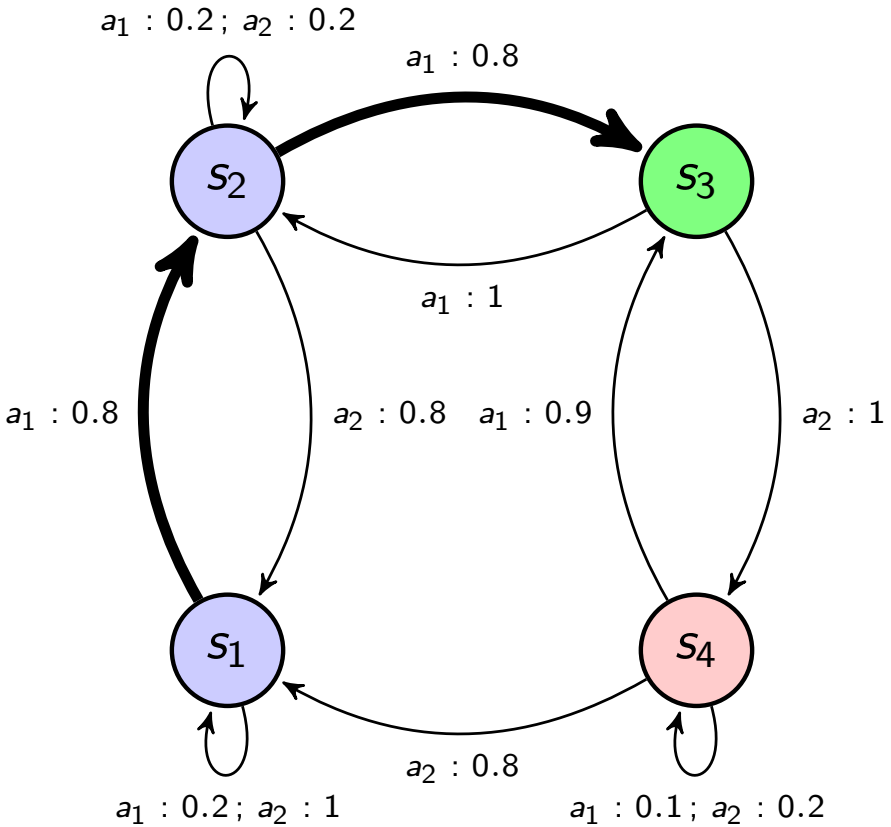


La récompense/pénalité n'est connue que arrivé sur l'état

TABLE – $r(s)$

etat	récompense
s_1	0
s_2	0
s_3	5
s_4	-5

MDP : Un exemple simple (inspiré de [Fundamentals of MultiAgent Systems, JM Vidal, 2010])



Exemple de MDP

- à partir de l'état s_1, comment obtenir le plus de récompense ?
- effectuer 2 fois a_1 est la politique $\pi^* = \{a_1, a_1\}$ qui a le plus de probabilité d'obtenir le gain maximum
- **comment trouver cette politique ?**

Utilité espérée

Utilité espérée d'une action

- A partir d'un état donné s, d'une fonction d'utilité u, calculer l'utilité attendue d'une action a :
- $E(u, s, a) = \sum_{s' \in S} (P(s, s', a) \times u(s'))$

Utilité espérée de a_1 à partir de s_2

Dans l'exemple précédent,

- $E(u, s_2, a_1) = \sum_{s' \in S} (P(s_2, s', a1) \times u(s'))$
- $E(u, s_2, a_1) = P(s_2, s_2, a_1) \times u(s_2) + P(s_2, s_3, a_1) \times u(s_3)$
- $E(u, s_2, a_1) = 0.2 \times 0 + 0.8 \times 5 = 4$
- l'utilité attendue de l'action a_1 à partir de l'état s_2 est de 4

Adapter l'utilitée espérée d'un état

Utilité espérée d'un état

- Un état s est plus utile s'il permet d'effectuer une action utile.
- On ajoute à l'état une partie de la plus grande utilité espérée à partir d'une action :
- $u(s) = r(s) + \gamma . \max_a(\sum_{s'}(P(s, s', a) \times u(s')))$

  $\gamma \in [0, 1]$ est un coefficient de réduction

Utilité espérée de s_2

Dans l'exemple précédent,

- $E(u, s_2, a_1) = 4$, $E(u, s_2, a_2) = 0$, la meilleure action est donc a_1
- si on pose $\gamma = 0.3$, $u(s_2) = r(s_2) + 0.3 \times 4 = 0 + 1.2 = 1.2$
- l'utilité de l'état s_2 n'est plus nulle et vaut 1.2
- par ricochet, l'action a_1 de l'état s_1 devient plus utile que les autres car elle permet d'atteindre l'état s_2 devenu utile...

Apprentissage par renforcement

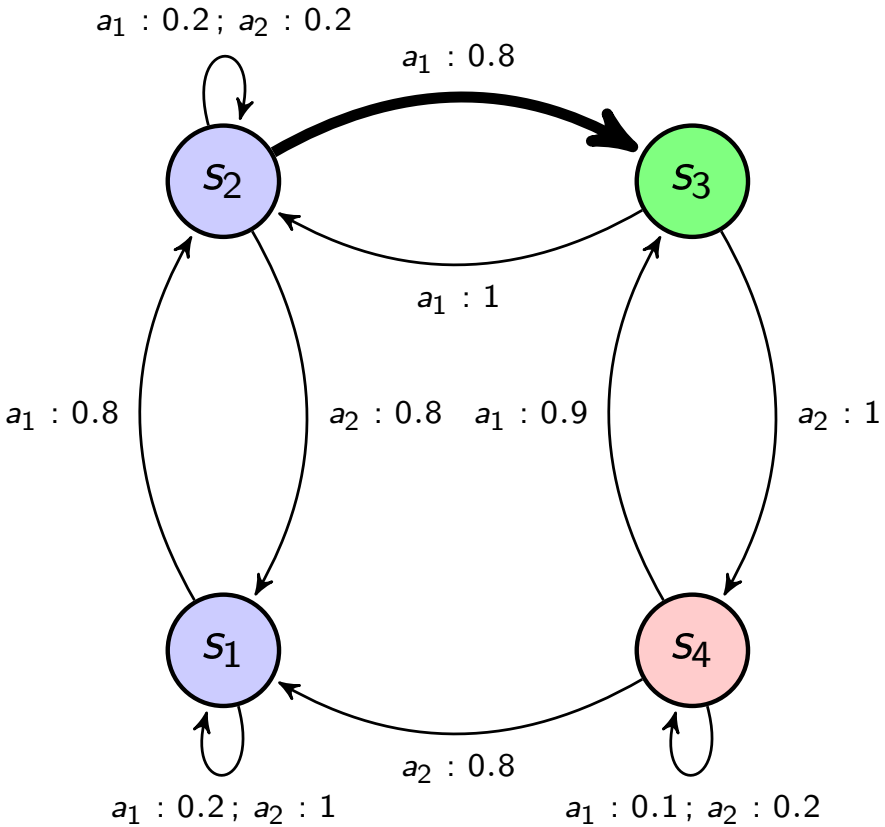
Apprentissage par renforcement : principes (1/3)

- à chaque arrivée sur un état, on note l'action ayant menée à cette état en renforçant la note existante (l'utilité/Qualité) avec une partie de l'utilité du noeud.
- Arrivé sur l'état s' à partir de l'état s et de l'action a, on note la Qualité de l'arc (s, a, s') :

$$Q(s,a) = \lambda \times (r + \gamma \times max_{a'}(Q(s',a')) + (1-\lambda) \times Q(s,a)$$

$\lambda \in [0,1]$ est le coefficient d'apprentissage
$\gamma \in [0,1]$ est le coefficient de réduction
r est la récompense obtenue sur l'état s'

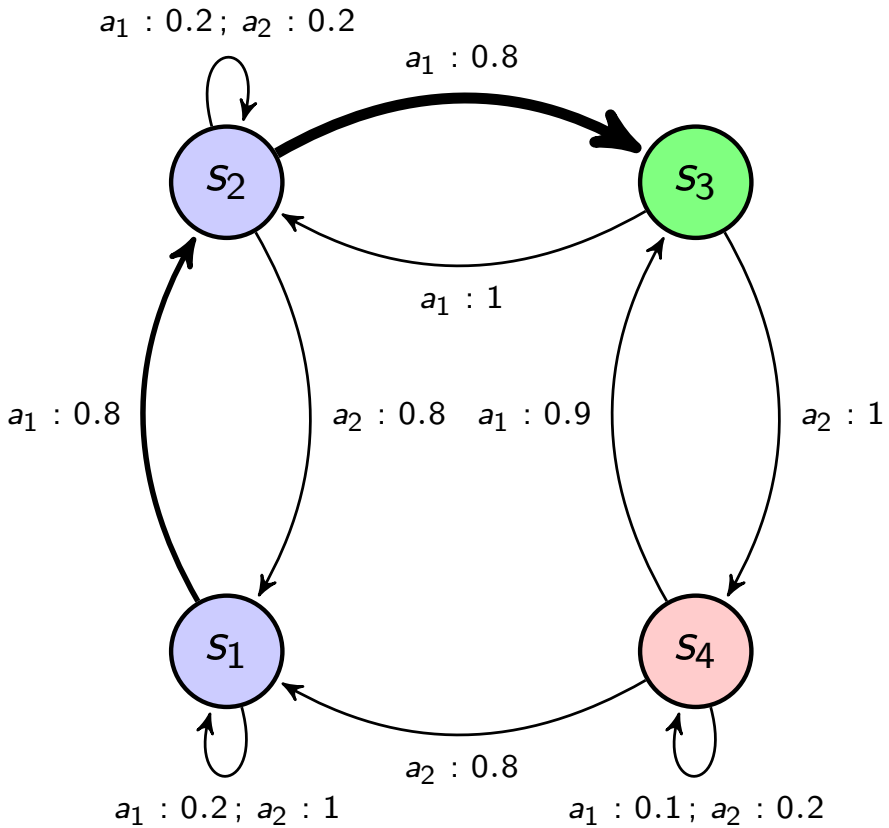
MDP : Un exemple simple (issu de [Fundamentals of MultiAgent Systems, JM Vidal, 2010])



Exemple de QLearning

- arrivé en s_3 à partir de s_2 grâce à a_1, on note : $Q(s_2, a_1) = \lambda \times (r + \gamma \times max_{a'}(Q(s_3, a')) + (1 - \lambda) \times Q(s_2, a_1)$
- on suppose initialement $\forall i, j : Q(s_i, a_j) = 0$
- on suppose $\lambda = 0.4$ et $\gamma = 0.3$: $Q(s_2, a_1) = 0.4 \times (5 + 0.3 \times 0) + (1 - 0.4) \times 0$ $Q(s_2, a_1) = 2$

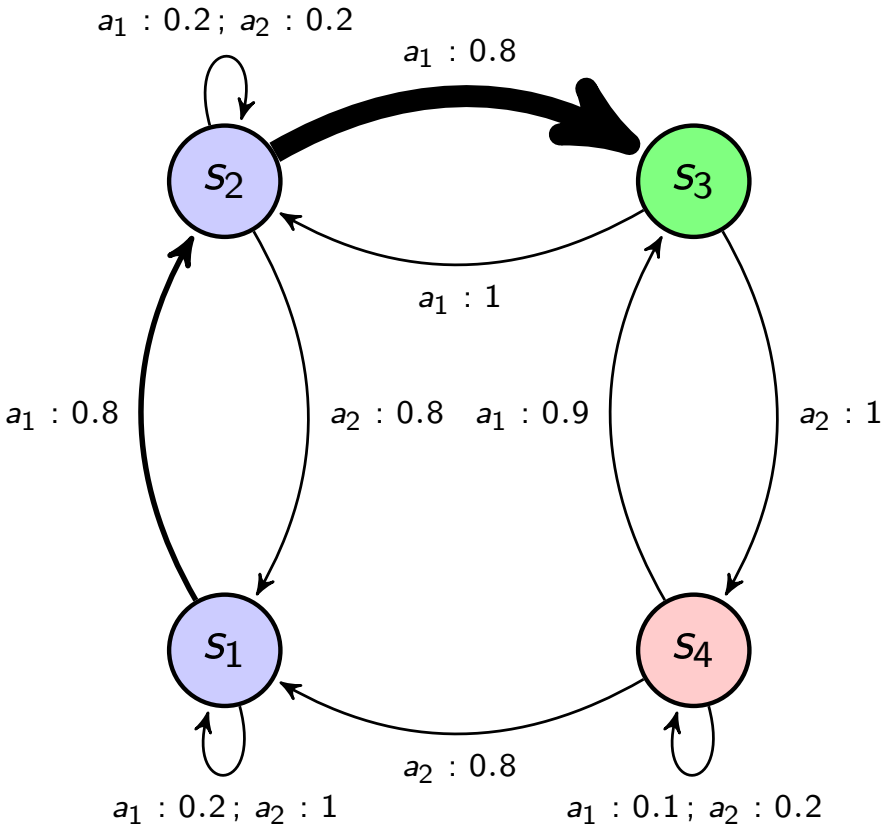
MDP : Un exemple simple (issu de [Fundamentals of MultiAgent Systems, JM Vidal, 2010])



Exemple de QLearning

- à la prochaine recherche de récompense, en partant de s_1 et en arrivant sur s_2 par l'action a_1, on note : $Q(s_1, a_1) =$
  $\lambda \times (r + \gamma \times max_{a'}(Q(s_2, a')) +$
  $(1 - \lambda) \times Q(s_1, a_1)$
  $Q(s_1, a_1) =$
  $0.4 \times (0 + 0.3 \times 2) + (1 - 0.4) \times 0$
  $Q(s_1, a_1) = 0.24$

MDP : Un exemple simple (issu de [Fundamentals of MultiAgent Systems, JM Vidal, 2010])



Exemple de QLearning

- en repassant une nouvelle fois en s_3 à partir de s_2 par l'action a_1, on note : $Q(s_2, a_1) = \lambda \times (r + \gamma \times max_{a'}(Q(s_3, a')) + (1 - \lambda) \times Q(s_2, a_1)$ $Q(s_2, a_1) = 0.4 \times (5 + 0.3 \times 0) + (1 - 0.4) \times 2$ $Q(s_2, a_1) = 2 + 0.6 \times 2 = 3.2$

Q-Learning : principe (2/3)

Apprentissage par renforcement : principe (2/3)

- L'idée est donc d'effectuer plusieurs cycles de recherche de récompenses, de l'état initial vers un état but
- et de renforcer à chaque passage l'utilité/la qualité de l'action qui mène à des récompenses
- ou qui mène à des états menant à des récompenses

Q-Learning : principe (3/3)

Apprentissage par renforcement : exploration

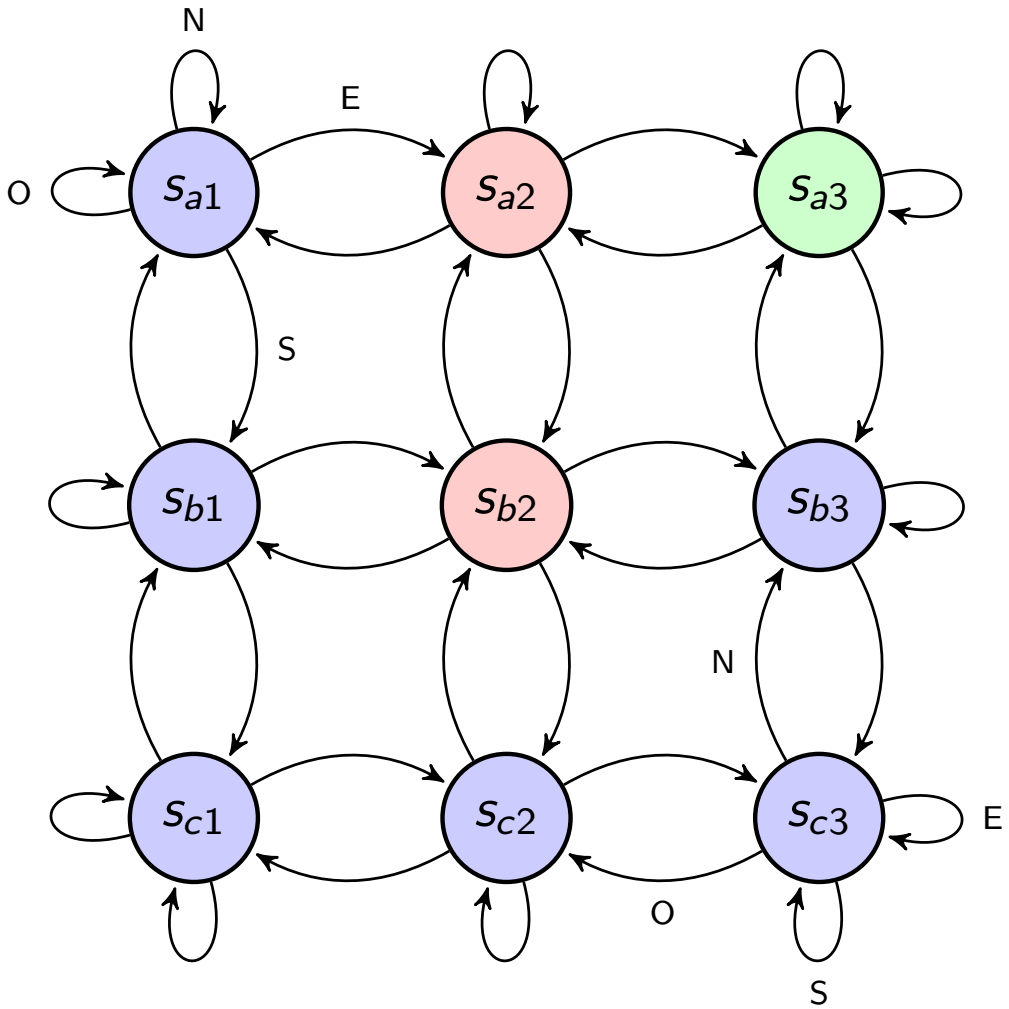
- Initialement, le système
  - ne connaît pas les états où se trouve les récompenses,
  - ne connaît pas a priori l'état d'arrivée d'une action,
- il commence donc par choisir des actions aléatoirement, il **explore**
- au bout d'un certain temps ou lorsqu'il a atteint un état but, le système reprend une recherche de solution à partir de l'état initial
- A chaque cycle, le système présente un comportement **de moins en moins exploratoire**, et de plus en plus guidé par les qualités

Q-Learning : algorithme

Q-Learning : algorithme

procedure QLEARNING
  $\forall s \forall a Q(s,a) \leftarrow 0$
  for $n \leftarrow 1, nbCycles$ **do** ▷ nbCycles d'apprentissage
    $\lambda \leftarrow 1$; $\epsilon \leftarrow 1$;
    $etatCourant \leftarrow etatInitial$
    for $i \leftarrow 1, nbMaxActions$ **do** ▷ max supposé d'actions à tester
      $s \leftarrow etatCourant$
      $nb \leftarrow random(0,1)$
      if $(nb < \epsilon)$ **then**
        $a \leftarrow randomAction(s)$ ▷ choix aléatoire d'une action à partir de s
      else
        $a \leftarrow argMax_{a'}(Q(s,a'))$ ▷ choix de l'action de s avec Q maximum
      end if
      $s' \leftarrow a(s)$ ▷ s' est l'état d'arrivé après exécution de a en s
      $Q(s,a) \leftarrow \lambda \times (r + \gamma \times max_{a'}(Q(s',a'))) + (1-\lambda) \times Q(s,a)$
      $\lambda \leftarrow 0.99 \times \lambda$ ▷ décrémenter les coefficients
      $\epsilon \leftarrow 0.99 \times \epsilon$
      if $(s' = etatFinal)$ **then**
        Sortie boucle i
      end if
    end for
  end for

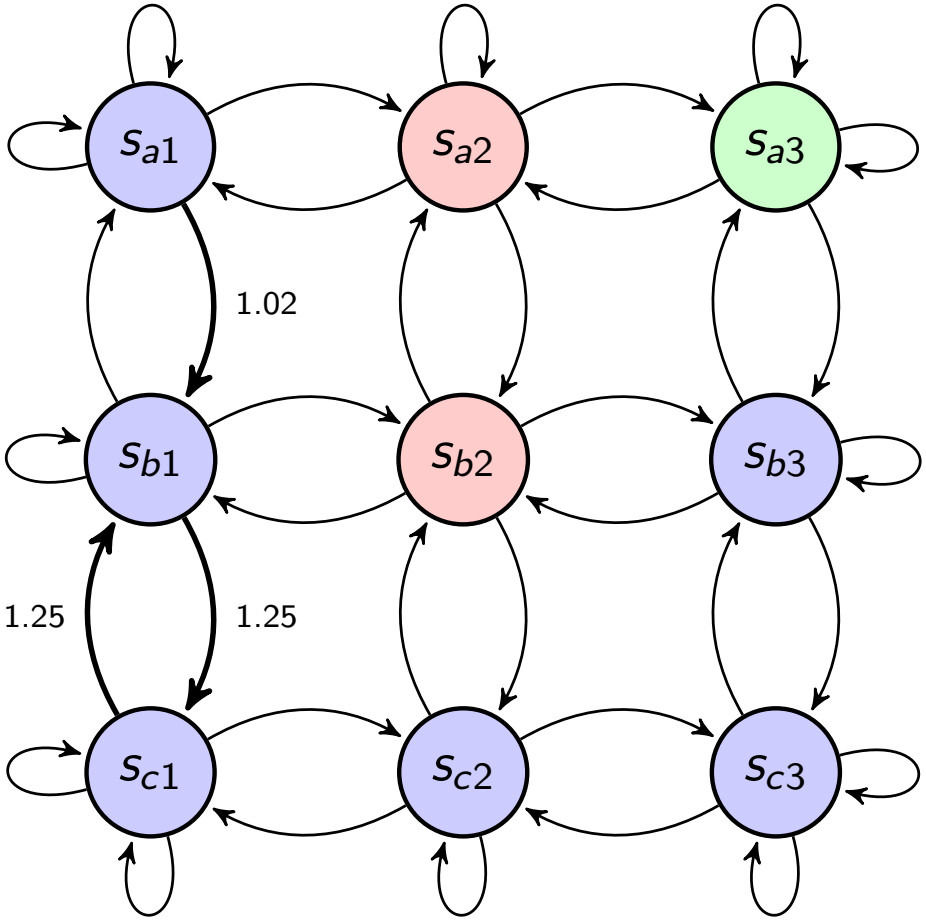
QLearning : exemple de parcours



Exemple de MDP

- récompense en $sa3 = 10$
- pénalités en $sa2$ et $sb2$ valent -1
- récompense aux autres états $sij = 1$
- 4 actions à chaque état : N, S, O, E

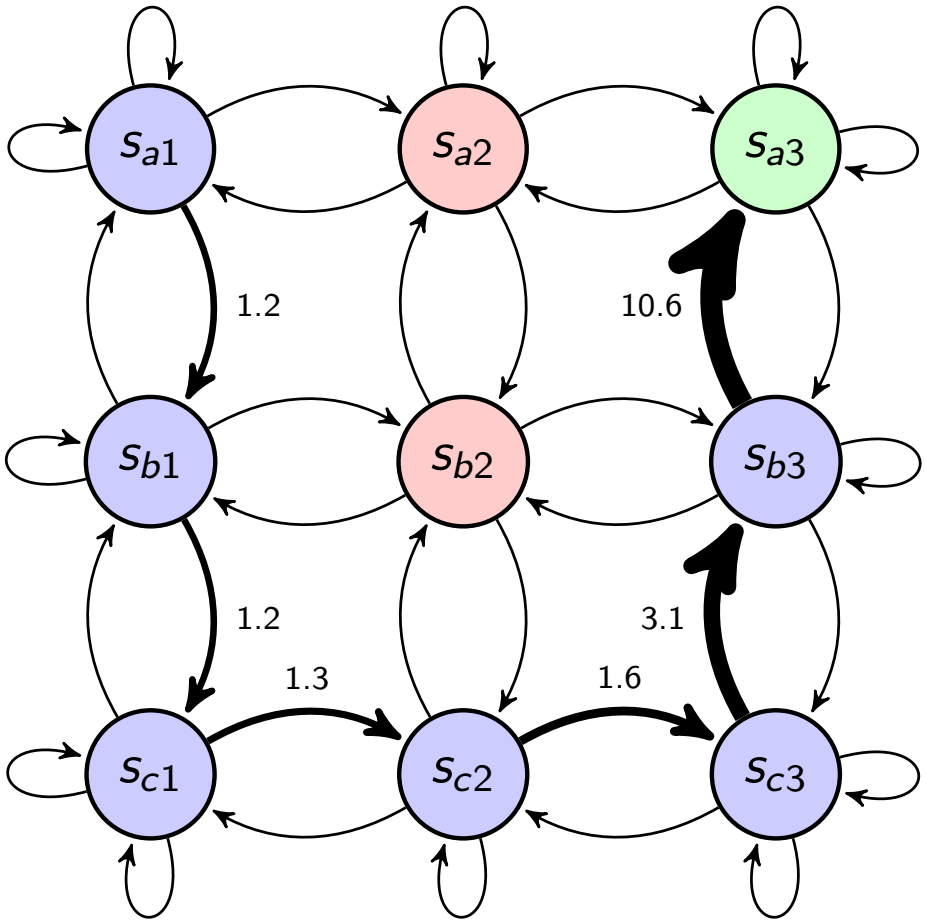
QLearning : exemple de d'apprentissage sur 5 cycles



Exemple de MDP

- $\gamma = 0.2$
- réduction de 1% de λ et ϵ à chaque cycle
- évitement des pénalités
- mais état but non trouvé

QLearning : exemple de d'apprentissage sur 50 cycles



Exemple de MDP

- $\gamma = 0.2$
- réduction de 1% de λ et ϵ à chaque cycle
- évitement des pénalités
- et **état but trouvé**
- (ici affichage uniquement des arcs de plus grande valeur)