Apprentissage par renforcement

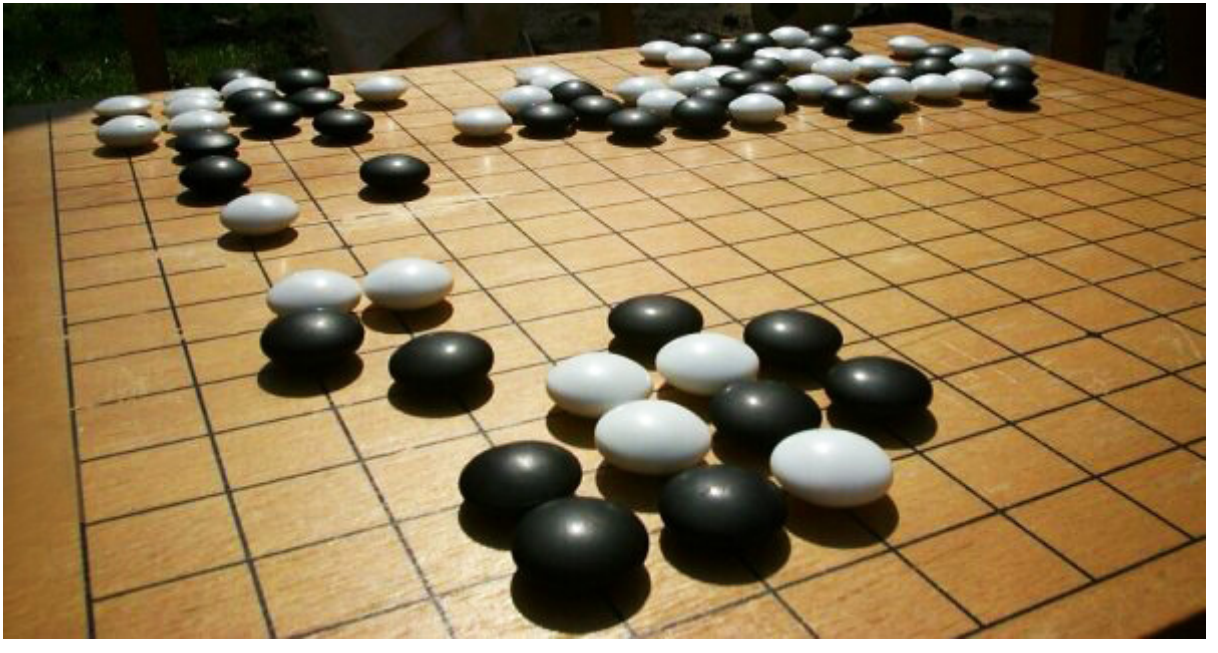
Introduction

Le jeu de Go

- Facteur de branchement très important (∼200)
- Difficile de définir une fonction d'évaluation d'une position

⇒ Approches « *arbre minmax*» ne marche pas

⇒ Peut-on apprendre une bonne stratégie ?



Introduction

Comment apprendre à traverser un labyrinthe sans se tromper ?

Ou encore, apprendre une bonne stratégie pour :

- passer l'aspirateur dans une pièce donnée ?
- piloter les ascenseurs d'une tour ?

Introduction

Apprentissage par renforcement

= Apprentissage en environnement (partiellement) inconnu
= Apprentissage par essais et erreurs
= Apprentissage par interaction avec l'environnement (exploration)

Introduction

Apprentissage par renforcement

= Apprentissage en environnement (partiellement) inconnu
= Apprentissage par essais et erreurs
= Apprentissage par interaction avec l'environnement (exploration)

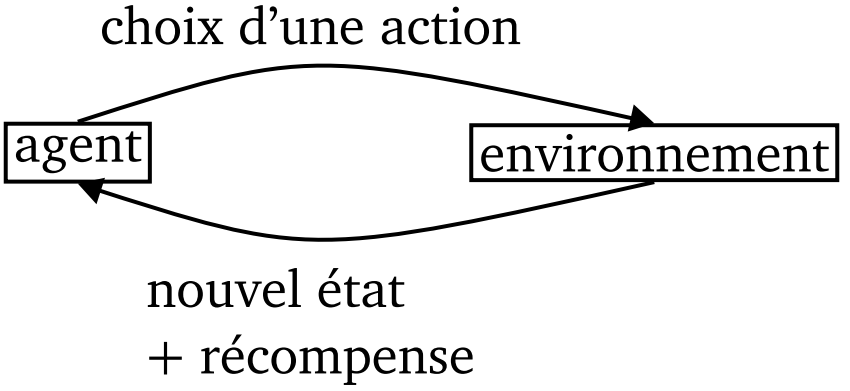
≠ Classification
car on veut ici apprendre de bonnes ***séquences*** d'actions
≠ Planification
car on n'a ici qu'une connaissance ***im***parfaite de l'environnement
car ici l'environnement peut ***changer***
car ici on n'a souvent pas un état initial et un but fixés

Introduction : Un cadre général

Un agent évolue dans un environnement donné.
Il peut effectuer certaines actions dépendant de l'état courant :

- l'état de l'environnement
- et son propre état

ce qui amène dans un nouvel état.
Certaines actions sont liées à des récompenses / coûts immédiats.

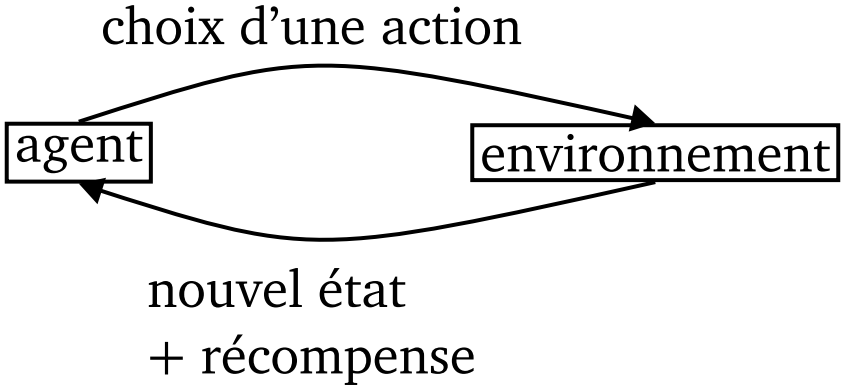


Introduction : Un cadre général

Un agent évolue dans un environnement donné.
Il peut effectuer certaines actions dépendant de l'état courant :

- l'état de l'environnement
- et son propre état

ce qui amène dans un nouvel état.
Certaines actions sont liées à des récompenses / coûts immédiats.



L'agent doit apprendre quelle action choisir dans chaque état afin de suivre une séquence d'action qui lui soit la plus favorable possible.

Introduction : Un cadre général

Apprentissage par renforcement :

permet d'adapter un ***agent*** à un ***environnement***
en tenant compte de ***récompenses***/punitions
(les ***renforcements***)

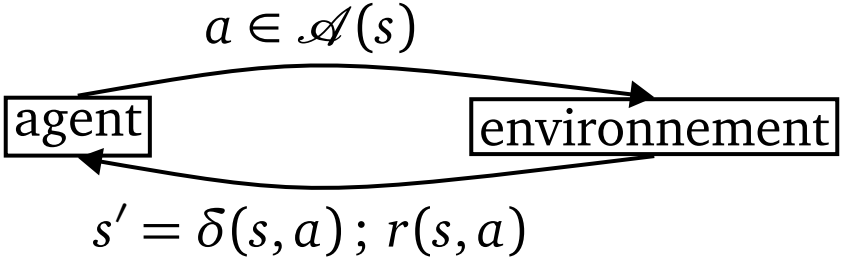
Plan du cours

- Processus de décision markoviens (*MDP*)
- Programmation dynamique
- Apprentissage par renforcement

THE référence : Sutton, R. S. & Barto, A. G. Reinforcement Learning : An Introduction. 2ème édition en préparation, disponible en pdf sur internet. 1ère édition : MIT Press.

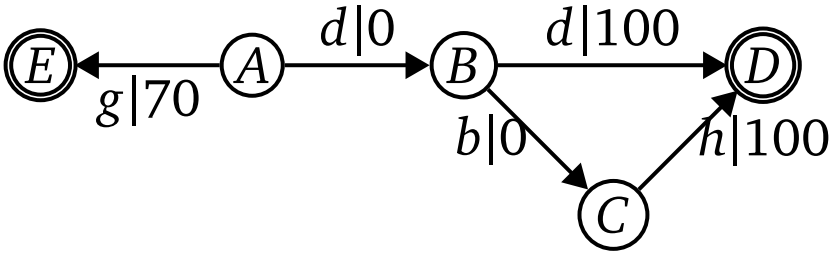
Processus de décision Markovien : États, actions, récompenses

$\mathscr{S}$: un ensemble d'états
$\mathscr{A}$: un ensemble d'actions
$\mathscr{A}(s)$: ensemble des actions réalisables dans l'état s
$r(s,a) \in \mathbf{R}$: récompense immédiate
$\delta(s,a) \in \mathscr{S}$: état résultant



Processus de décision Markovien : États, actions, récompenses

Exemple.



Hypothèses « *markovienne*»:

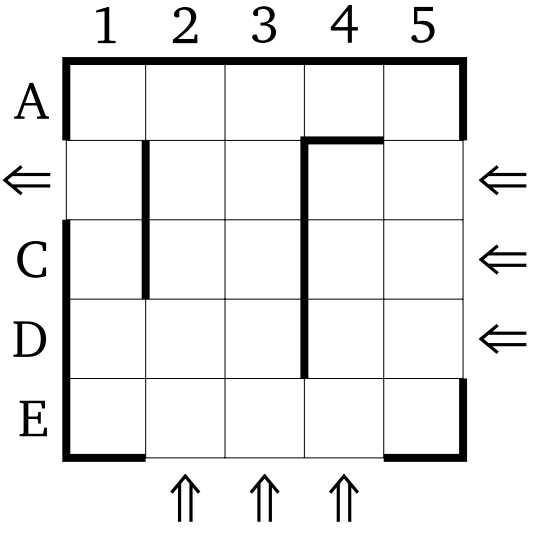
le futur ne dépend pas du passé, sachant le présent

Remarque : on se place pour le moment dans le cas déterministe

Processus de décision Markovien : États, actions, récompenses

Exemple. Un labyrinthe :

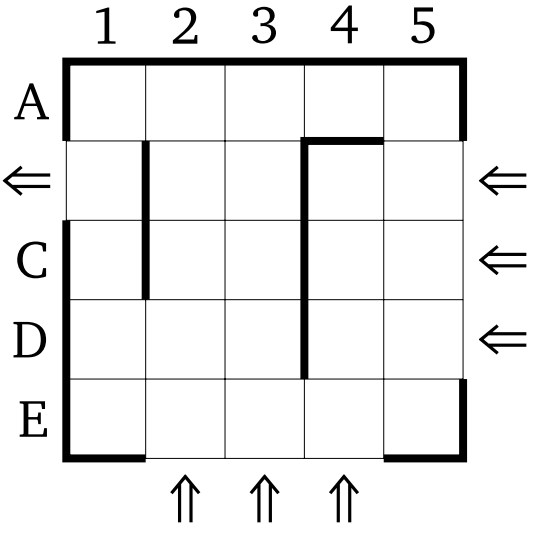
- Plusieurs entrées possibles, 1 sortie
- Déplacements possibles : ↑ ↓ ← →
  (sauf à travers les murs)
- Chaque déplacement coûte 5€
- Gain à la sortie : 100€



Processus de décision Markovien : États, actions, récompenses

Exemple. Un labyrinthe :

- Plusieurs entrées possibles, 1 sortie
- Déplacements possibles : $\uparrow \downarrow \leftarrow \rightarrow$ (sauf à travers les murs)
- Chaque déplacement coûte 5€
- Gain à la sortie : 100€

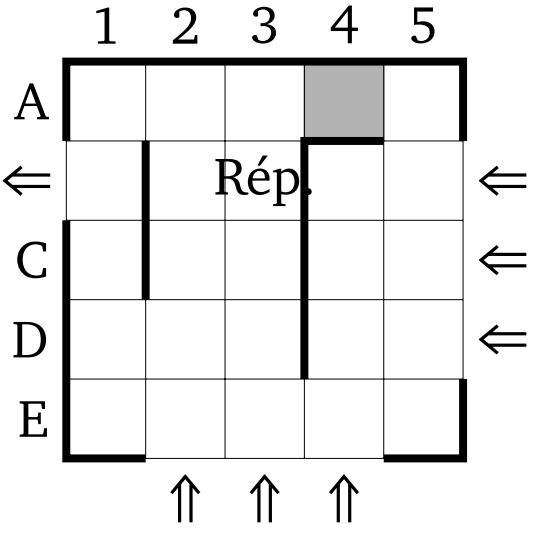


- Combien y a-t-il d'états ? $|\mathscr{S}| =$
- Quelles sont les actions ? $\mathscr{A} =$
- À quelles actions sont associés des coûts / récompenses ?

Processus de décision Markovien : États, actions, récompenses

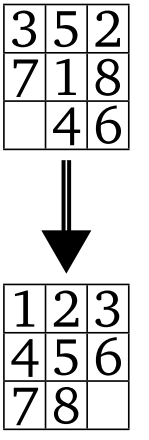
Exemple. Un labyrinthe :

- Plusieurs entrées possibles, 1 sortie
- Déplacements possibles : $\uparrow \downarrow \leftarrow \rightarrow$ (sauf à travers les murs)
- Chaque déplacement coûte 5€
- Gain à la sortie : 100€ / ***50€ si blessé***
- ***Réparation possible en (C,2)***



- Combien y a-t-il d'états ? $|\mathscr{S}| =$
- Quelles sont les actions ? $\mathscr{A} =$
- À quelles actions sont associés des coûts / récompenses ?

Processus de décision Markovien : États, actions, récompenses



Exemple. Le taquin :

- une grille 3×3,
  8 tuiles numérotées de 1 à 8
- un état initial quelconque,
  il faut ranger les tuiles par numéros croissants
- Combien y a-t-il d'états ? $|\mathscr{S}| =$
- Quelles sont les actions ? $\mathscr{A} =$
- À quelles actions sont associés des coûts / récompenses ?

Processus de décision Markovien : Politique

Une ***politique*** π formalise le choix des actions :

$$\pi : \mathscr{S} \to \mathscr{A} \qquad \text{telle que } \pi(s) \in \mathscr{A}(s)$$

Processus de décision Markovien : Politique

Une ***politique*** π formalise le choix des actions :

$$\pi : \mathscr{S} \to \mathscr{A} \qquad \text{telle que } \pi(s) \in \mathscr{A}(s)$$

- Professeur Tournesol : «*toujours plus à l'Ouest*»

Processus de décision Markovien : Politique

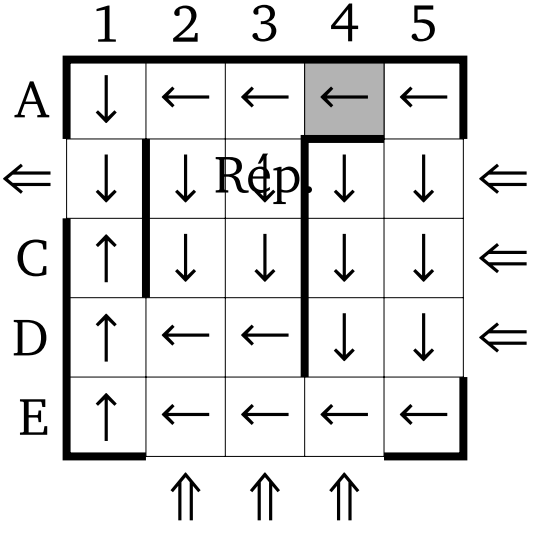
Une ***politique*** π formalise le choix des actions :

$$\pi : \mathscr{S} \to \mathscr{A} \qquad \text{telle que } \pi(s) \in \mathscr{A}(s)$$

- Professeur Tournesol : «*toujours plus à l'Ouest*»
- Le labyrinthe donné en exemple

  Politique 1 :
  - vers le Nord si possible ; sinon
  - vers l'Est si possible ; sinon
  - vers l'Ouest si possible ; sinon
  - vers le Sud

Politique 2 :



Processus de décision Markovien : Politique

Une ***politique*** π formalise le choix des actions :

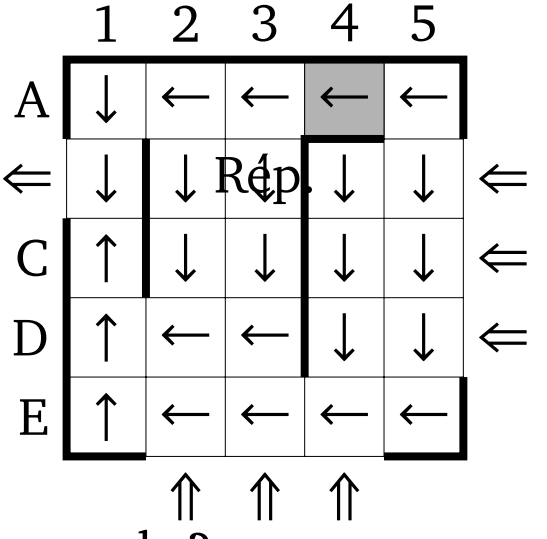
$$\pi : \mathscr{S} \to \mathscr{A} \qquad \text{telle que } \pi(s) \in \mathscr{A}(s)$$

- Professeur Tournesol : «*toujours plus à l'Ouest*»
- Le labyrinthe donné en exemple

Politique 1 :

  - vers le Nord si possible ; sinon
  - vers l'Est si possible ; sinon
  - vers l'Ouest si possible ; sinon
  - vers le Sud

Politique 2 :



Combien de politiques possibles sur cet exemple ?

Processus de décision Markovien : Politique

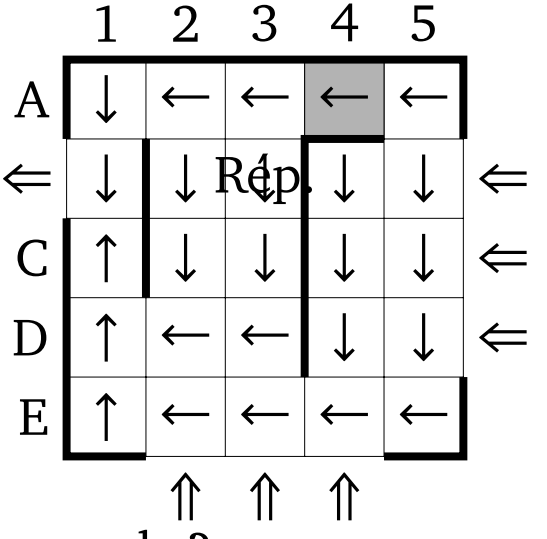
Une ***politique*** π formalise le choix des actions :

$$\pi : \mathscr{S} \to \mathscr{A} \qquad \text{telle que } \pi(s) \in \mathscr{A}(s)$$

- Professeur Tournesol : «*toujours plus à l'Ouest*»
- Le labyrinthe donné en exemple

  Politique 1 :
  - vers le Nord si possible ; sinon
  - vers l'Est si possible ; sinon
  - vers l'Ouest si possible ; sinon
  - vers le Sud

Politique 2 :



  Combien de politiques possibles sur cet exemple ?
- Le taquin 3×3 : combien de politiques possibles ?

Processus de décision Markovien : Politique

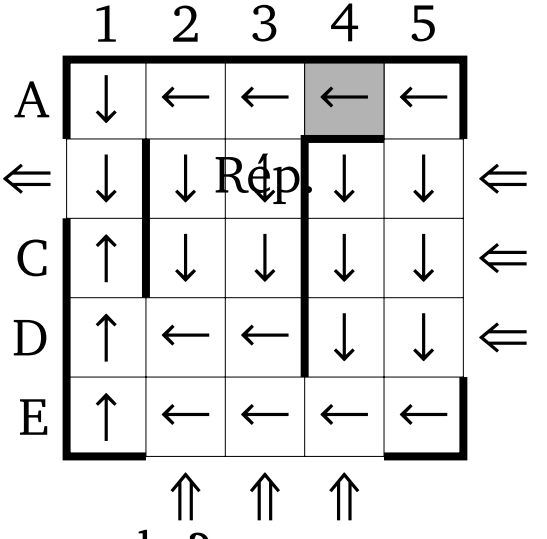
Une ***politique*** π formalise le choix des actions :

$$\pi : \mathscr{S} \to \mathscr{A} \qquad \text{telle que } \pi(s) \in \mathscr{A}(s)$$

- Professeur Tournesol : «*toujours plus à l'Ouest*»
- Le labyrinthe donné en exemple

  Politique 1 :
  - vers le Nord si possible ; sinon
  - vers l'Est si possible ; sinon
  - vers l'Ouest si possible ; sinon
  - vers le Sud

Politique 2 :



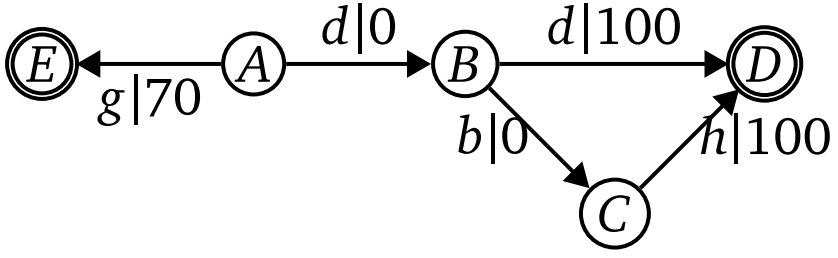
  Combien de politiques possibles sur cet exemple ?
- Le taquin 3×3 : combien de politiques possibles ?

Processus de décision Markovien : Politique

- politique $\neq$ stratégie :
  une politique donne une information très «*locale*» – que jouer dans un état donné
  nous parlerons plus tard de «*stratégie d'apprentissage*»

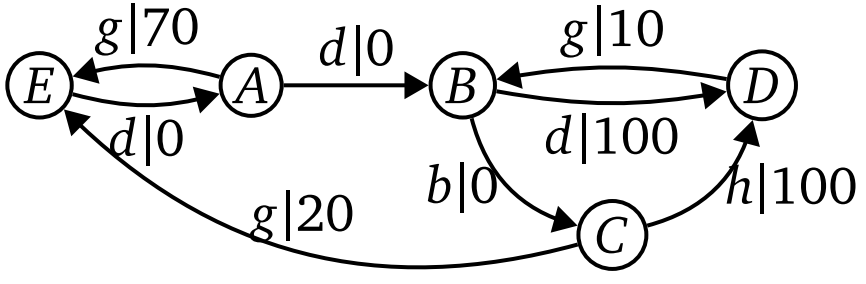
Processus de décision Markovien : Épisodiques ou continus

Tâche épisodique : la séquence d'actions se termine toujours après un nombre fini d'actions – parfois un nombre indéterminé – dans un / des états «*terminaux*»



Exemple : les jeux de sociétés (règles empêchant les parties infinies)

Tâche continue : la séquence d'actions est infinie



Exemple : contrôle de systèmes, tels que ascensceurs...

Processus de décision Markovien : Valeurs d'une politique

Une politique dit quoi faire dans chaque état.
On cherche une politique optimale dans le long terme.
$\Rightarrow$ Récompense à long terme pour une séquence d'états/actions
$\mathbf{s} = (s_0, a_0, s_1, a_1, \ldots)$

Processus de décision Markovien : Valeurs d'une politique

Une politique dit quoi faire dans chaque état.
On cherche une politique optimale dans le long terme.
$\Rightarrow$ Récompense à long terme pour une séquence d'états/actions
$\mathbf{s} = (s_0, a_0, s_1, a_1, \ldots)$

Tâche épisodique : $R(\mathbf{s}) = \sum_{i=0}^{T} r(s_i, a_i)$
où $T =$ longueur de la séquence

Tâche continue : $R(\mathbf{s}) = \sum_{i=0}^{+\infty} \gamma^i r(s_i, a_i)$
où $\gamma =$ constante $\in]0, 1[=$ ***facteur de dépréciation*** (discount rate)

- assure la convergence de la série si récompense bornées
- favorise les récompenses immédiates

Processus de décision Markovien : Valeurs d'une politique

Une politique dit quoi faire dans chaque état.
On cherche une politique optimale dans le long terme.
$\Rightarrow$ Récompense à long terme pour une séquence d'états/actions
$\mathbf{s} = (s_0, a_0, s_1, a_1, \ldots)$

Tâche épisodique : $R(\mathbf{s}) = \sum_{i=0}^{T} r(s_i, a_i)$
où $T =$ longueur de la séquence

Tâche continue : $R(\mathbf{s}) = \sum_{i=0}^{+\infty} \gamma^i r(s_i, a_i)$
où $\gamma =$ constante $\in]0, 1[=$ ***facteur de dépréciation*** (discount rate)

- assure la convergence de la série si récompense bornées
- favorise les récompenses immédiates

Remarque. Une tâche épisodique peut être étudiée comme une tâche continue, si on ajoute soit une boucle aux états terminaux, soit un retour vers un état initial.
$\Rightarrow$ Dans la suite, on considère souvent qu'on a une tâche continue et un $\gamma < 1$.

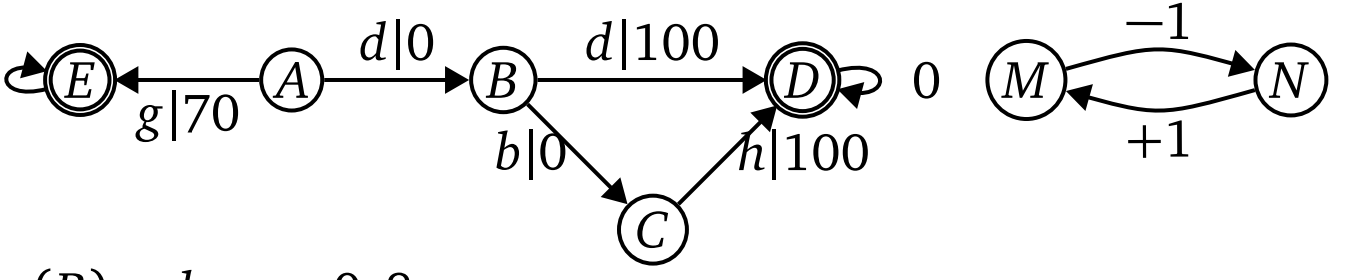
Processus de décision Markovien : Valeurs d'une politique

Gain suivant une politique π à partir d'un état s_0 :

$$R(\pi, s_0) = R(\mathbf{s})$$

où **s** est la séquence d'états à partir de s_0 suivant π.
$(a_i = \pi(s_i), s_{i+1} = \delta(s_i, a_i))$
On l'appelle aussi la ***valeur*** de l'état s_0 pour la politique π.

Exemple.



$\pi_1(A) = d$, $\pi_1(B) = b$, $\gamma = 0,8$

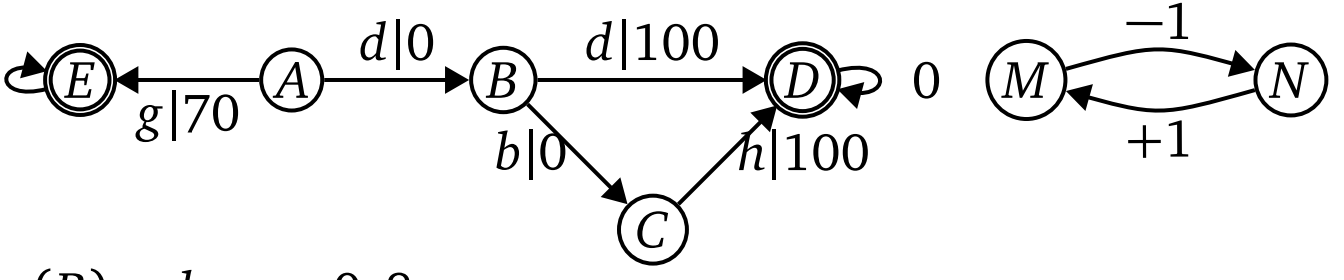
Processus de décision Markovien : Valeurs d'une politique

Gain suivant une politique π à partir d'un état s_0 :

$$R(\pi, s_0) = R(\mathbf{s})$$

où **s** est la séquence d'états à partir de s_0 suivant π.
$(a_i = \pi(s_i), s_{i+1} = \delta(s_i, a_i))$
On l'appelle aussi la ***valeur*** de l'état s_0 pour la politique π.

Exemple.



$\pi_1(A) = d,\ \pi_1(B) = b,\ \gamma = 0,8$

$\pi_2(A) = d,\ \pi_2(B) = d,\ \gamma = 0,8$

Processus de décision Markovien : Politiques optimales

Certaines politiques sont meilleures que d'autres :

$$\pi \succeq \pi' \text{ si } R(\pi, s_0) \geq R(\pi', s_0) \text{ pour tout } s \in \mathscr{S}$$

La relation $\succeq$ est réflexive et transitive.
Relation stricte induite : $\pi \succ \pi'$ si $\pi \succeq \pi'$ et $\pi' \not\succeq \pi$.

Processus de décision Markovien : Politiques optimales

Certaines politiques sont meilleures que d'autres :

$$\pi \succeq \pi' \text{ si } R(\pi, s_0) \geq R(\pi', s_0) \text{ pour tout } s \in \mathscr{S}$$

La relation $\succeq$ est réflexive et transitive.
Relation stricte induite : $\pi \succ \pi'$ si $\pi \succeq \pi'$ et $\pi' \not\succeq \pi$.
Politiques optimales : π telle qu'il n'existe pas de $\pi' \succ \pi$

Propriété. Il existe toujours au moins une politique optimale.

Processus de décision Markovien : Politiques optimales

Certaines politiques sont meilleures que d'autres :

$$\pi \succeq \pi' \text{ si } R(\pi, s_0) \geq R(\pi', s_0) \text{ pour tout } s \in \mathscr{S}$$

La relation $\succeq$ est réflexive et transitive.
Relation stricte induite : $\pi \succ \pi'$ si $\pi \succeq \pi'$ et $\pi' \not\succeq \pi$.
Politiques optimales : π telle qu'il n'existe pas de $\pi' \succ \pi$

Propriété. Il existe toujours au moins une politique optimale.

Questions auxquelles on va répondre dans la suite :

- Existe-t-il une politique optimale ?
- Comment calculer / apprendre une politique optimale ?

Processus de décision Markovien : Équations de Bellman

On commence par voir comment on peut calculer les valeurs d'une politique fixée.

Propriété. Pour toute politique π, pour tout état s :

$$R(\pi, s) = r(s, \pi(s)) + \gamma R(\pi, \delta(s, \pi(s))$$

Processus de décision Markovien : Équations de Bellman

On commence par voir comment on peut calculer les valeurs d'une politique fixée.

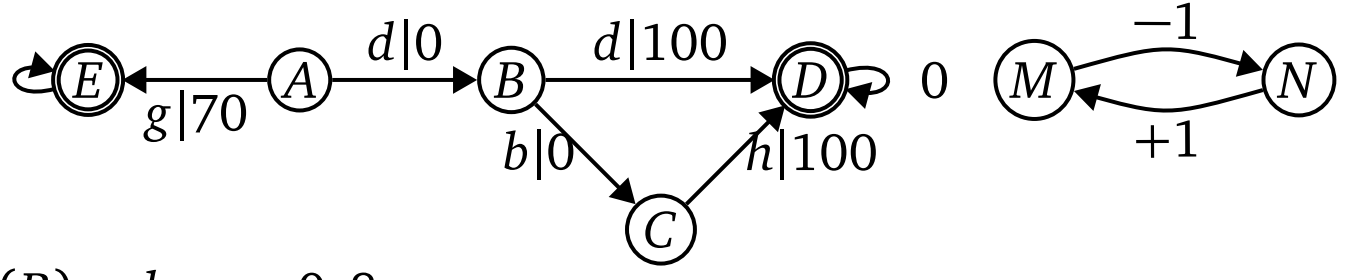
Propriété. Pour toute politique π, pour tout état s :

$$R(\pi, s) = r(s, \pi(s)) + \gamma R(\pi, \delta(s, \pi(s))$$

Cette propriété permet de calculer R pour une politique π donnée, si on a une connaissance parfaite de r et δ.

Résolution d'un système de $|\mathcal{S}|$ équations linéaires
$R(\pi, s) = r(s, \pi(s)) + \gamma R(\pi, \delta(s, \pi(s))$ pour tout $s \in \mathcal{S}$

Exemple.



$\pi(A) = d$, $\pi(B) = b$, $\gamma = 0,8$

Processus de décision Markovien : Équations de Bellman

Pour améliorer la politique courante :

Propriété. Pour une politique π donnée, s'il existe $s \in \mathscr{S}$ et $a \in \mathscr{A}(s)$ tels que

$$R(\pi, s) < r(s, a) + \gamma R(\pi, \delta(s, a))$$

soit π' égale à π sauf en s : on pose $\pi'(s) = a$.
Alors $\pi' \succ \pi$.

Processus de décision Markovien : Équations de Bellman

La «*fameuse*» fonction Q :

$$Q(\pi, s, a) = r(s, a) + \gamma R(\pi, \delta(s, a))$$

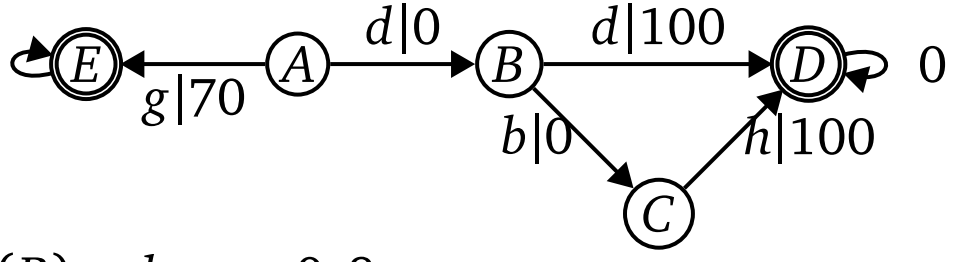
On a alors $R(\pi, s) = Q(\pi, s, \pi(s))$.

Propriété. Pour une politique π donnée, Q vérifie :

$$Q(\pi, s, a) = r(s, a) + \gamma Q(\pi, s', \pi(s') \text{ (où } s' = \delta(s, a) \text{)}$$

La fonction Q permet de détailler ce qui se passe si on dévie de la politique courante pour 1 transition seulement.

Exemple.



$\pi(A) = d$, $\pi(B) = b$, $\gamma = 0,8$

Processus de décision Markovien : Équations de Bellman

Système d'équations vérifié par les politiques optimales :
si π^* est optimale, alors pour tous $s \in \mathscr{S}$, $a \in \mathscr{A}(s)$

$$Q(\pi^*, s, a) = r(s, a) + \max_{a' \in \mathscr{A}(s')} Q(\pi^*, s', a') \text{ (où } s' = \delta(s, a) \text{)}$$

Mais ça n'est pas un système d'équations linéaires ordinaire (argmax)
$\Rightarrow$ pas aussi facile à résoudre...

Programmation dynamique

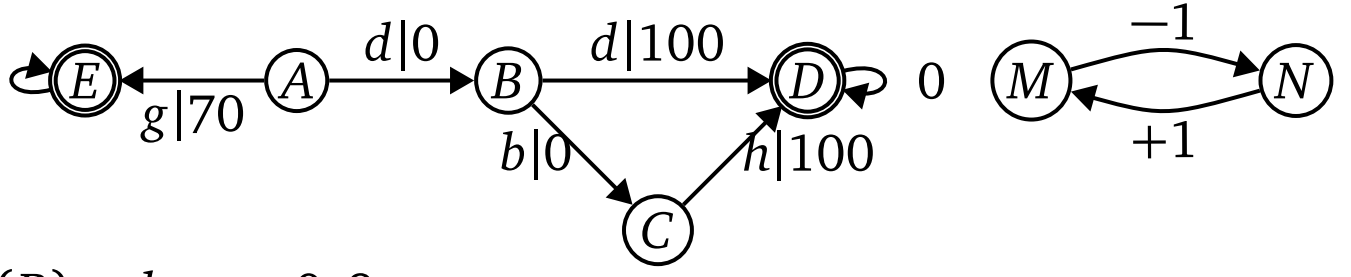
Ce terme regroupe (lorsqu'on parle de MDPs) des méthodes qui permettent de «*résoudre*» un MDP lorsqu'on ***connaît le modèle***.

Programmation dynamique : Évaluation de politique

Un algorithme itératif avec un tableau intermédiaire temp

1. $R(\pi, s) \leftarrow 0$ pour tout $s \in \mathcal{S}$;
2. Tant que pas fini faire :
   (a) pour tout $s \in \mathcal{S}$:
   $\text{temp}(s) \leftarrow r(s, \pi(s)) + \gamma R(\pi, \delta(s, \pi(s))$
   (b) $R(\pi, s) \leftarrow \text{temp}(s)$ pour tout $s \in \mathcal{S}$

Exemple.



$\pi(A) = d$, $\pi(B) = b$, $\gamma = 0,8$

- On peut montrer que ça converge …
- Convergence plus rapide si on n'utilise pas de tableau temp

Programmation dynamique : Amélioration de politique

Un autre algorithme itératif simple, pour améliorer petit à petit la politique.

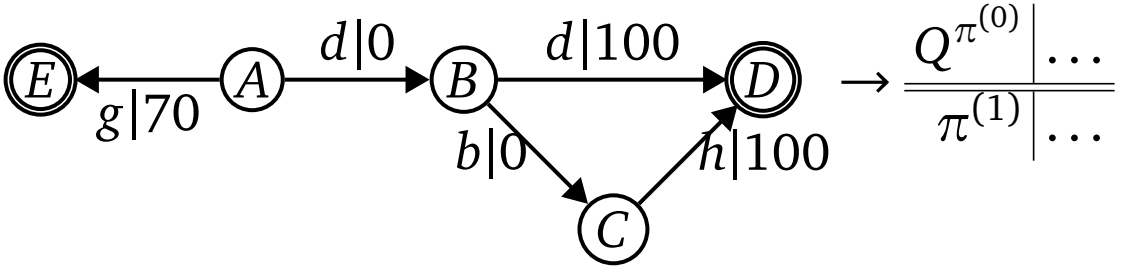
1. définir une première politique π ;
2. tant que «*pas fini*» faire :
   (a) pour tous $s \in \mathcal{S}, a \in \mathcal{A}(s)$: calculer $R(\pi, s)$;
   (b) pour tout $s \in \mathcal{S}$:
   $\pi'(s) \leftarrow \underset{a \in \mathcal{A}(s)}{\operatorname{argmax}} Q(\pi, s, a) = r(s, a) + \gamma R(\pi, \delta(s, a))$;
   (c) $\pi \leftarrow \pi'$;

Conditions d'arrêt : peu d'écart entre deux itérations successives (ou plus de temps)

Calcul de $R(\pi, s)$: Équations, ou algorithme précédent

Programmation dynamique : Amélioration de politique

Exemple.



$\rightarrow$

$Q^{\pi^{(0)}}$	$\dots$
$\pi^{(1)}$	$\dots$

Programmation dynamique : Amélioration itérative de valeurs

L'algorithme précédent fait évaluer complètement la politique courante à chaque itération $\Rightarrow$ coûteux !

Programmation dynamique : Amélioration itérative de valeurs

L'algorithme précédent fait évaluer complètement la politique courante à chaque itération $\Rightarrow$ coûteux !

Un algorithme itératif simple, qui améliore petit à petit une ***politique implicite*** :

1. $R(s) \leftarrow 0$ pour tout $s \in \mathscr{S}$;
2. tant que «*pas fini*» faire ://*(boucle principale)*
   pour tout $s \in \mathscr{S}$ faire :
   (a) pour tout $a \in \mathscr{A}(s) : Q(s,a) \leftarrow r(s,a) + \gamma R(\delta(s,a))$
   (b) $R(s) \leftarrow \max_{a \in \mathscr{A}(s)} Q(s,a)$;
3. pour tout $s \in \mathscr{S}$: //*(calcul de la politique finale π)*
   (a) pour tout $a \in \mathscr{A}(s) : Q(s,a) \leftarrow r(s,a) + \gamma R(\delta(s,a))$
   (b) $\pi(s) \leftarrow a \in \mathscr{A}(s)$ qui maximise $Q(s,a)$;

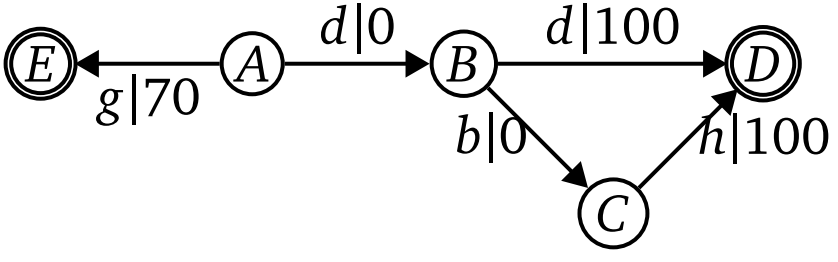
Programmation dynamique : Amélioration itérative de valeurs

L'algorithme précédent fait évaluer complètement la politique courante à chaque itération $\Rightarrow$ coûteux !

Un algorithme itératif simple, qui améliore petit à petit une ***politique implicite*** :

1. $R(s) \leftarrow 0$ pour tout $s \in \mathscr{S}$;
2. tant que «*pas fini*» faire ://*(boucle principale)*
   pour tout $s \in \mathscr{S}$ faire :
   (a) pour tout $a \in \mathscr{A}(s) : Q(s,a) \leftarrow r(s,a) + \gamma R(\delta(s,a))$
   (b) $R(s) \leftarrow \max_{a \in \mathscr{A}(s)} Q(s,a)$;
3. pour tout $s \in \mathscr{S}$: //*(calcul de la politique finale π)*
   (a) pour tout $a \in \mathscr{A}(s) : Q(s,a) \leftarrow r(s,a) + \gamma R(\delta(s,a))$
   (b) $\pi(s) \leftarrow a \in \mathscr{A}(s)$ qui maximise $Q(s,a)$;

Exemple.



Digression : Qu'appelle-t-on *programmation dynamique* ?

- La réponse la plus fréquente : «*Method for problem solving used in math and computer science in which large problems are broken down into smaller...*»
- Trouvé sur `stackoverflow.com` : «*Dynamic programming is when you use past knowledge to make solving a future problem easier*»
- Sur `topcoder.com` : «*an algorithmic technique which is usually based on a recurrent formula and one (or some) starting states. A subsolution of the problem is constructed from previously found ones. DP solutions have a polynomial complexity which assures a much faster running time than other techniques like backtracking, brute-force etc.* »

Digression : Qu'appelle-t-on *programmation dynamique* ?

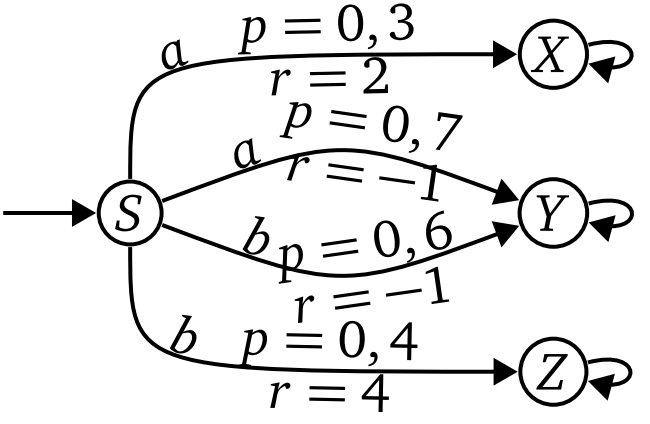
Explication de Bellman, dans son autobiographie :
(Eye of the Hurricane : An Autobiography, 1984)

> *We had a very interesting gentleman in Washington named Wilson. He was Secretary of Defense, and he actually had a pathological fear and hatred of the word research. . . .I had to do something to shield Wilson . . .from the fact that I was really doing mathematics. . .What title, what name, could I choose ? . . .planning, is not a good word for various reasons. I decided therefore to use the word "programming". I wanted to get across the idea that this was . . .multistage . . .[The word "dynamic"] also has a very interesting property as an adjective, and that it's impossible to use the word dynamic in a pejorative sense. Try thinking of some combination that will possibly give it a pejorative meaning. It's impossible.* ***Thus, I thought dynamic programming was a good name.*** *It was something not even a Congressman could object to. So I used it as an umbrella for my activities.*

Transitions non déterministes

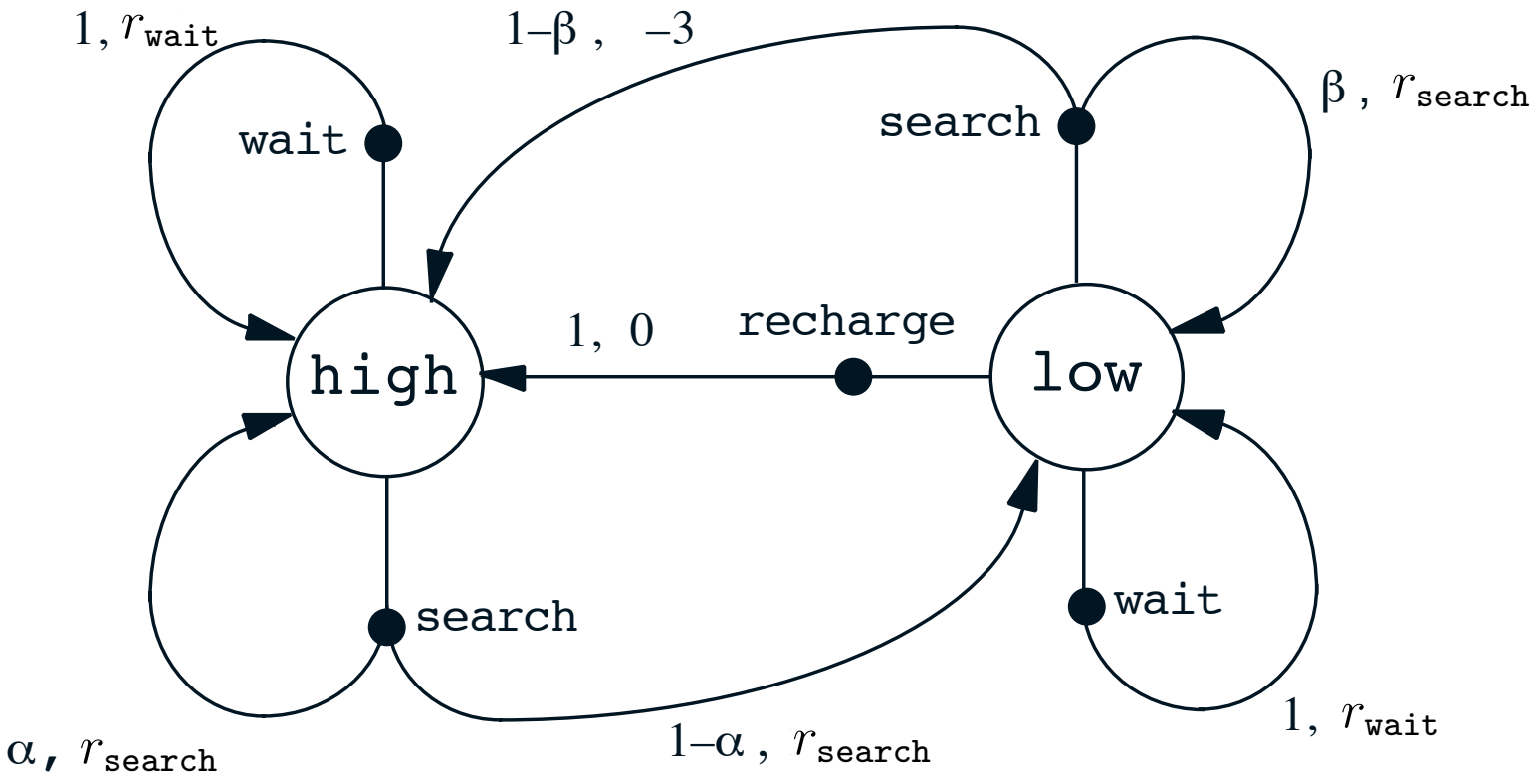
Dans de nombreuses situations, le résultat d'une action ne peut être prédit avec certitude. Par exemple :

- Jeux à plusieurs joueurs
  $\Rightarrow$ le résultat d'un coup d'un joueur dépend des coups des autres
- Robot en déplacement
  $\Rightarrow$ le résultat d'une accélération dépend de l'adhérence
  ...
- $p(s, a, s', r) \in [0, 1]$:
  probabilité d'aller en s' avec récompense $r \in \mathbf{N}$ lorsqu'on exécute l'action a dans l'état s



Transitions non déterministes

Exemple. Un robot recycleur de canettes [Sutton et Barto] :



- le robot peut attendre, chercher une canette, ou aller à la borne électrique ;

Transitions non déterministes

- il prend sa décision en fonction de la charge de la batterie, «*low*» ou «*high*»
- si le robot entreprend une recherche quand il est déchargé, il doit être «*sauvé*» : il passe dans l'état *high* mais à une pénalité de 3 ;
- s'il décide lui-même d'aller se recharger, il passe aussi dans l'état «*high*» mais n'a pas de pénalité
- r_{search} et r_{wait} sont les espérances des nombres de canettes trouvées quand le robot cherche ou attend.

Transitions non déterministes

Récompense à long terme ***moyenne*** pour une politique donnée π en partant d'un état s_0 :

$$R(\pi, s_0) = E[\sum_{i \geq 0} \gamma^i r(s_i, a_i \mid s_0)],$$

moyenne sur les séquences $(s_0, a_0, s_1, a_1, \ldots)$ possibles.

Transitions non déterministes

Récompense à long terme ***moyenne*** pour une politique donnée π en partant d'un état s_0 :

$$R(\pi, s_0) = E[\sum_{i \geq 0} \gamma^i r(s_i, a_i \mid s_0)],$$

moyenne sur les séquences $(s_0, a_0, s_1, a_1, \ldots)$ possibles.

En développant, en posant $a = \pi(s)$:

$$R(\pi, s) \; = \; \sum_{s', r} p(s, a, s', r)(r + \gamma R(\pi, s'))$$

$\Rightarrow$ on peut toujours résoudre un système d'équations pour calculer R

Transitions non déterministes : Évaluation itérative de politique

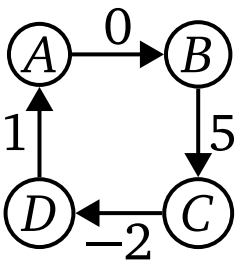
1. $R(\pi, s) \leftarrow 0$ pour tout $s \in \mathcal{S}$;
2. Tant que pas fini faire :
   (a) pour tout $s \in \mathcal{S}$:
   $\mathsf{temp}(s) \leftarrow \sum_{s',r} p(s, \pi(s), s', r)(r + \gamma R(\pi, s'))$
   (b) $R(\pi, s) \leftarrow \mathsf{temp}(s)$ pour tout $s \in \mathcal{S}$

Transitions non déterministes : Amélioration itérative de valeurs

1. $R(s) \leftarrow 0$ pour tout $s \in \mathscr{S}$;
2. tant que «*pas fini*» faire :
   pour tout $s \in \mathscr{S}$ faire :
   (a) pour tout $a \in \mathscr{A}(s)$:
   $Q(s,a) \leftarrow \sum_{s',r} p(s,a,s',r)(r + \gamma R(\pi, s'))$
   (b) $R(s) \leftarrow \max_{a \in \mathscr{A}(s)} Q(s,a)$;
3. pour tout $s \in \mathscr{S}$: *//(calcul de la politique finale π)*
   (a) pour tout $a \in \mathscr{A}(s)$: $Q(s,a) \leftarrow \sum_{s',r} p(s,a,s',r)(r + \gamma R(\pi, s'))$
   (b) $\pi(s) \leftarrow a \in \mathscr{A}(s)$ qui maximise $Q(s,a)$;

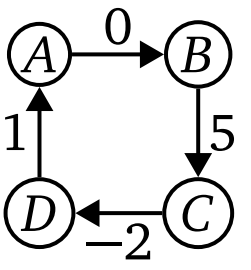
Exercices

Exercice 1. Calculer les valeurs des récompenses à long terme sur le graphe de transition ci-contre, en considérant un facteur $\gamma = 0,9$.



Exercices

Exercice 1. Calculer les valeurs des récompenses à long terme sur le graphe de transition ci-contre, en considérant un facteur $\gamma = 0,9$.

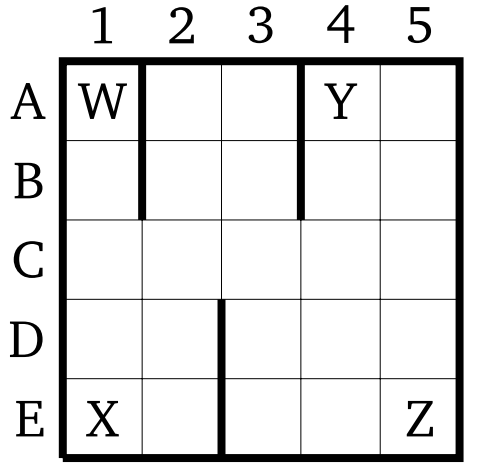


Exercice 2. Si on augmente toutes les récompenses immédiates d'un MDP d'une constante C, de combien sont augmentées les récompenses à long terme ?

Exercice 3. Donner les équations de Bellman pour le robot recycleur.

Exercices

Exercice 4. Un chauffeur de taxi d'une ville représentée par la grille 5 × 5 ci-contre peut prendre ou déposer des clients aux points W, X, Y et Z. Au début d'un épisode, le taxi est sur une case quelconque, et il y a un passage sur l'un des 4 points de charge / décharge, à prendre et déposer sur l'un des autres points de charge / décharge.



Les actions possibles sont « *Nord* », « *Sud* », « *Est* », « *Ouest* », « *Charger* », « *Déposer* ». Il y a une récompense de −1 pour chaque déplacement, de −10 pour une tentative de charge ou décharge au mauvais endroit, et de +20 quand on dépose le passager à sa destination.

Exercices

Question 4.1. Combien y a-t-il d'états possibles ?

Question 4.2. Dessiner le chemin du graphe de transitions allant d'un état où le taxi est en A3 et où un client attend en W jusqu'à l'état où le client est déposé à sa destination en Y.

Question 4.3. Donner les valeurs des récompenses à long terme de la politique optimale, en supposant un facteur $\gamma = 1$, pour chacune des cases lorsque :

1. il y a un client à prendre en W et à déposer en Y ;
2. il y a un client dans le taxi, à déposer en Y.

Exercice 5. Démontrer l'équation de Bellman pour la fonction $R(\pi, \cdot)$ dans le cas non déterministe.

Apprentissage par renforcement

Lorsqu'on connaît le MDP / les probabilités de transitions / les récompenses :

- *résoudre* le MDP = calculer une politique optimale π^*, ou la fonctions Q^* associée
- on peut le faire en résolvant les équantion de Bellman ou en utilisant les algorithmes de programmation dynamique.

Comment *résoudre* un MDP qu'on ne connaît pas à l'avance ?

Rappels : on parle d'apprentissage par renforcement quand

- l'environnement est (partiellement) inconnu
- l'environnement peu changer
- on peut interagir avec l'environnement

Il va falloir ***explorer*** le MDP, l'échantillonner.
On ne suppose plus qu'on connaît les fonctions r et δ.

Apprentissage par renforcement : Les N bandits

Un exemple classique

- un joueur devant N machines à sous
- à chaque partie, le joueur met 1€ dans l'une des machines
- chaque machine k a une espérance de gain ***inconnue du joueur***

$$E_k = \sum_{r \in \{-1\} \cup \mathbf{N}} p_k(r) r$$

Si le joueur connaît les E_k :

toujours jouer, dès le début, la machine avec le plus fort E_k

Si le joueur ne connaît pas les E_k : *explorer*

- essayer plusieurs fois toutes les machines
- chaque essaie coûte 1€

$\Rightarrow$ quelle stratégie pour converger vers la bonne machine
en «*perdant*» le moins d'argent possible

Nombreuses applications : essais cliniques, publicités sur pages web, ...

Apprentissage par renforcement : Les N bandits

Stratégie «*bornée*» on joue une fois chaque machine, puis on ne change plus : on joue toujours celle qui a eu la meilleure récompense au premier tour – même si elle ne rapporte quasiment plus rien !

Apprentissage par renforcement : Les N bandits

Stratégie «*bornée*» on joue une fois chaque machine, puis on ne change plus : on joue toujours celle qui a eu la meilleure récompense au premier tour – même si elle ne rapporte quasiment plus rien !

Pour faire plus malin, il faut mettre à jour la connaissance des machines au fur et à mesure du jeu : qualité de machine k à l'instant $t =$

$$Q_t(\mathsf{mach}_k) = \frac{\text{somme gains sur machine } k \text{ jusqu'à } t}{\text{nb fois où machine } k \text{ jouée jusqu'à } t}$$

$Q_t(\mathsf{mach}_k)$ = valeur estimée de la machine k après t parties = valeur estimée de E_k après t parties

Apprentissage par renforcement : Les N bandits

Stratégie «*gourmande*» / impatiente : à la $t+1$-ième partie, on choisit l'une des machines ayant la meilleure qualité courante :

$$\underset{k \in 1 \ldots N}{\operatorname{argmax}} Q_t(\mathsf{mach}_k)$$

Apprentissage par renforcement : Les N bandits

Stratégie «*gourmande*» / impatiente : à la $t+1$-ième partie, on choisit l'une des machines ayant la meilleure qualité courante :

$$\underset{k \in 1 \ldots N}{\operatorname{argmax}} Q_t(\mathsf{mach}_k)$$

Problème : on risque d'«*oublier*» une machine qui n'a quasiment jamais été testée mais qui est pourtant la meilleure.
Autrement dit : aucune garantie de convergence

Exemple. 3 machines, avec les gains suivants :

- machine 1 : 200€ 1 fois sur 100 ; -1€ 99 fois sur 100
- machine 2 : 2€ 1 fois sur 10 ; −0,5€ 9 fois sur 10
- machine 3 : 3€ 1 fois sur 10 ; −0,5€ 9 fois sur 10

⇒ la stratégie gourmande va en général converger vers la machine 3, dont le gain moyen est de −0,15€, alors que la machine 1 a un gain moyen de 0€.

Apprentissage par renforcement : Les N bandits

Stratégie ϵ-«*gourmande*» : à la $t+1$-ième partie, on choisit

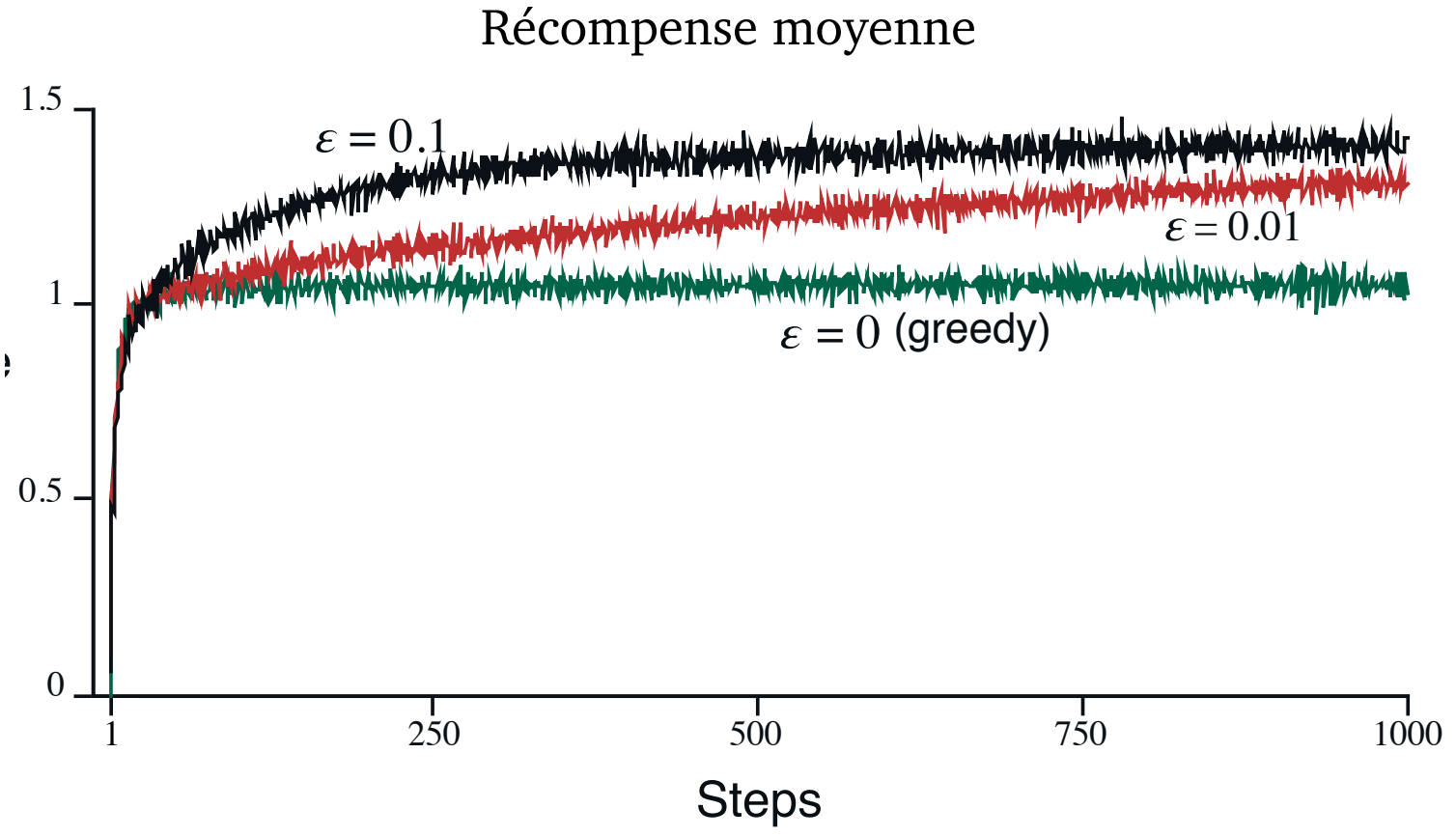
- avec une probabilité $1-\epsilon$: $\text{argmax}_{k\in 1\ldots N} Q_t(\mathsf{mach}_k)$ = exploitation ;
- avec une probabilité ϵ : une machine tirée au hasard = exploration.

$\Rightarrow$ on converge, avec une probabilité de 1, vers la machine ayant la meilleure espérance de gains.

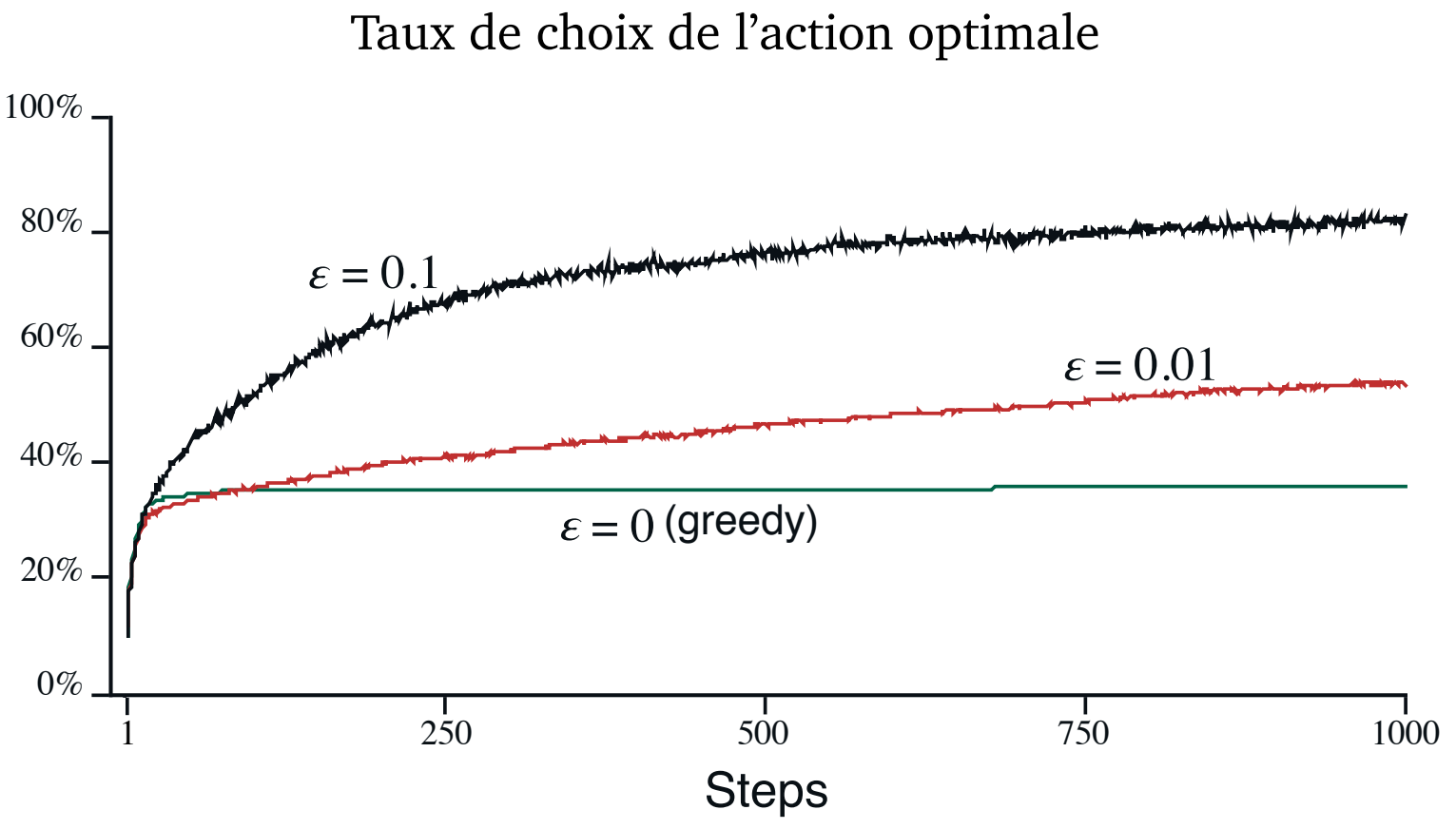
ϵ représente un compromis entre l'exploiration de la connaissance déjà acquise et l'exploration du système.

Apprentissage par renforcement : Les N bandits

Expérience de Sutton et Barto, moyennes sur 2000 problèmes avec $N = 10$



Apprentissage par renforcement : Les N bandits



Remarque. Avec un taux de 0,01 on finit par avoir de meilleurs résultats

Apprentissage par renforcement : Les N bandits

Mise à jour de $Q_t = \frac{1}{t}\sum_{i=1}^{t} R_i$:

$$Q_t = Q_{t-1} + \frac{1}{t}(R_t - Q_{t-1}) = (1 - \frac{1}{t})Q_{t-1} + \frac{1}{t}R_t$$

(Vérifier...)

Apprentissage par renforcement : Les N bandits

Mise à jour de $Q_t = \frac{1}{t}\sum_{i=1}^{t} R_i$:

$$Q_t = Q_{t-1} + \frac{1}{t}(R_t - Q_{t-1}) = (1 - \frac{1}{t})Q_{t-1} + \frac{1}{t}R_t$$

(Vérifier...)

$\Rightarrow$ Un algorithme d'apprentissage par renforcement simple

1. pour tout $k \in 1 \dots N$: $Q(\mathsf{mach}_k) \leftarrow 0$; $N(\mathsf{mach}_k) \leftarrow 0$;
2. tant que *pas fini* :
   (a) $k \leftarrow \begin{cases} \mathrm{argmax}_{k \in 1 \dots N} Q(\mathsf{mach}_k) \text{ avec proba. } 1-\epsilon \\ \mathsf{random}(1 \dots N) \text{ avec proba. } \epsilon \end{cases}$
   (b) jouer machine k, observer récompense R ;
   (c) $N(\mathsf{mach}_k) \leftarrow 1 + N(\mathsf{mach}_k)$;
   $Q(\mathsf{mach}_k) \leftarrow (1-\alpha)Q(\mathsf{mach}_k) + \alpha R$;

- $\alpha = 0$: pas de mise à jour ;
- $\alpha = 1$: pas de mémoire ;
- $\alpha = \frac{1}{N(\mathsf{mach}_k)}$: les observations comptent de moins en moins

Apprentissage par renforcement : Cas général

On introduit de l'exploration dans les algorithmes de programmation dynamique.

Apprentissage par renforcement : Cas général

Un algorithme générique d'apprentissage par renforcement

1. initialiser une représentation du MDP
2. répéter
   (a) choisir un état de départ $s \in \mathscr{S}$;
   (b) répéter
      i. choisir une action $a \in \mathscr{A}(s)$;
      ii. exécuter a en observant récompense r + nouvel état s' ;
      iii. mettre à jour la représentation du MDP ;
      iv. $s \leftarrow s'$.

Remarque. On n'utilise plus de fonctions de récompense et de transition, mais des observations.

Les méthodes d'apprentissage par renforcement diffèrent les unes des autres par :

- la représentation du MDP (fonctions R, Q, ...)
- la stratégie de choix d'une action en 2(c)ii

Apprentissage par renforcement : Cas général

- la stratégie de mise à jours de la représentation du MDP en 2(c)iii

Apprentissage par renforcement : Cas général

Q-learning :

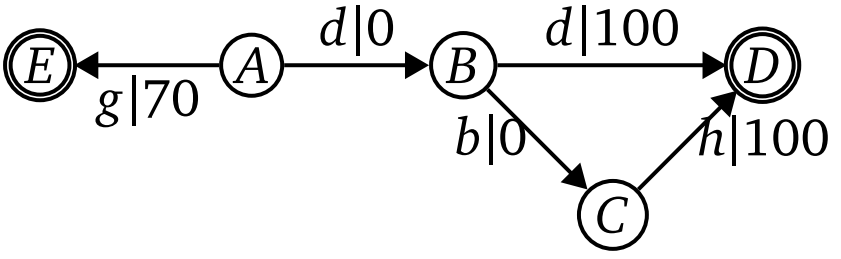
1. $Q(s,a) \leftarrow 0$ pour tout s, $a \in \mathscr{A}(s)$;
2. répéter
   (a) choisir un état de départ $s \in \mathscr{S}$;
   (b) répéter
      i. choisir une action $a \in \mathscr{A}(s)$, stratégie ϵ-gourmande ;
      ii. exécuter a en observant récompense r + nouvel état s' ;
      iii. $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a' \in \mathscr{A}(s')} Q(s',a'))$;
      iv. $s \leftarrow s'$.

$\alpha \in [0,1]$ est le ***taux d'apprentissage***
par exemple : $\alpha = 1/(1 + \mathsf{nb\_visites}(s,a))$

Apprentissage par renforcement : Cas général

L'algorithme de Q-learning ne précise pas combien de fois on effectue la boucle de mise à jour de Q : en fait, l'agent va alterner des phases d'exploration, durant lesquelles il effectue cette boucle, et des phases où il utilise ce qu'il a appris, même si c'est imparfait, pour réaliser certains buts.

Apprentissage par renforcement : Cas général



Apprentissage par renforcement : Cas général

Propriété. Convergence de l'algorithme de Q-learning :

- Si les récompenses immédiates sont bornées, et
- si $0 \leq \gamma < 1$, et
- si le choix de l'action à effectuer dans la boucle est tel que toute paire état/action sera évaluée infiniment souvent

alors la fonction Q apprise par l'algorithme converge vers la fonction $Q(\pi^*, \cdot, \cdot)$ d'une politique π^* optimale.

Preuve.....

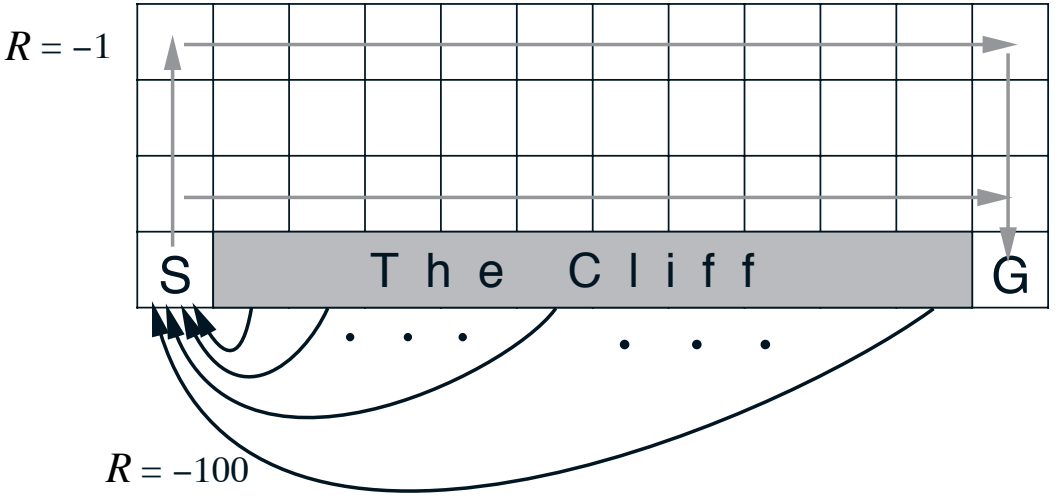
Apprentissage par renforcement : Cas général

Amélioration possible : lorsque les récompenses apparaissent peu souvent, on peut rétropropager Q pour toutes les paires état/action rencontrées lors d'un épisode.

Sur l'exemple introductif, en considérant des épisodes d'apprentissages qui commencent en (A,4) : quel est le nombre minimum d'épisodes nécessaire pour apprendre complètement la fonction Q sans / avec rétropropagation ?

Apprentissage par renforcement : Cas général

Exemple. Une tâche épisodique, sans dépréciation ($\gamma = 1$) :



La politique optimale passe le long de la falaise.
Mais exploration : de temps en temps (avec proba ϵ) l'agent saute ! !
Impossible d'apprendre cela avec l'algorithme de Q-learning
(Pourquoi ?)

Apprentissage par renforcement : Cas général

SARSA

1. $Q(s,a) \leftarrow 0$ pour tout s, $a \in \mathscr{A}(s)$;
2. répéter
   (a) choisir un état de départ $s \in \mathscr{S}$;
   (b) choisir une action $a \in \mathscr{A}(s)$, stratégie ϵ-gourmande ;
   (c) répéter
      i. exécuter a en observant récompense r + nouvel état s' ;
      ii. choisir une action $a' \in \mathscr{A}(s')$, stratégie ϵ-gourmande ;
      iii. $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma Q(s',a'))$;
      iv. $s \leftarrow s'$, $a \leftarrow a'$.

Apprentissage par renforcement : Cas général

SARSA

1. $Q(s,a) \leftarrow 0$ pour tout s, $a \in \mathscr{A}(s)$;
2. répéter
   (a) choisir un état de départ $s \in \mathscr{S}$;
   (b) choisir une action $a \in \mathscr{A}(s)$, stratégie ϵ-gourmande ;
   (c) répéter
      i. exécuter a en observant récompense r + nouvel état s' ;
      ii. choisir une action $a' \in \mathscr{A}(s')$, stratégie ϵ-gourmande ;
      iii. $Q(s,a) \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma Q(s',a'))$;
      iv. $s \leftarrow s'$, $a \leftarrow a'$.

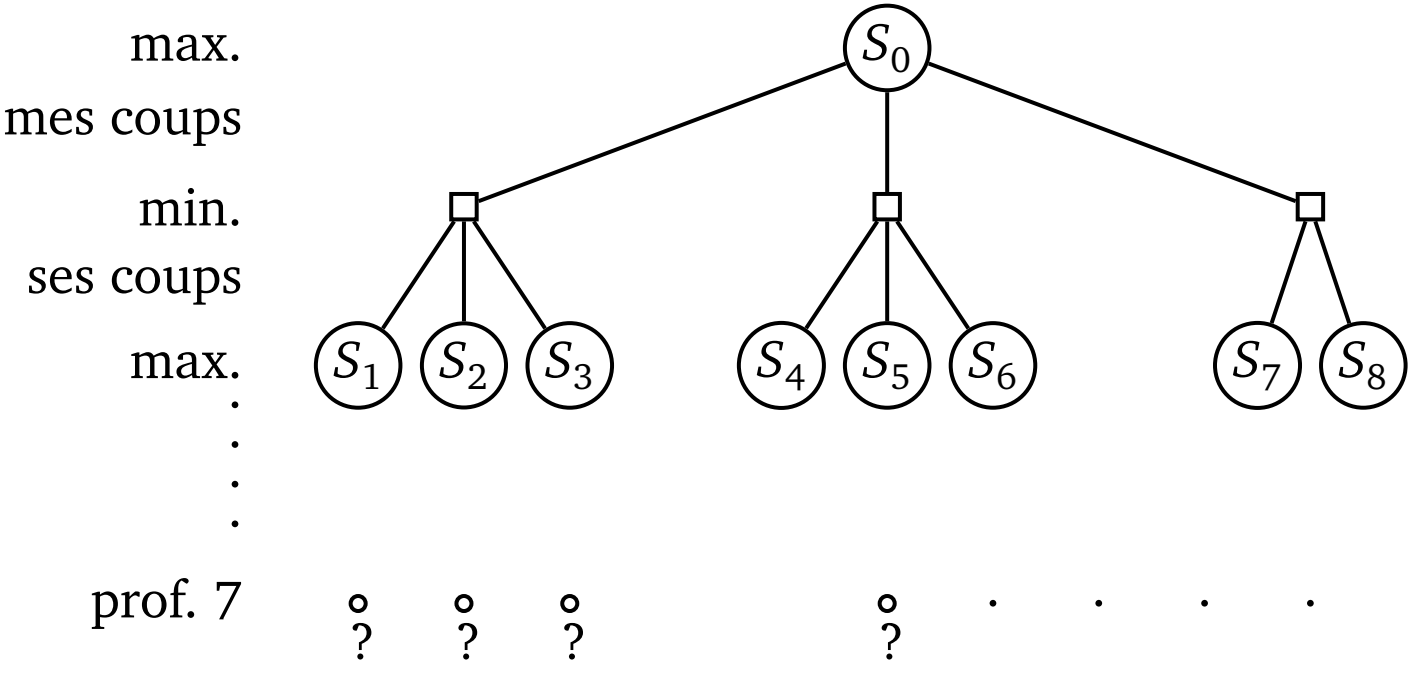
- Algorithme «*on policy*» : on met à jour avec la qualité de l'action correspondant à l'action déterminée par la politique
- Avec cet algorithme, l'algorithme converge vers un chemin plus éloigné de la falaise.
- Si ϵ tends vers 0, les deux algorithmes tendent vers la politique optimale.

Apprentissage par renforcement : MCTS

MCTS = Monte-Carlo Tree Search
Algorithme utilisé pour les jeux – tâches épisodiques
Proposé pour le jeu de Go dans les années 2000

Apprentissage par renforcement : MCTS

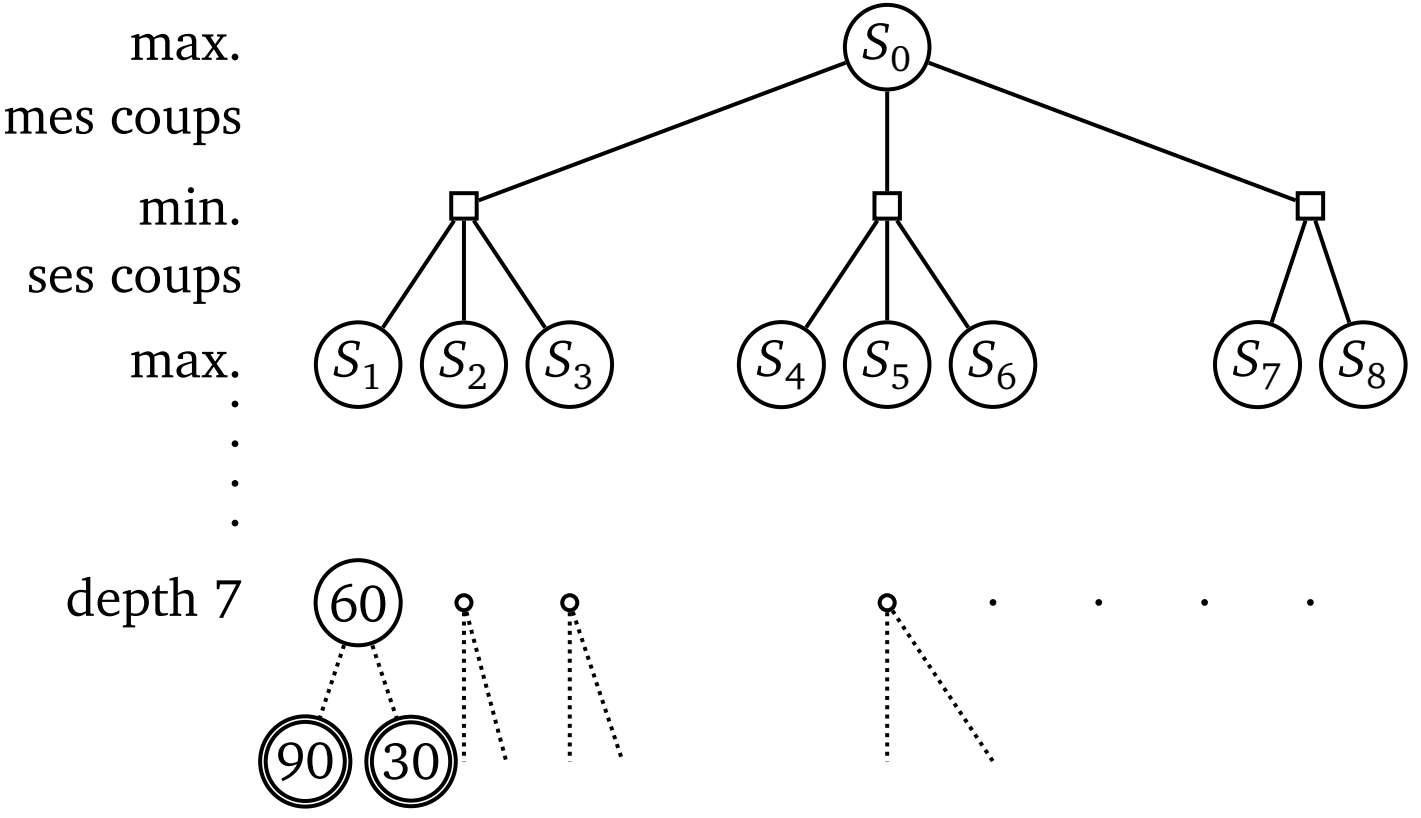
Arbre de recherche à horizon fixé avec fonction heuristique



Marche bien pour jeux à 2, somme nulle, si bonne heurisitique

Apprentissage par renforcement : MCTS

Évaluation «*Monte-Carlo evaluation*» : à chaque fois qu'on a besoin de la valeur d'un nœud : simuler k fin de parties aléatoires



Apprentissage par renforcement : MCTS

- Pratique quand on n'a pas de bonne heuristique
  (Go, «*General Game Playing*»)
- Méthode optimiste : suppose que l'adversaire joue au hasard
- CadiaPlayer a gagné trois années de suite la compétition de GGP dans les années 2000

Monte-Carlo Tree Search (MCTS)

- pas d'horizon fixé
- on peut perdre moins de temps à évaluer les séquences peu prometteuses
- on utilise l'information / les évaluations dès que trouvées
- on ne développe que les branches prometteuses

$\Rightarrow$ on peut descendre plus profondément dans ces branches

Apprentissage par renforcement : MCTS

Arbre de recherche étendu en cycles de 4 étapes

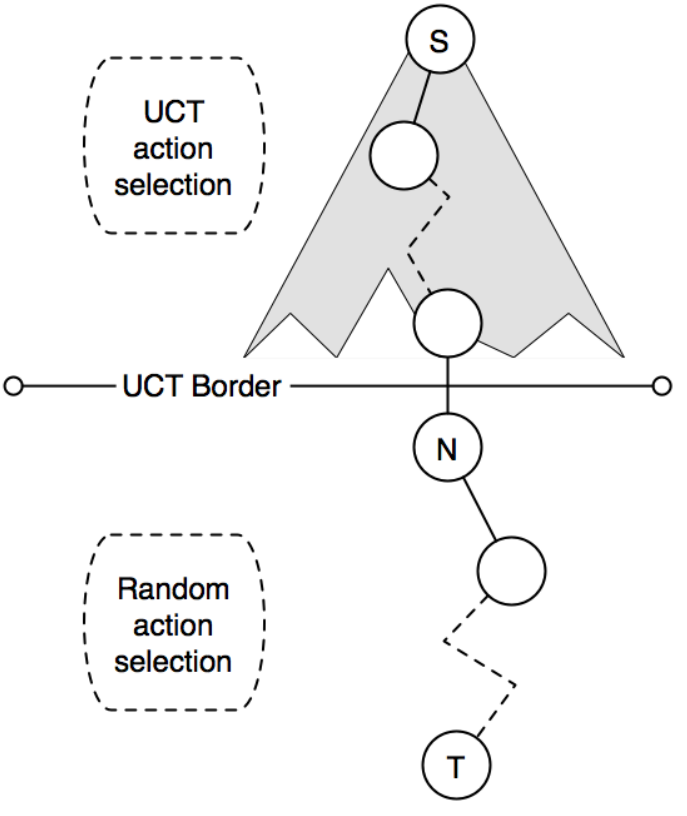
Sélection: commençant à la racine / état courant, on descend dans l'arbre d'après les nb visites et récompenses mesurées jusqu'ici, jusqu'à arrivé à un nœud non étendu.

Expansion: On ajoute les successeurs de ce nœud

Simulation: Depuis ce nœud jusqu'à un état terminal

Backpropagation:

Rétropropagation = mise à jour en amont des nb visites et qualités sur la séquence parcourue



Apprentissage par renforcement : MCTS

(fig. from CadiaPlayer - Bjornsson and Finsson, 2009)

Apprentissage par renforcement : MCTS

Pour la sélection à l'état s : compromis exploitation / exploration
sélectionner l'action a telle que

$$\underset{a}{\operatorname{argmax}}\{\mathsf{value}(s,a) + C\sqrt{\frac{\log N(s)}{N(s,a)}}\}$$

Généralisation des renforcements

Dans la pratique : difficile / impossible de mémoriser une table de Q ($|\mathscr{S}| \times E[|\mathscr{A}(s)|] \sim 10^{29}$ pour Othello !)

$\rightarrow$ on apprend une approximation de Q.

Par exemple : pour Othello, un réseau de neurones :

- 64 unités en entrées
- une couche cachée
- 64 unités en sorties : $Q(s, a)$ pour chacun des coups possibles a

$\rightarrow$ chaque fois qu'on doit mettre Q à jour, on applique une méthode de rétropropagation.

Généralisation des renforcements

1. Initialiser Q ;
2. répéter
   (a) choisir $s \in \mathscr{S}$; $\mathscr{E} \leftarrow \emptyset$;
   (b) répéter :
      i. choisir $a \in \mathscr{A}(s)$;
      ii. $r \leftarrow r(s,a)$; $s' \leftarrow \delta(s,a)$
      iii. $\alpha \leftarrow 1/(1 + \mathsf{nb\_visites}(s,a))$;
      iv. $q \leftarrow (1-\alpha)Q(s,a) + \alpha(r + \gamma \max_{a' \in \mathscr{A}(s)} Q(s',a'))$;
      v. $\mathscr{E} \leftarrow \mathscr{E} \cup \{((s,a),q)\}$;
      vi. $s \leftarrow s'$;
   (c) « *ré-apprendre*» Q à partir des exemples de $\mathscr{E}$;

Remarque : en 2c, il ne s'agit pas d'apprentissage de classifieur, mais de ***régression***.

Problème : dès lors qu'on a une telle représentation appochée de Q, la méthode risque de ne pas converger.

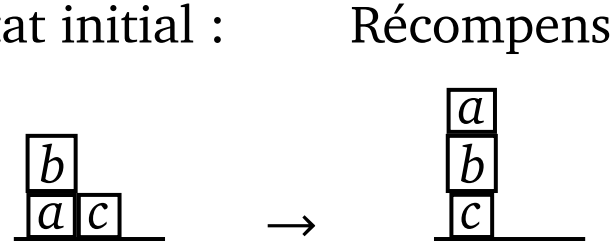
Apprentissage par renforcement relationnel

La représentation de Q, et la méthode d'apprentissage, dépendent de la représentation des états et des actions.

Lorsque les états sont représentés à l'aide de relations, et non plus de simples paires (attribut,valeur), on parle d'apprentissage par renforcement relationnel.

Apprentissage par renforcement relationnel

Exemple : le monde des cubes



Actions possibles : déplacement des cubes – si rien posé dessus ; par exemple, dans l'état initial, deplacer(b, c).

Apprentissage par renforcement relationnel

Exemple enregistré :
$\text{but} = \text{sur}(a, b)$,
$\text{etat} = \{\text{sur\_table}(a), \text{sur\_table}(c), \text{sur}(b, a), \text{libre}(b), \text{libre}(c)\}$,
$\text{action} = \text{deplacer}(b, c)$,
$\text{q} = 0.81$
...

Exercices

Exercice 6. Un agent se déplace dans le monde dessiné ci-contre : il y a en général quatre actions possibles (gauche, droite, haut, bas), sauf sur les bords. Chaque action à une récompense de -1 sauf les actions qui font arriver sur la case F3, marquée \$, qui provoquent une récompense de 100.

	A	B	C	D	E	F	G	H
5	↓	↓	↓	↓	↓	↓	↓	↓
4	↓	↓	↓	↓	↓	↓	↓	↓
3	↓	↓	↓	↓	↓	\$	↓	↓
2	↓	↓	↓	↓	↓	↑	↓	↓
1	→	→	→	→	→	↑	←	←

On considère d'abord une politique π_0 indiquée sur dessin : il se dirige toujours vers le bas, sauf quand il est sur la ligne 1 ou quand il est sur la case F2

Question 6.1. Avec cette politique π_0, quelles sont, en fonction de γ, les récompenses à long terme pour les cases A5 et H1 ?

Question 6.2. En supposant qu'on peut calculer $Q(\pi_0, s, a)$ pour toutes les paires (état,action), comment peut-t-on améliorer la politique π_0 ?

On considère qu'il y a maintenant un vent fort du bas vers le haut sur les colonnes D, E et F : lorsque l'agent effectue une action depuis une case

de ces colonnes, il atterrit une case plus haut que la case prévue, dans la limite des cases de la grille. Par exemple, si le robot part à gauche depuis la case E3, il atterrit en fait en F4 ; mais s'il part à gauche depuis E5, il atterrit en F5 (car il ne peut sortir de la grille).

Question 6.3. Quelle sont maintenant les récompenses à long terme, toujours avec la politique π_0, des cases H5 et A5 ?

Question 6.4. En supposant un coefficient $\gamma = 1$, dessinez une politique optimale.