

# **Apprentissage par renforcement (Reinforcement Learning (RL)) Chapitre 3: Le problème de RL**

**T. AL-ANI  
A<sup>2</sup>SI-ESIEE-PARIS**

Adapté de (R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction <http://www-anw.cs.umass.edu/~rich/book/the-book.html>) par Tarik AL ANI,  
A<sup>2</sup>SI-ESIEE-PARIS

## Chapitre 3: Le Problème de RL

---

Objectifs de ce chapitre :

- ❑ décrire le problème de RL
- ❑ présenter une forme idéalisée du problème de RL pour lequel nous devons préciser des résultats théoriques;
- ❑ introduire des principes clés mathématiques: fonctions de la valeur et les équations de Bellman;
- ❑ décrire les compromis entre l'applicabilité et l'attrance mathématique.

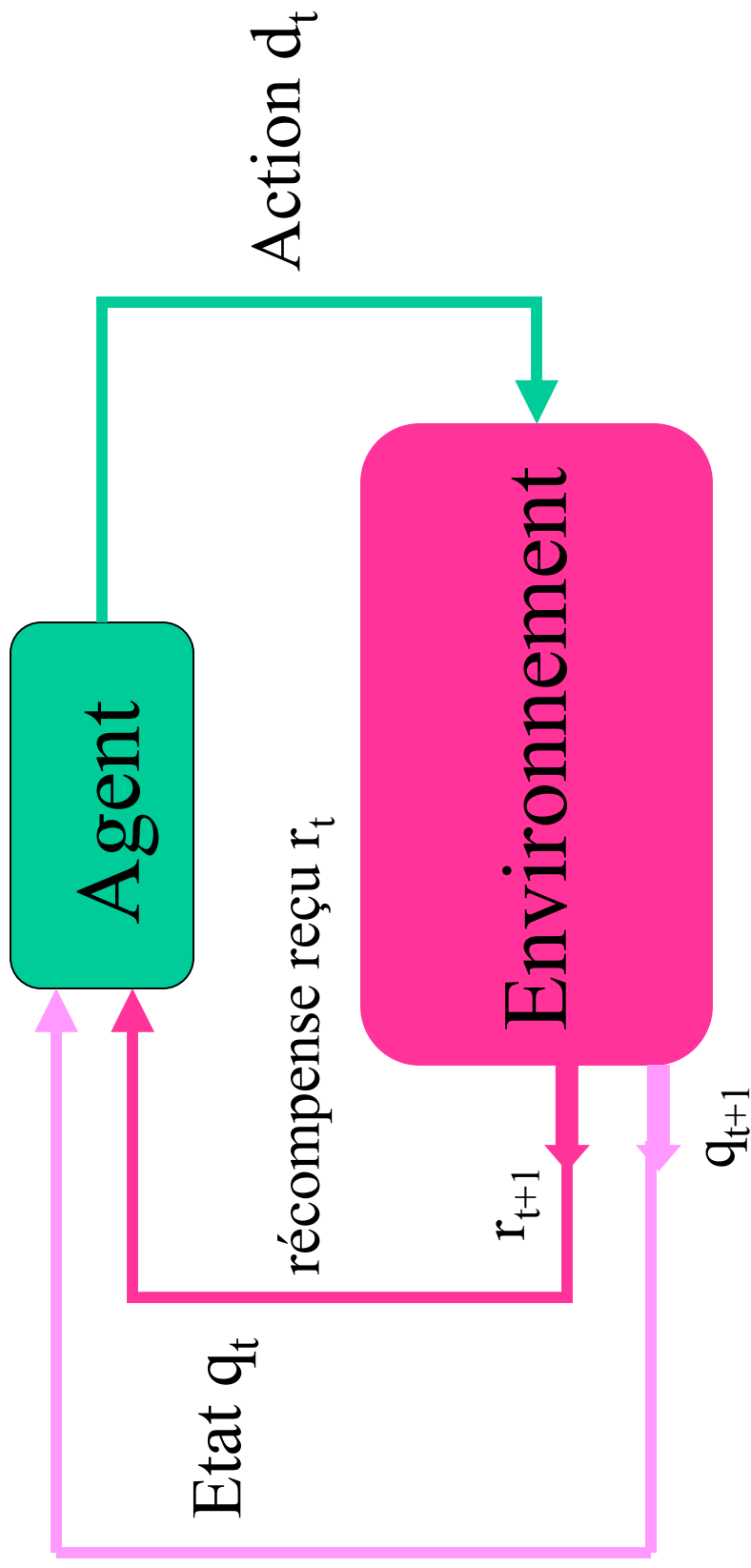
# Sommaire

---

- Interaction agent- environnement
  - États
  - Actions
  - Récompenses
- Politique: règle stochastique pour sélectionner les actions
- Revenue: la fonction du revenu future que l'agent essaie de maximiser
- Tâches épisodiques et continues
- Propriété de Markov
- Processus de Décision de Markov
  - Probabilités de Transition
  - Recompenses espérées
- Fonctions de la Valeur pour une politique
  - Fonction de la valeur de l'action pour une politique
  - Fonction de la Valeur Optimal de l'état
  - Fonction de la Valeur Optimal de l'action
- Fonctions de la Valeur Optimal
- Politiques optimales
- Équation de Bellman
- La nécessité d'effectuer une approximation

# Interface Agent-Environnement

---



# Interface Agent-Environnement

---

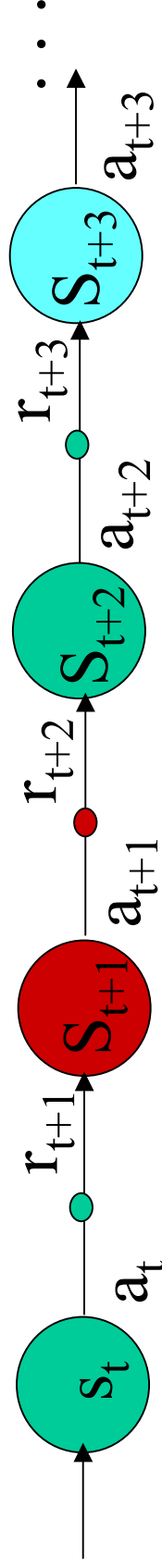
L'agent et l'environnement interagissent aux pas du temps discrets :  $t = 0, 1, 2, \dots$

l'agent observe l'état au pas  $t$  :  $s_t \in S$

produit une action au pas  $t$  :  $a_t \in A(s_t)$

reçoit une récompense :  $r_{t+1} \in \mathfrak{R}$

et évolue vers un état :  $s_{t+1}$



# L'agent apprend une Politique (Policy)

---

**Politique** au pas  $t, \pi_t$  :

une association des états aux probabilités de l'action

$\pi_t(s, a)$  = probabilité que  $a_t = a$  quand  $s_t = s$

- Les méthodes de RL spécifient comment l'agent modifie sa politique en fonction de son expérience.
- L'objectif de l'agent est de maximiser la récompense à long terme.

## **Avoir le bon Degré d'Abstraction**

- En temps réel, les pas du temps ne sont pas nécessairement des intervalles fixes, ils peuvent être des pas successifs arbitraires de prises de décisions et d'actions .
- Les actions peuvent être de bas niveau (e.g., voltages aux moteurs), ou haut niveau (e.g., accepter un job offert), etc. Les actions peuvent être entièrement mentales, e.g. certaines actions peuvent contrôler ce que l'agent doit choisir, ou où il doit fixer son attention.

## Avoir le bon Degré d'Abstraction

---

□ Les états peuvent être des “sensations” de bas niveau, tel qu’une mesure directe d’un capteur, ou ils peuvent être abstraits, tel que des descriptions symboliques des objets dans une pièce, ou basés sur une mémoire des sensations dans le passé, ou même entièrement mentales ou subjectifs (e.g., l’état d’être “surpris” or “perdu”).

# Avoir le bon Degré d'Abstraction

---

- ❑ Un animal ou un robot, est général, constitué de plusieurs agents RL et autres composants.
- ❑ Certaines parties de l'agent qui ne peuvent pas être modifiées arbitrairement peuvent être considérées comme une partie de l'environnement. Le calcul de la récompense est dans l'environnement de l'agent parce que l'agent ne peut pas la modifier arbitrairement.
- ❑ L'environnement n'est pas nécessairement inconnu à l'agent, il est surtout non complètement contrôlable.

# Domaines d'application

---

Domaines très variés :

- commande des systèmes,
- robotique,
- informatique,
- optimisation combinatoire (réseaux par exemple),
- communication,
- des actions quotidiennes dans la vie,
- jeux,
- ...

# Un Exemple

---

Bio-réacteur : Déterminer moment par moment les températures et les taux de de mélange.

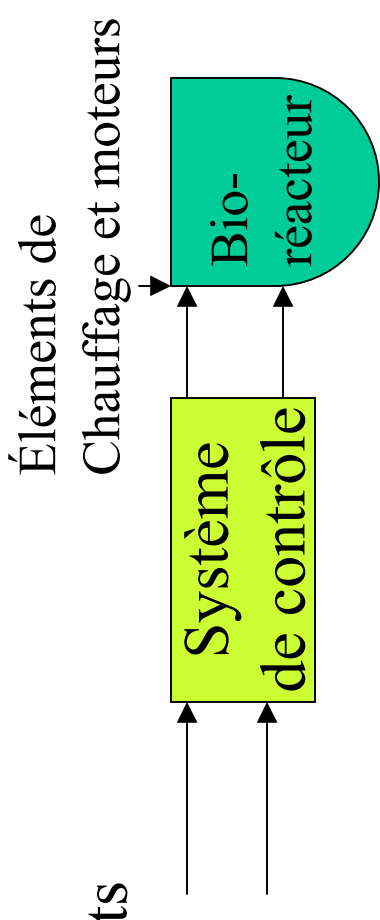
**Décisions** : vecteurs à deux composants

- Températures cibles
- Taux de mélange cible

**États** : vecteur à plusieurs composants

- Thermocouples
- autres capteurs
- Entrées symboliques qui représentent les ingrédients dans la cuve et les mélange cible.

**Récompense** : des mesures, moment par moment, du taux aux quels le mélange chimique utile est produit par le réacteur.



## Autre exemple

---

Robot mobile de recyclage : assembler des cannettes vides de boisson dans un atelier (tapis roulant par exemple). Ce robot possède des capteurs pour détecter les cannettes, un bras équipé d'un préhenseur pour saisir ces cannettes et les ranger dans un casier. Ce robot fonctionne grâce à des batteries rechargeables. Le système de contrôle du robot possède des composants pour interpréter l'information sensorielle, pour naviguer, et pour contrôler le bras et le préhenseur.

## Autre exemple (suite)

---

Des décisions de haut niveau concernant la façon de chercher les canettes sont effectuées par un agent effectuant un apprentissage par renforcement en se basant sur le niveau courant de la charge de la batterie.

L'agent doit décider si le robot devrait

- chercher activement une canette pour une certaine période d temps,
- rester stationnaire et attendre que quelqu'un lui ramène une canette,
- retourner à sa base pour recharger ses batteries.

## Objectifs (Goals) et Récompenses (Rewards)

- Est-ce qu'un signal scalaire de récompense  $r$  est une notion adéquate pour un objectif ?— peut être pas, mais il est étonnement flexible.  
Une récompense est notre moyen de communication avec un agent de **ce que nous** souhaitons réaliser, et **non pas comment nous** souhaitons le réaliser.
- **Exemple 1** : Pour apprendre à un robot à marcher, certains chercheurs ont fournis une récompense  $r$ , à chaque pas dans le temps, qui est proportionnelle au mouvements du robot vers l'avant.

## Objectifs et Récompenses

---

□ Est-ce-qu'un signal scalaire de récompense  $r$  est une notion adéquate pour un objectif ? — peut être pas, mais il étonnement flexible. (suite)

**Exemple 2** : Pour apprendre à un robot comment échapper d'un labyrinthe, la récompense  $r$  est souvent zéro jusqu'à ce qu'il échappe quand  $r$  devient  $+1$ .

Autre approche : donner  $-1$  pour chaque pas avant qu'il s'échappe. Ceci encourage le robot à échapper le plus vite possible.

# Objectifs et Récompenses

---

□ Est-ce-qu'un signal scalaire de récompense  $r$  est une notion adéquate pour un objectif ?— peut être pas, mais il étonnement flexible. (suite)

**Exemple 3** : Pour le robot de recyclage, la récompense  $r$  est souvent zéro mais elle devient +1 pour chaque canette vide rangée dans un casier par le robot. On pourrait aussi donner au robot des récompenses négatives (pénalisation) quand il produit un choc sur les installations ou autres objets dans l'atelier.

**Exemple 4** : Pour un jeu d'échec, +1 gagner, -1 perdre, 0 fin jeu sans aucun gagnant.

Dans tous ces exemples l'agent apprend pour maximiser la **récompense totale à long terme (long run total reward)**.

# Objectifs et Récompenses

---

- Un objectif doit être à l'extérieur du contrôle direct de l'agent — à l'extérieur de l'agent.
- L'agent doit être capable de mesurer les réussites:
  - explicitement;
  - fréquemment pendant la durée de sa vie.

# Revenues (Returns)

---

Supposons la séquence suivante des récompenses après le pas  $t$  :

$$r_{t+1}, r_{t+2}, r_{t+3}, \dots$$

**Que souhaitons nous maximiser?**

En général, nous souhaitons maximiser le **revenue espéré** (**expected return**),  $E\{R_t\}$ , pour chaque pas  $t$ .

**Tâches épisodiques (Episodic tasks)** : une interaction peut naturellement être découpée en épisodes, e.g., jouer un jeu, se balader dans une labyrinthe.

$$R_t = r_{t+1} + r_{t+2} + \dots + r_T, \quad (1)$$

où  $T$  est le dernier pas auquel un **état terminal (terminal state)** est arrivé, à la fin de l'épisode.

# Revenues pour des Tâches Continues (Continual Tasks)

---

## Tâches Continues (Continual tasks) :

Interaction ne possédant pas des épisodes naturelles. Dans ce cas T et R pourraient être infinis et ainsi l'équation (1) n'est plus applicable.

## Discounted return:

$$R_t = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \dots = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad (2)$$

Où  $\gamma, 0 \leq \gamma \leq 1$  est le **taux de réduction (discount rate)** qui détermine la valeur courante des revenus dans le future : une récompense reçue k pas de temps dans le future aura une valeur égale à  $\gamma^k$  fois sa valeur immédiate.

# Revenues pour des Tâches Continues (Continual Tasks)

---

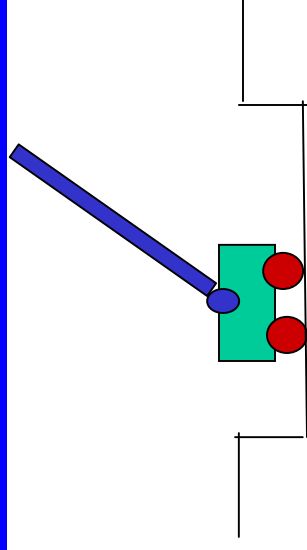
Si  $\gamma < 0$ , la somme dans l'équation (2) aura une valeur fini si la séquence  $\{r_k\}$  est bornée.

Si  $\gamma \rightarrow 0$ , l'agent est «**myope (myopic)**» en un sens qu'il est concerné uniquement par la maximisation des récompenses immédiates : son objectif dans ce cas, est d'apprendre comment choisir  $a_t$  pour maximiser uniquement  $r_{t+1}$ .

Si  $\gamma \rightarrow 1$ , l'agent est «**hypermétrope (far-sighted)**» en un sens qu'il est concerné très fortement par la maximisation des récompenses dans le future.

# Un Example

---



**Éviter un échec** : le pendule tombe s'il dépasse un angle critique ou si le chariot heurte la fin de course.

**Tâche épisodique** où l'épisode finit au moment de tomber :

récompense = +1 avant échouer

$\Rightarrow$  revenue = nombre des pas avant échouer

**Tâche continue** avec une revenue réduite:

récompense =  $-1$  à l'échec; 0 autrement

$\Rightarrow$  revenue =  $-\gamma^k$ , pour  $k$  pas avant l'échec

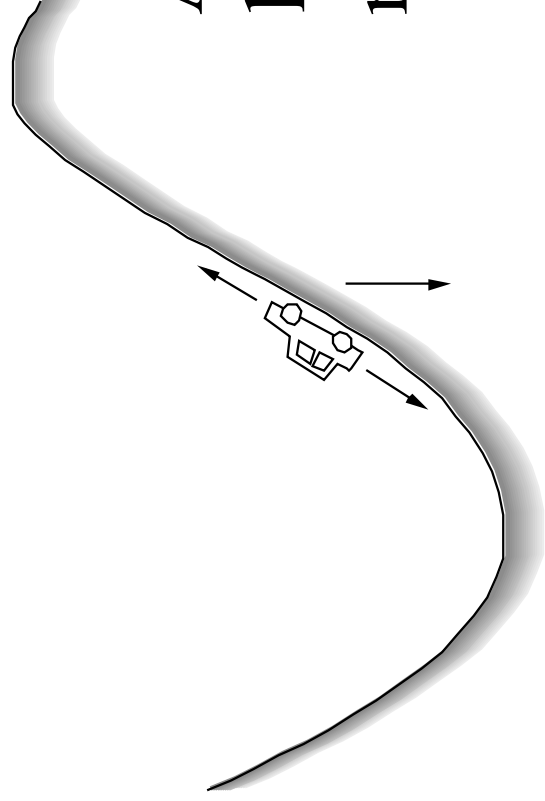
**Dans les deux cas, le revenue est maximisé en évitant un échec pour le plus longtemps possible.**

Adapté de (R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction <http://www-anw.cs.umass.edu/~rich/book/the-book.html>) par Tarik AL ANI, A<sup>2</sup>SI-ESIEE-PARIS



## Autre Exemple

---



Atteindre le sommet de  
la colline le plus  
rapidement possible.

récompense =  $-1$  à chaque pas où on est pas au sommet de la colline  
 $\Rightarrow$  revenue =  $--$  nombre de pas avant atteindre le sommet

**Le revenue est maximisé en minimisant nombre  
de pas pour atteindre le sommet.**



## Une notation unifiée

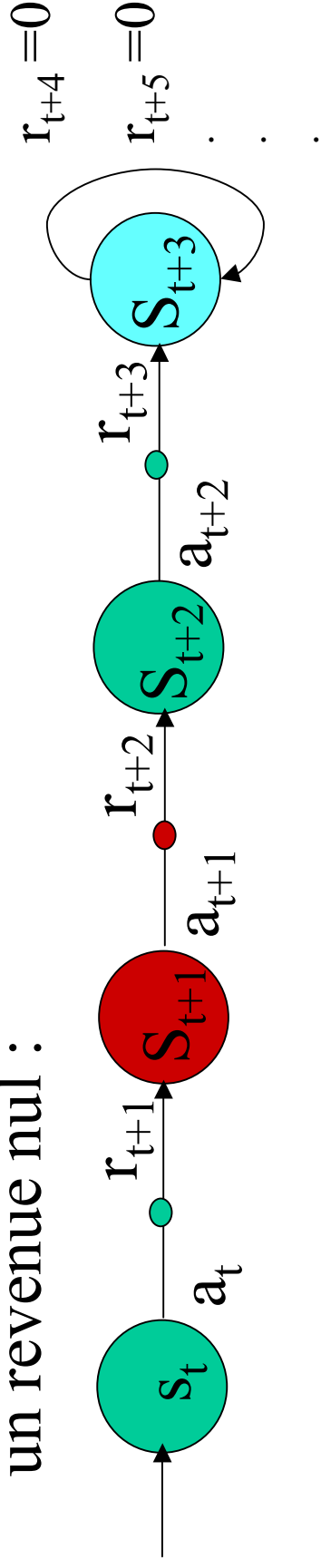
---

- Dans les tâches épisodiques, nous numérotions les pas du temps de chaque épisode en partant de zéro.
- D'habitude, nous ne devons pas faire des distinctions entre les épisodes, ainsi nous écrivons  $S_t$  à la place de  $S_{t,j}$  pour l'état au pas  $t$  de l'épisode  $j$ .

## Une notation unifiée

---

- Considérons que chaque épisode finit dans un **état absorbant (absorbing state)** qui produit toujours un **revenue nul** :



- Nous pouvons couvrir tous les cas en utilisant

$$R_t = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}, \quad (3)$$

où  $\gamma$  peut être 1 uniquement si le **revenue est nul** à l'état absorbant.

## **Propriété de Markov**

---

- ❑ Par “état” au pas  $t$ , nous désignons l’information reçue par l’agent à partir de l’environnement.
- ❑ L’état peut inclure des “sensations” immédiates comme les mesures sensorielles , des “sensations” de haut niveau, et des structures construites dans le temps à partir des séquences de sensations, par exemple : nous regardons un objet, puis on tourne notre regard ailleurs, et nous pensons dans notre esprit que cet objet est toujours là.

## Propriété de Markov (Markov Property)

---

- Idéalement, un état devrait résumer les sensations dans le passé pour retenir toute l'information "essentielle", i.e., il devrait avoir la **Propriété de Markov (Markov Property)** :

$$\Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0\} = \\ \Pr\{s_{t+1} = s', r_{t+1} = r \mid s_t, a_t\}$$

pour tout  $s', r$ , et les historiques  $s_t, a_t, r_t, s_{t-1}, a_{t-1}, \dots, r_1, s_0, a_0$ .

(4)

## **Processus de Décision de Markov (Markov Decision Processes)**

---

- **Si une tâche de RL a la propriété de Markov, elle est alors essentiellement un Processus de Décision de Markov (Markov Decision Process (MDP)).**
- **Si les ensembles d'états et des actions sont finis, alors le processus est fini (finite MDP).**

# Processus de Décision de Markov (Markov Decision Processes)

---

- Pour définir un MDP fini, nous devons donner:
- les ensembles des états et des actions
  - Une dynamique à un pas (one-step “dynamics”) définie par les probabilités de transition (transition probabilities) :

$$P_{ss'}^a = \Pr\{s_{t+1} = s' | s_t = s, a_t = a\} \quad \forall s, s' \in S, a \in A(s). \quad (5)$$

- La récompense immédiate espérée (expected immediate reward) :

$$\mathcal{R}_{ss'}^a = E\{r_{t+1} | s_t = s, a_t = a, s_{t+1} = s'\} \quad \forall s, s' \in S, a \in A(s). \quad (6)$$

Adapté de (R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction <http://www-anv.cs.umass.edu/~rich/book/the-book.html>) par Tarik AL ANI, A<sup>2</sup>SI-ESIEE-PARIS

## Un Exemple de MDP fini

---

Robot de recyclage (voir diapo. 10)

- Les décisions sont prises en se basant sur le niveau courant d'énergie : haut, bas.
- Récompense = nombre des canettes collectées.

## Robot de recyclage MDP (suite)

---

Ensemble d'état  $S = \{\text{haut, bas}\}$

Deux ensembles pour l'action en fonction du niveau d'énergie :

$A(\text{haut}) = \{\text{cherche, attend}\}$

$A(\text{bas}) = \{\text{cherche, attend, recharge}\}$

Si le niveau d'énergie est haut, alors une période de recherche active peut toujours être poursuivie sans risquer de faire chuter la batterie.

## Robot de recyclage MDP (suite)

---

Une période de recherche commençant par un niveau d'énergie **haut** maintient ce niveau haut avec une probabilité  $\alpha$  et le réduit avec une probabilité  $1-\alpha$ .

D'autre part, Une période de recherche commençant par un niveau d'énergie **bas** maintient ce niveau bas avec une probabilité  $\beta$  et le réduit avec une probabilité  $1-\beta$ . Dans le dernier cas, le robot doit être rechargé, et la batterie sera de nouveau au niveau haut.

## Robot de recyclage MDP (suite)

Chaque canette collectée par le robot est compté comme une unité de récompense, tandis qu'une récompense de -3 se produit quand le robot doit être rechargé.

Les récompenses immédiates espérés définis comme suit

$R_{\text{recherche}}$  = nombre espéré de canettes pendant la phase de recherche

$R_{\text{attend}}$  = nombre espéré de canettes pendant la phase d'attente

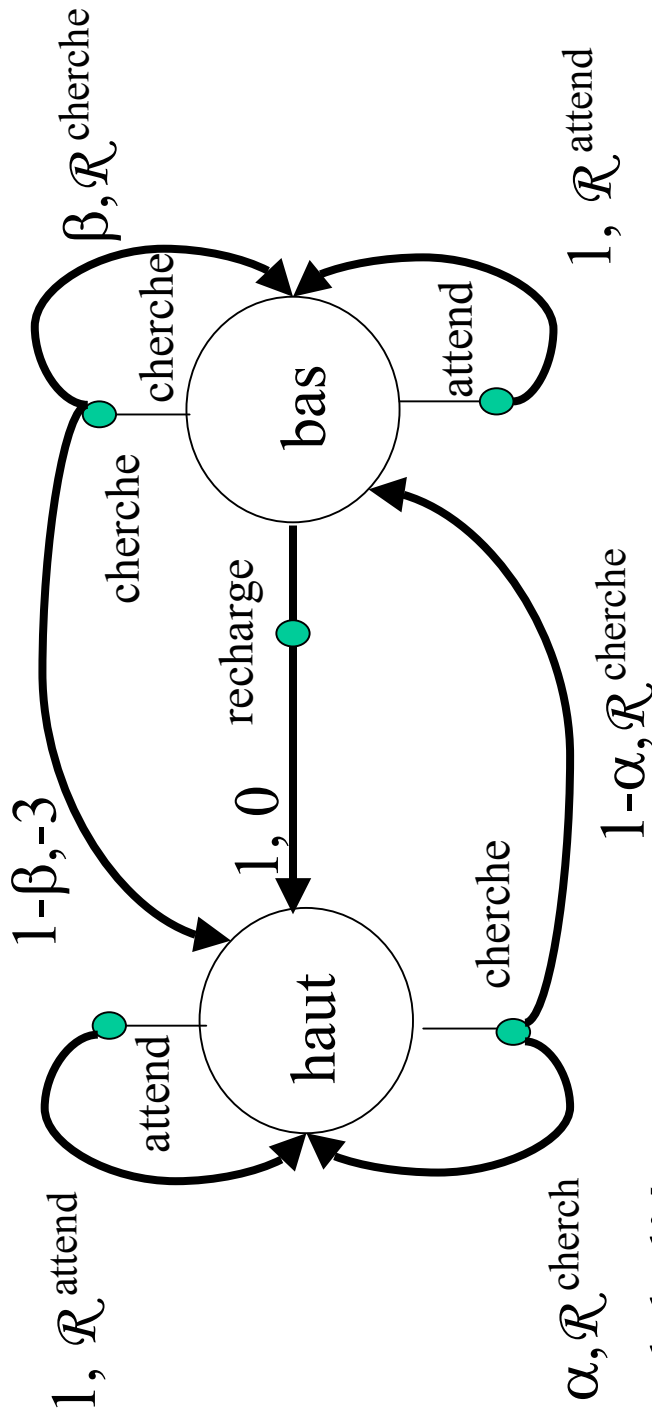
où  $R_{\text{recherche}} > R_{\text{attend}}$

Supposons, pour simplifier, qu'aucune canette peut être collectée pendant la phase de rechargement, et aucune canette peut être collectée à un pas dans lequel le niveau d'énergie de la batterie est sensiblement chutée.

## Robot de recyclage MDP (suite)

$S = S_t$	$S' = S_{t+1}$	$a = a_t$	$p^a_{ss'}$	$R^a_{ss'}$
H	H	cherche	$\alpha$	$\mathcal{R}_{cherche}$
H	L	cherche	$1-\alpha$	$\mathcal{R}_{cherche}$
L	H	cherche	$1-\beta$	-3
L	L	cherche	$\beta$	$\mathcal{R}_{cherche}$
H	H	attend	1	$\mathcal{R}_{attend}$
H	L	attend	0	$\mathcal{R}_{attend}$
L	H	attend	0	$\mathcal{R}_{attend}$
L	L	attend	1	$\mathcal{R}_{attend}$
L	H	recharge	1	0
L	L	recharge	0	0

# Robot de recyclage MDP (suite)



○ noeud de l'état

● noeud du couple état-action

→ probabilité de transition, récompense immédiate espérée associé ◆

Adapté de (R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction <http://www-anv.cs.umass.edu/~rich/book/the-book.html>) par Tarik AL ANI, A<sup>2</sup>SI-ESIEE-PARIS

## Fonctions de la Valeur (Value Functions)

---

- La valeur de l'état est le **revenue espéré** en commençant à partir de cet état; elle dépend de la **politique de l'agent** (agent's policy)

**Fonction de la Valeur de l'état pour la politique  $\pi$  :**

$$V^\pi(s) = E_\pi \left\{ R_t | s_t = s \right\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s \right\}$$

(7)

## Fonctions Valeur (Value Functions)

---

□ La valeur de choisir une action dans un état sous la politique  $\pi$  est le revenu espéré en commençant à partir de cet état, choisissant cet action et poursuivant la politique  $\pi$  :

**Fonction de la Valeur de l'action pour la politique  $\pi$  :**

$$Q^\pi(s, a) = E_\pi \left\{ R_t | s_t = s, a_t = a \right\} = E_\pi \left\{ \sum_{k=0}^{\infty} \gamma^k r_{t+k+1} | s_t = s, a_t = a \right\}$$

(8)

# Équation de Bellman pour la Politique $\pi$

---

L'idée principale :

$$\begin{aligned} R_t &= r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} \dots \\ &= r_{t+1} + \gamma(r_{t+2} + \gamma r_{t+3} + \gamma^2 r_{t+4} \dots) \\ &= r_{t+1} + \gamma R_{t+1} \end{aligned} \quad (9)$$

Alors :

$$\begin{aligned} V^\pi(s) &= E_\pi \{ R_t \mid s_t = s \} \\ &= E_\pi \{ r_{t+1} + \gamma V(s_{t+1}) \mid s_t = s \} \end{aligned} \quad (10)$$

# Équation de Bellman pour la Politique $\pi$

---

Ou, sans utiliser l'opérateur de l'espérance:

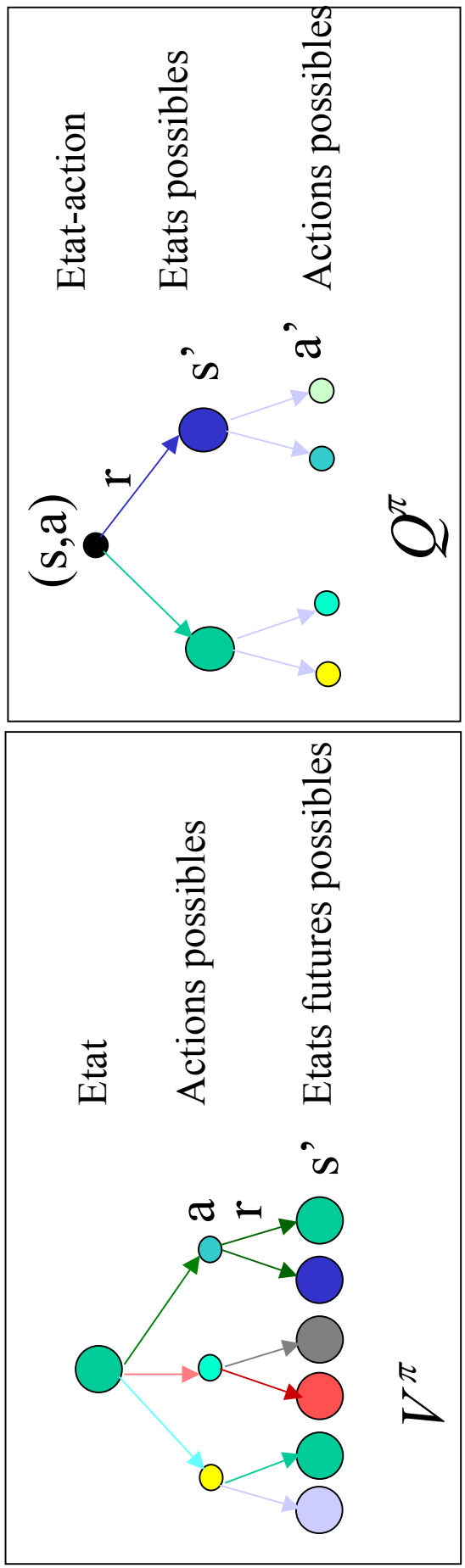
$$V^\pi(s) = \sum_a \pi(s, a) \sum_{s'} P_{ss'}^a [R_{ss'}^a + \gamma V^\pi(s')] \tag{11}$$

C'est un ensemble linéaire d'équations, une pour chaque état.

La fonction de la valeur pour  $\pi$  est son solution unique. Équation (11) est tout simplement effectuée la moyenne sur toute les possibilités, en pondérant chaque possibilité par la probabilité d'occurrence. Elle signifie que la valeur de l'état de départ  $s$  doit être égal à la valeur réduite espérée de l'état future, plus la récompense espéré sur le chemin.

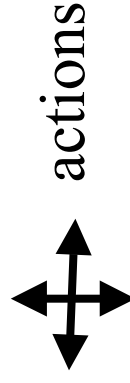
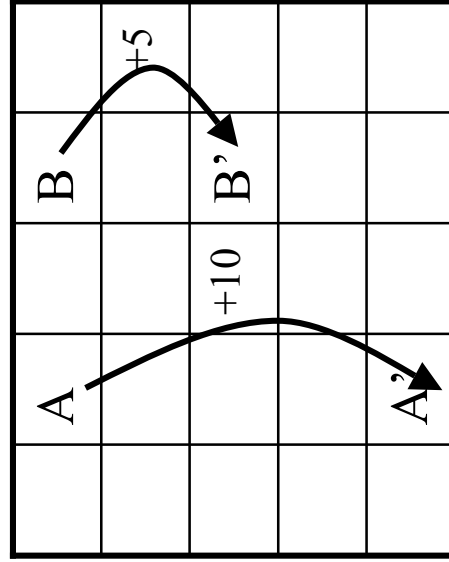
# Équation de Bellman pour la Politique $\pi$

Diagrammes de mise à jour (Backup diagrams) :



# Gridworld

- Actions : nord, sud, est, ouest; déterministe.
- Si l'action pourrait amener l'agent en dehors de la grille : pas de mouvement mais récompense = -1
- Autres actions produisent un récompense = 0, sauf les actions qui amènent l'agent en dehors des états spéciaux A et B. De A, toutes les quatre actions produisent une récompense de +10 et amène l'agent à A'. De B, toutes les quatre actions produisent une récompense de +5 et amène l'agent à B'.



Fonction de la valeur d'état  $V^\pi$  (éq. 11) pour une choix équiprobable aléatoire des états ;  $\gamma = 0.9$

3.3	8.8	4.4	5.3	1.5
1.5	3.0	2.3	1.9	0.5
0.1	0.7	0.7	0.4	-0.4
-1.0	-0.4	-0.4	-0.6	-1.2
-1.9	-1.3	-1.2	-1.4	-2.0

□ État de l'environnement

Adapté de (R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction <http://www-anw.cs.umass.edu/~rich/book/the-book.html>) par Tarik AL ANI, A<sup>2</sup>SI-ESIEE-PARIS





# Golf

---

- En dehors de green, nous ne pouvons pas arriver au trou en roulant (\*) et la valeur dans ce cas est plus grande. Si nous pouvons arriver au green à partir d'une position (état) en dehors de green, alors cet état doit avoir une valeur diminuée par 1 (-2, -3, ...)
- Pour simplifier, supposons que ne pouvons effectuer « putt » (\*\*) très précisément et d'une façon déterministe, mais à une distance limitée. Ceci nous donne le contour lisse noté -2; toutes les positions entre cette ligne et le green nécessitent exactement deux coups pour achever le trou.
- De la même façon, n'importe quelle position à l'intérieure de la distance de roulement à partir de la ligne de contour -2 doit avoir une valeur de -3, et ainsi de suite pour obtenir toutes les ligne des contours
- Rouler ne nous sort pas hors des pièges de sable, ainsi ces pièges ont une valeur  $-\infty$ . Globalement, il faut 6 coups pour aller de « té » (\*\*\*) au trou par roulement.

\*) « rouser » (putting) : Avec le fer-droit, action de faire rouler la balle vers le trou sur la pelouse d'arrivée, c'est-à-dire le vert

\*\*) Coup effectué sur le vert, qui consiste à faire rouler la balle en direction du trou à l'aide d'un fer droit.

\*\*\*) Petite pièce de bois ou de plastique ressemblant à une cheville et servant à surélever la balle pour effectuer le coup de départ.

# Fonctions des Valeurs Optimales

## Optimal Value Functions

□ Pour des MDPs finis, les politiques peuvent être **partiellement ordonnées (partially ordered)**:

$$\pi \geq \pi' \quad \text{ssi} \quad V^\pi(s) \geq V^{\pi'}(s) \quad \forall s \in S \quad (12)$$

□ Il y'a au moins une (possible plusieurs) politique qui est mieux que ou égale à toutes les autres. C'est une **politique optimale (optimal policy)  $\pi^*$** .

□ Les politiques optimales partagent la même fonction **de la valeur optimale de l'état (optimal state-value function)** :

$$V^*(s) = \max_{\pi} V^\pi(s) \quad \text{for all } s \in S \quad (13)$$

# Fonctions des Valeurs Optimales

## Optimal Value Functions

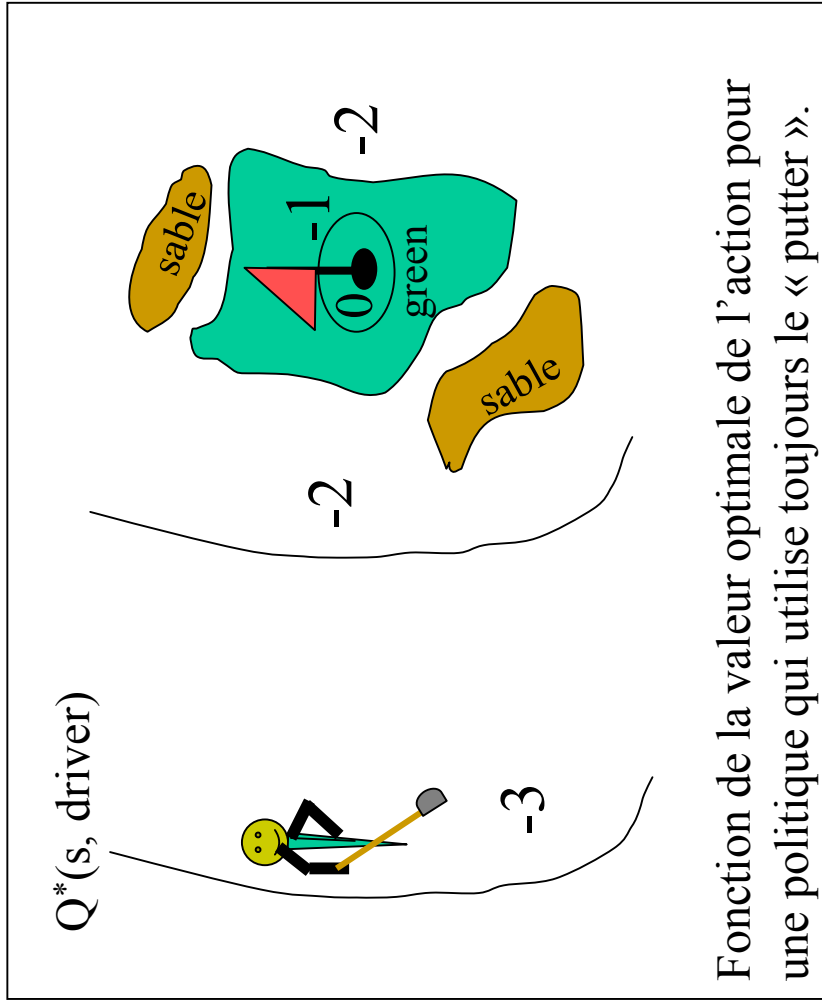
□ Les politiques optimales partagent aussi la même fonction de la valeur optimale de l'action (optimal action-value function) :

$$Q^*(s, a) = \max_{\pi} Q^{\pi}(s, a) \quad \forall s \in S \text{ et } a \in A(s) \quad (14)$$

Ceci est le revenu espéré de choisir une action  $a$  dans l'état  $s$  et ensuite poursuivre une politique optimale.

# fonction optimale de la valeur de l'action de Golf

- ▣ Nous pouvons lancer la balle plus loin avec un « driver » à la place de « putter », mais avec moins de précision.
- ▣ Les contours sont les états si nous choisissons le « driver » d'abord, ensuite choisir soit le « driver » soit le « putter » lequel est le mieux.



Fonction de la valeur optimale de l'action pour une politique qui utilise toujours le « putter ».

## fonction optimale de la valeur de l'action de Golf

---

▣ Nous pouvons achever le trou en un seul coup en utilisant le « driver » uniquement si nous sommes déjà très prêt; ainsi le contour  $-1$  pour  $Q^*(s, \text{driver})$  couvre une petite portion de « green ». Cependant, si nous avons deux coups, alors nous pouvons achever le trou à partir d'une distance plus grande (voir contour  $-2$ ). Dans ce cas, nous n'avons besoin de diriger la balle toujours de l'intérieur de petit contour  $-1$ , mais de n'importe où sur le « green »; là où nous pouvons utiliser le « putter ». La fonction de la valeur optimale de l'action donne les valeurs après commuter vers une première action particulière, dans ce cas, vers le « driver », mais ensuite en utilisant n'importe quelles actions qui sont les meilleures. Le contour  $-3$  reste plus loin et inclut le « té » de départ. A partir de « té », la meilleure séquence d'actions est deux « drive » et un « putt », amenant la balle au trou en trois coups.



# Equation d'optimalité pour $V^*$ de Bellman

## (Bellman Optimality Equation for $V^*$ )

La valeur de l'état sous une politique optimale doit être égale au revenu espéré pour la meilleure action à partir de cet état :

$$\begin{aligned} V^*(s) &= \max_{a \in A(s)} Q^{\pi^*}(s, a) \\ &= \max_{a \in A(s)} E \left\{ r_{t+1} + \gamma V^*(s_{t+1}) \mid s_t = s, a_t = a \right\} \\ &= \max_{a \in A(s)} \sum_{s'} P_{ss'}^a \left[ R_{ss'}^a + \gamma V^*(s') \right] \end{aligned} \quad (15)$$

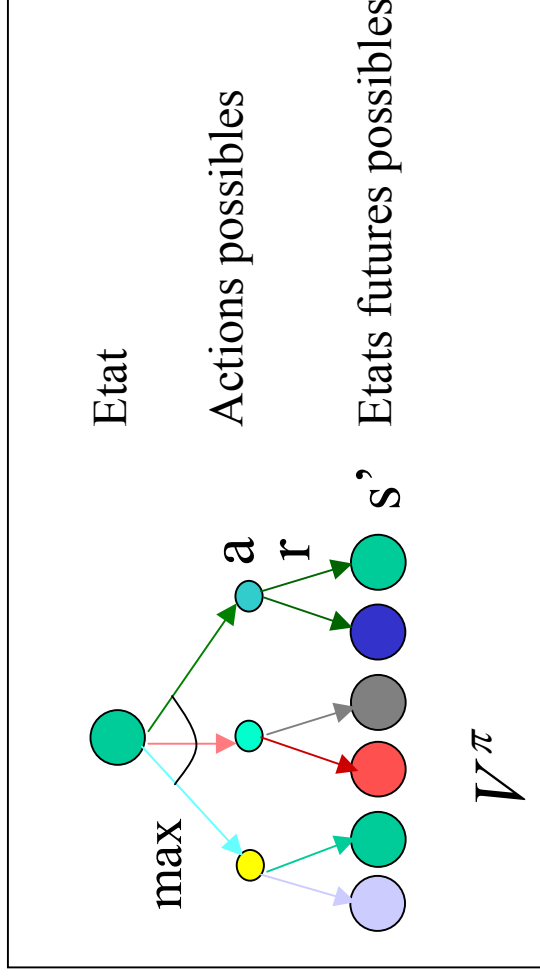
Backup diagram:

$V^*$  est l'unique solution de ce système d'équations non linéaires.

# Equation d'optimalité pour $V^*$ de Bellman (Bellman Optimality Equation for $V^*$ )

---

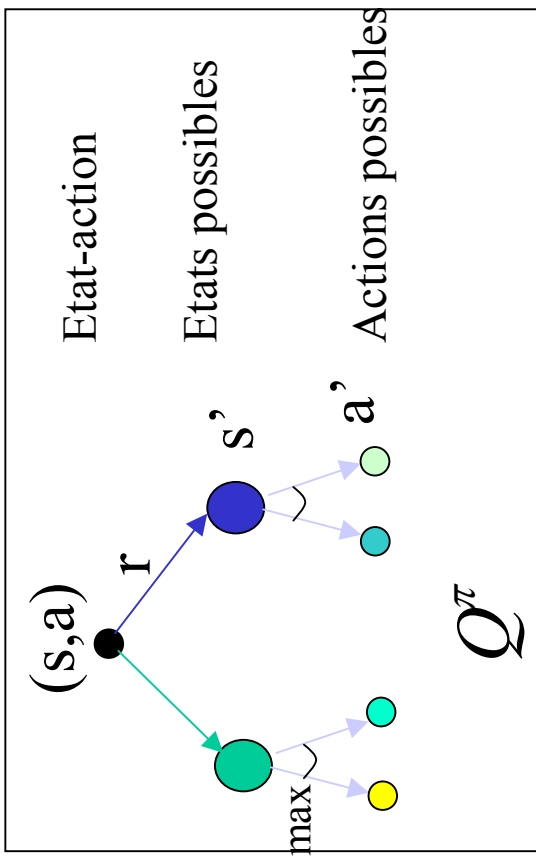
Backup diagram:



# Équation d'optimalité de Bellman pour $Q^*$

$$\begin{aligned} Q^*(s, a) &= E \left\{ r_{t+1} + \gamma \max_{a'} Q^*(s_{t+1}, a') \mid s_t = s, a_t = a \right\} \\ &= \sum_{s'} P_{ss'}^a \left[ R_{ss'}^a + \gamma \max_{a'} Q^*(s', a') \right] \end{aligned} \quad (15)$$

Backup diagram:



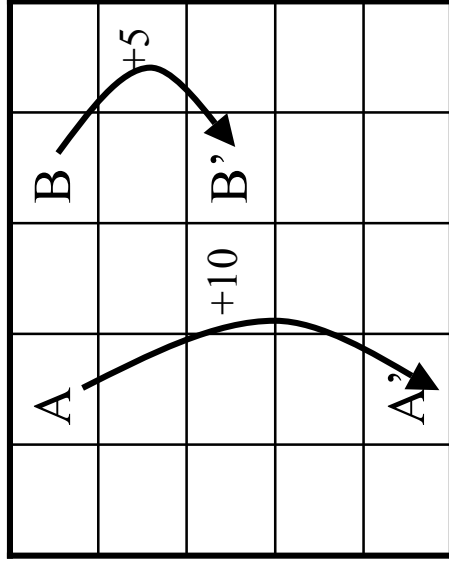
$Q^*$  est l'unique solution de ce système d'équations non linéaires.

# Pourquoi les fonctions optimales de la valeur d'état sont utiles

N'importe quelle politique qui est gourmande par rapport à  $V^*$  est une politique optimale.

Alors, étant donné  $V^*$ , une recherche un pas vers l'avant produit des actions optimales à long terme.

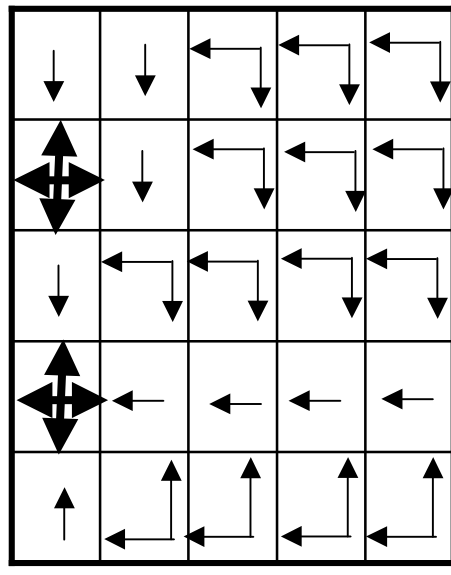
Exemple : gridworld: en (c) les politiques optimales (une flèche correspond à une action optimale.



(a) gridworld

22.0	24.4	22.0	19.4	17.5
19.8	22.0	19.8	17.8	16.0
17.8	19.8	17.8	16.0	14.4
16.0	17.8	16.0	14.4	13.0
14.4	16.0	14.4	13.0	11.7

(b)  $V^*$



(c)  $\pi^*$

Adapté de (R. S. Sutton and A. G. Barto: Reinforcement Learning: An Introduction <http://www-anw.cs.umass.edu/~rich/book/the-book.html>) par Tarik AL ANI, A<sup>2</sup>SI-ESIEE-PARIS

# Concernant les fonctions optimales de la valeur de l'action?

---

Étant donné  $Q^*$ , l'agent n'a même pas besoin d'effectuer une recherche un pas vers l'avant :

$$\pi^*(s) = \arg \max_{a \in A(s)} Q^*(s, a)$$

(16)

# Résoudre l'Équation d'Optimalité de Bellman

---

- Trouver une politique optimale en résolvant l'Equation d'Optimalité de Bellman nécessite les démarches suivantes:
  - Connaître précisément la dynamique de l'environnement ;
  - prévoir assez d'espace et de temps pour le calcul;
  - respecter la propriété de Markov.
- Combien d'espace et de temps?
  - polynomial en nombre d'états (via les méthodes de Programmation Dynamique; Chapitre 4),
  - Mais, le nombre d'états est souvent élevé (e.g., backgammon  $\sim 10^{20}$  states).

# Résoudre l'Equation d'Optimalité de Bellman

---

- ❑ Nous devons effectuer des approximations.
- ❑ Beaucoup de méthodes de RL peuvent être considérées comme une approximation pour résoudre l'équation d'optimalité de Bellman.