

# Apprentissage par Renforcement

JÉRÉMIE MARY

équipe TAO  
LRI

27 février 2006

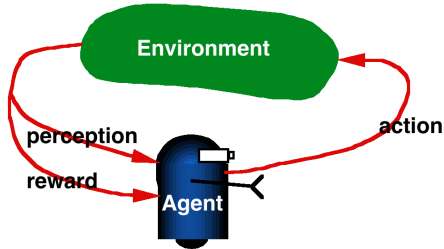


# Plan

- 1 Introduction
- 2 Apprendre et utiliser un MDP
- 3 Eligibility Traces
- 4 Au delà des MDPs
- 5 Références



# Apprentissage par Interaction



- 1 Agent complet,
- 2 Influencé par le temps,
- 3 L'agent apprend et planifie en permanence,
- 4 Le but est **d'influer** sur l'environnement,
- 5 Le **monde est stochastique** et bruité.



# Formalisme Usuel

$$o_1 a_1 r_2, o_2 a_2 r_3 \dots \underbrace{o_i a_i r_{i+1}}_{\text{Expérience unitaire}}, \dots$$

- 1 L'agent agit de manière à **maximiser l'espérance du cumul des récompenses** pour un horizon temporel donné.
- 2 Les **Observations** peuvent être des vecteurs ou des structures,
- 3 Les **Actions** peuvent être multi-dimensionnelles,
- 4 Les **Récompenses** sont des réels,
- 5 L'agent n'a qu'une connaissance partielle de son environnement.

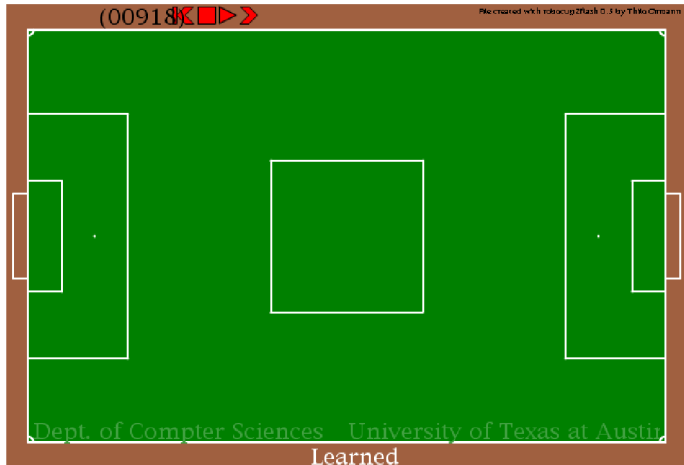


# Google Think of RL

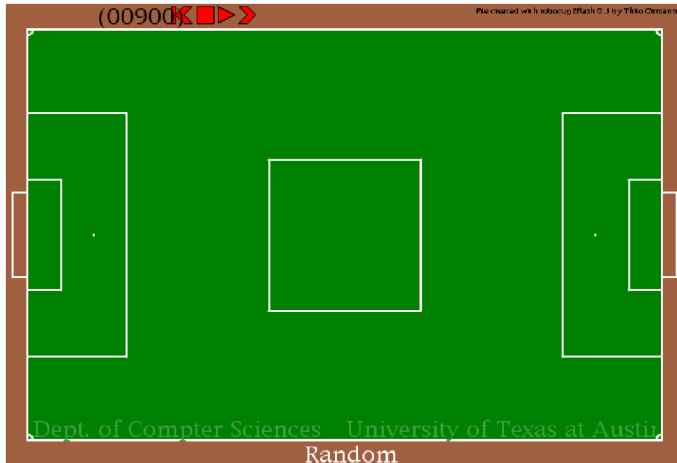
- 1 Différences Temporelles (mise à jour d'une prédiction basée sur une autre prédiction),
- 2 Eligibility traces,
- 3 Apprentissage off-line,
- 4 Approximation de fonctions,
- 5 Apprentissage hiérarchique,
- 6 MDP/POMDP et au delà...



# Stone & Sutton - Learned

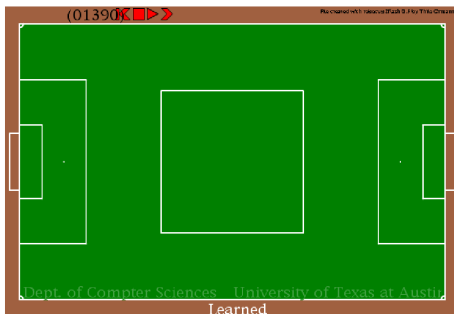


# Stone & Sutton - Random

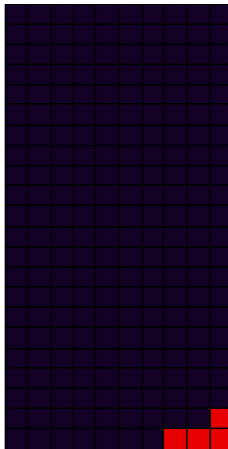


# Conservation de Balle

- 1 Match 4 vs 3 :
  - Apprentissage par renforcement conservation pendant **10.2 secondes**
  - Passes aléatoires conservation pendant **6.3 secondes**
- 2 Match 5 vs 4 :
  - Apprentissage par renforcement conservation pendant **12.3 secondes**
  - Passes aléatoires conservation pendant **8.3 secondes**



# Tetris Demo [Bagnell & Schneider]



# Place de l'Apprentissage par Renforcement

- 1 A la frontière entre
  - Le contrôle optimal
  - La recherche opérationnelle
  - Psychologie - Neurosciences
- 2 Différent de l'apprentissage Supervisé et Non-Supervisé :
  - on veut prendre des **décisions séquentielles**,
  - pour des **récompenses non-immédiates**.
- 3 Applications :
  - Robotique (*Navigation, Robosoccer, marche, jonglage*),
  - Contrôle (*processus industriels, appels téléphoniques, allocations de ressources multimédias, hélicoptères, gestion ascenseurs*),
  - Jeux (*Backgammon, Échecs, Othello, Tetris, Backgammon, ...*),
  - Recherche opérationnelle (*Stockage, transport, emplois du temps*),
  - Modélisation de systèmes biologiques, IHM, ...



# Markov Decision Processes (MDPs)

$o_1 a_1 r_2, o_2 a_2 r_3 \dots o_i a_i r_{i+1}, \dots$

## Hypothèse de Markov

- $P(o_{t+1} | o_t, a_t, o_{t-1}, a_{t-1}, \dots, o_1, a_1) = P(o_{t+1} | o_t, a_t)$
- $P(r_{t+1} | o_t, a_t, o_{t-1}, a_{t-1}, \dots, o_1, a_1) = P(r_{t+1} | r_t, a_t)$
- Probabilités de transition :  $P_{ss'}^a = P(s_{t+1} = s' | s_t = s, a_t = a)$
- Fonction de Récompense :  $R_{ss'}^a = E(r_{t+1} | s_{t+1}, s_t = s, a_t = a)$





# Notations

- $S$  espace des états.
- $A$  espace de actions.
- $P$  probabilités de transitions  $P(i|j, a)$
- $R$  Fonction de récompense  $R(i)$  (ou  $R(i, a)$ )
- $\pi$  politique  $S \rightarrow A$
- $V^\pi(i)$  gain espéré de la politique  $\pi$  en partant de l'état  $i$ .

$$V^{\pi}(i) = \underbrace{E_{\pi}(r_0 + \gamma r_1 + \gamma^2 r_2 + \dots | s_0 = i)}_{\text{renforcement escompté (horizon infini) } 0 \leq \gamma < 1}$$

Autre possibilité : renforcements moyens :

$$V^{\pi}(i) = \lim_{T \rightarrow \infty} \frac{1}{T} E_{\pi}(r_0 + r_1 + \dots | s_0 = i)$$



# Notations

## Théorème

Dans les MDP il **existe toujours une politique déterministe** qui maximise simultanément la valeur de chaque état.

$$\pi^*(i) = \arg \max_{\pi} V^{\pi}(i)$$

$$V^*(i) = \max_{\pi} V^{\pi}(i)$$



# Équations d'Optimalité de Bellman

- Évaluation de la politique :

$$V^{p^i}(i) = E_{\pi}(r_0 + \gamma r_1 + \gamma^2 r_2 + \dots | s_0 = i)$$

- Par l'hypothèse de Markov :

$$\forall s \in \mathcal{S}, V^{\pi}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s)) V^{\pi}(s')$$

- De même on définit :

$$Q^{\pi}(s, a) = E_{\pi}(r_0 + \gamma r_1 + \gamma^2 r_2 \dots | s_0 = s, a_0 = a)$$

- et par markov :

$$\forall s \in \mathcal{S}, a \in \mathcal{A} Q^{\pi}(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s)) Q^{\pi}(s', \pi(s'))$$



# Contrôle Optimal

$$\forall s \in \mathcal{S}, V^*(s) = \max_{a \in \mathcal{A}} \left[ R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) V^*(s') \right]$$

$$\forall s \in \mathcal{S}, \pi^*(s) = \arg \max_{a \in \mathcal{A}} \left[ R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) V^*(s') \right]$$

$$\forall s \in \mathcal{S}, Q^*(s, a) = R(s, a) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, a) \max_{b \in \mathcal{A}} Q^*(s', b)$$

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

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



# Plannification

Si l'on dispose d'un modèle exact (i.e. fonction de récompense et matrice de transitions) et d'une politique  $\pi$ .

## Itération de valeur pour V

1 Initialiser  $V_0$  arbitrairement

2 Répéter

- $\forall s \in \mathcal{S}, V_{k+1}(s) = R(s, \pi(s)) + \gamma \sum_{s' \in \mathcal{S}} P(s'|s, \pi(s)) \gamma V_k(s')$

Tant que  $\max_{s \in \mathcal{S}} |V_{k+1}(s) - V_k(s)| \leq \epsilon$

3 Renvoyer  $V_k$



# Plannification

Si l'on dispose d'un modèle exact (i.e. fonction de récompense et matrice de transitions) et d'une politique  $\pi$ .

## Itération de valeur pour Q

1 Initialiser  $Q_0$  arbitrairement

2 Répéter

- $\forall s \in S, a \in A \quad Q_{k+1}(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) Q_k(s', \pi(s'))$

Tant que  $\max_{s \in S, a \in A} |Q_{k+1}(s, a) - Q_k(s, a)| \leq \epsilon$

3 Renvoyer  $Q_k$



# Contrôle Optimal

Étant donné un modèle exact (i.e. fonction de récompense et matrice de transitions).

## Itération de valeur pour la politique optimale avec $V$ (resp. $Q$ )

- 1 Initialiser  $V_0$  (resp.  $Q_0$ ) arbitrairement
- 2 Répéter

- $\forall s \in S, V_{k+1}(s) = \max_{a \in A} \left[ R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) V_k(s') \right]$

(resp.

$$\forall s \in S, a \in A Q_{k+1}(s, a) = R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) \max_{b \in A} Q_k(s', b))$$

Tant que  $\max_{s \in S} |V_{k+1}(s) - V_k(s)| \leq \varepsilon$

(resp.  $\max_{s \in S, a \in A} |Q_{k+1}(s, a) - Q_k(s, a)| \leq \varepsilon$ )

- 3 Renvoyer  $V_k$  (resp.  $Q_k$ )



# Preuve de Convergence

- Soit  $\Delta_k = \|Q^* - Q_k\|_\infty$

- 

$$\begin{aligned} Q_{k+1}(s, a) &= R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) \max_{b \in A} Q_k(s', b) \\ &\leq R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) \max_{b \in A} [Q^*(s', b) + \Delta_k] \\ &= \left[ R(s, a) + \gamma \sum_{s' \in S} P(s'|s, a) \max_{b \in A} Q^*(s', b) \right] + \gamma \Delta_k \\ &= Q^*(s, a) + \gamma \Delta_k \end{aligned}$$

**L'algorithme est contractant !**  
**(l'écart à  $Q^*$  est multiplié par  $\gamma$  à chaque itération)**





# Méthodes Indirectes

- 1 On évalue d'abord les probabilités conditionnelles (grâce aux données)

$$\hat{P}(j|i, a) = \frac{\#j \leftarrow i, a}{\#j \leftarrow i, .}$$

- 2 On calcule la politique optimale par rapport au modèle obtenu
- 3 Dilemme **Exploration-Exploitation**

**Converge asymptotiquement vers la politique optimale** si toutes les paires états-actions sont visitées infiniment souvent.



# Méthode Directe : Q-learning [Watkins'88]

$$s_0 a_0 r_0 s_1 a_1 r_1 s_2 a_2 r_2 \dots s_k a_k r_k$$

- 1 L'unité d'expérience est  $\langle s_k a_k r_k s_{k+1} \rangle$
- 2 La mise à jour est :

$$Q_{new}(s_k, a_k) = (1 - \alpha)Q_{old}(s_k, a_k) + \alpha \left[ r_k + \gamma \max_{b \in A} Q_{old}(s_{k+1}, b) \right]$$

- 3 Seuls sont mis à jour les états visités et on conserve une grande table des valeurs de  $Q$ .



# Preuve de convergence

## Hypothèses

- Chaque paire état-action est mise à jour infiniment souvent.
- $\sum \alpha = \infty$
- $\sum \alpha^2$  fini.

$$Q_{new}(s_k, a_k) = (1 - \alpha)Q_{old}(s_k, a_k) + \alpha \left[ r_k + \gamma \max_{b \in A} Q_{old}(s_{k+1}, b) \right]$$

On remarque que :

$$E(r_k + \gamma \max_{b \in A} Q_{old}(s_{k+1}, b)) = R(s_k) + \gamma \left[ \sum_{j \in S} P(j|s_k, a_k) \max_{b \in A} Q_{old}(j, b) \right]$$

Le Q-learning est donc une approximation stochastique de l'itération de la valeur Q :

- Q-value iteration  $Q_{k+1} = T(Q_k)$
- Q-learning :  $(1 - \alpha)Q_k + \alpha(T(Q_k) + \eta_k)$  où  $\eta_k$  est un bruit de moyenne nulle



## En résumé

- 1 Q-learning est le premier **algorithme direct de contrôle optimal adaptatif**
- 2 Les représentations sont plus petites que les modèles
- 3 Focalise automatiquement son attention là où c'est utile
- 4 Ne résouds pas le dilemme exploration-exploitation (en particulier  $\epsilon$ -glouton, problèmes d'initialisation. . . )



# Techniques à la Monte-Carlo

On veut évaluer  $V^\pi(s)$

- 1 On démarre dans l'état  $s$  et on **exécute la politique pendant longtemps** afin d'estimer le retour empirique
- 2 On **répète plusieurs fois** l'opération et on moyenne sur les exécutions
- 3 Combien de trajectoires ?
- 4 C'est un estimateur sans biais dont la variance décroît avec le nombre de trajectoires



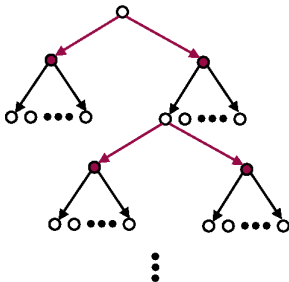
# Application de la Méthode Directe



# Application de la Méthode In-directe [Ng]



# Échantillonnage clairsemé (Sparse Sampling [Kearns, Mansour, Ng])



- 1 Utilise un modèle génératif pour obtenir  $n$  arbres avec  $m$  exemples pour chaque action dans chaque état généré.
- 2 Action presque **optimale en temps indépendant de la taille de l'espace d'états.**
- 3 Exponentiel en l'horizon...



# Classification pour le RL [Langford]

- 1 On utilise un Sparse Sampling pour obtenir un ensemble d'exemples pour lesquels on dispose d'une action quasi-optimale pour un sous échantillon des états
- 2 On passe un algorithme de classification sur ces données.
- 3 Puissants algorithmes pour le RL (et résultats théoriques)



# Eligibility Traces

## Le probleme

Étant donnée une politique  $\pi$  éventuellement stochastique, on veut estimer :

$$V^\pi(i) = E_\pi(r_0 + \gamma r_1 + \gamma^2 r_2 + \dots | s_0 = i)$$

A partir de la répétition de trajectoires générées en suivant la politique  $\pi$  à partir de l'état  $i$

Une trajectoire :

$$r_0 \ r_1 \ r_2 \ r_3 \ \dots \ r_k \ r_{k+1} \ \dots$$



# TD( $\lambda$ )

$r_0 \ r_1 \ r_2 \ r_3 \ \dots \ r_k \ r_{k+1} \ \dots$

0-Step ( $e_0$ ) :  $r_0 + \gamma V(s_1)$

- $V_{new}(s_0) = V_{old}(s_0) + \alpha \underbrace{[r_0 + \gamma V_{old}(s_1) - V_{old}(s_0)]}_{\text{différence temporelle}}$

- $V_{new}(s_0) = V_{old}(s_0) + \alpha [e_0 - V_{old}(s_0)]$



# TD( $\lambda$ )

$$r_0 \ r_1 \ r_2 \ r_3 \ \dots \ r_k \ r_{k+1} \ \dots$$

1-Step ( $e_1$ ) :  $r_0 + \gamma V(s_1) + \gamma^2 V(s_2)$

- $V_{new}(s_0) = V_{old}(s_0) + \alpha [e_1 - V_{old}(s_0)]$
- $V_{new}(s_0) = V_{old}(s_0) + \alpha [r_0 + \gamma V(s_1) + \gamma^2 V(s_2) - V_{old}(s_0)]$



# TD( $\lambda$ )

On peut ainsi définir une infinité de méthodes de mise à jour. On en définit une famille par combinaison linéaire :

$$V_{new}(s_0) = V_{old}(s_0) + \alpha \left[ \sum_k w_k e_k - V_{old}(s_0) \right]$$

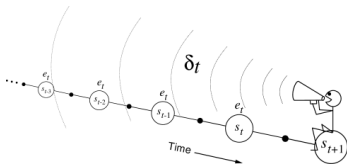
Traditionnellement on prend

$$w_k = (1 - \lambda)\lambda^k$$

$0 \leq \lambda \leq 1$  permet d'interpoler entre le 1-step (biaisé mais avec faible variance) et Monte-Carlo (non biaisé mais avec une plus grande variance).



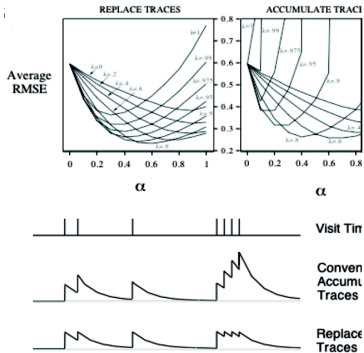
# TD( $\lambda$ )



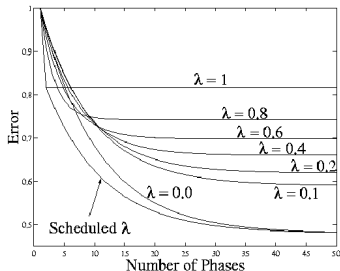
On prend un bloc élémentaire

$\langle s, a, r, s' \rangle$

- $\delta \leftarrow r + \gamma V(s') - V(s)$
- $e(s) \leftarrow e(s) + 1$
- $\forall i \in S$ :
  - $V(i) \leftarrow V(i) + \alpha \delta e(i)$
  - $e(i) = \gamma \lambda e(i)$



# Compromis Biais-Variance



## Résultat théorique

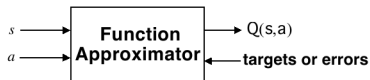
$$erreur_t \leq a_\lambda \frac{1 - b_\lambda^t}{1 - b_\lambda} + b_\lambda^t$$

- $t \rightarrow \infty$ , l'erreur converge vers  $\frac{a_\lambda}{1 - b_\lambda}$  (croissant en  $\lambda$ )
- La vitesse de convergence est  $b_\lambda^t$  (exponentielle,  $b_\lambda$  est décroissant en  $\lambda$ ).

On a intérêt a débiter avec  $\lambda$  grand et a le diminuer ensuite



# Approximation par des fonctions



On peut utiliser :

- une table,
- Des réseaux de neurones apprenant par backprop,
- réseaux RBF,
- des arbres de décision,
- ...

## Résultats

- Possibilité de recoder les données (Sparse coding)
- Approximateur linéaire peut diverger. Non linéaire peu étudié (knn prouvé non divergent)
- **Semble vaincre la malédiction de la dimension,**
- Ca marche mais pas (encore) de killer algorithm



# Demo : Cellphone assignment



# Au delà des MDPs

- 1 Repenser les Actions, États et Récompenses
- 2 Options à la place des Actions
- 3 POMDPs



# Repenser les actions : Hierarchical RL

## Objectif

On veut introduire l'abstraction en IA :

- Sur les états
- Sur le temps : échelles de temps



# Options

Il s'agit d'une généralisation des actions.

## Définition

Une option est un triplet  $o = \langle I, \pi, \beta \rangle$

- $I \subseteq \mathcal{S}$  est l'ensemble des états dans lesquels  $o$  peut-être débutée,
- $\pi : \mathcal{S} \times \mathcal{A} \rightarrow [0, 1]$  est la stratégie suivie pendant l'option  $o$ ,
- $\beta : \mathcal{S} \rightarrow [0, 1]$  est la probabilité de terminer deans chaque état.

Une option est supposée être **appelée et rendre la main**. Exemple charment :

$I$  : Tous les états dans lesquels le chargement est en vue

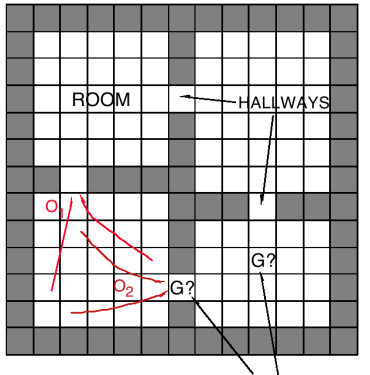
$\pi$  : Contôlleur bricolé à la main

$\beta$  : terminée quand chargé ou chargement n'est plus visible.

**Une option peut prendre un nombre variable de pas de temps.**



# Exemple

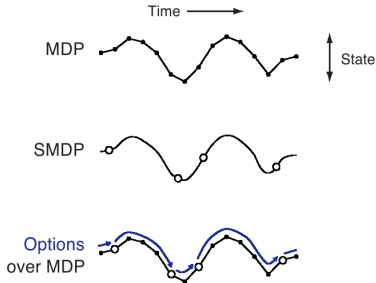


Goal states are given  
a terminal value of 1

- 4 pieces
- 4 couloirs
- 4 actions primitives qui échouent 33% du temps
- 8 options
- Étant donné un objectif, trouver le plus court chemin
- $\gamma = 0.9$ , toutes les autres récompenses à 0



# SMDP



- 1 Temps discret, écarts homogènes
- 2 Temps continu, évènements discrets
- 3 Temps discret, écarts non-homogènes

**MDP+Options = SMDP (on recycle tout)**



# Généralisation de "Iterative value"

- 1 Initialiser avec  $\forall s, V_0(s) = 0$
- 2 Itération :

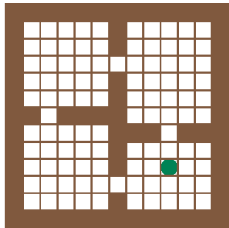
$$\forall s \in S \quad V_{k+1}(s) = \max_{o \in O} \left[ r_s^o + \sum_{s' \in S} P(s|o, s') V_k(s') \right]$$

L'algorithme converge vers  $V_O^*$ .

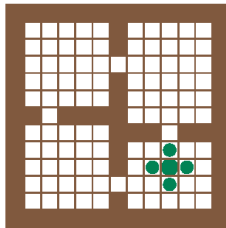
- Si  $O = A$  l'algorithme est équivalent à value iteration
- Si  $A \subseteq O$  alors  $V_O^* = V^*$



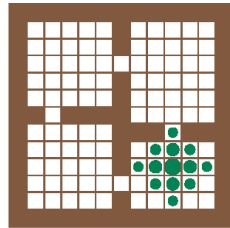
# Exemple



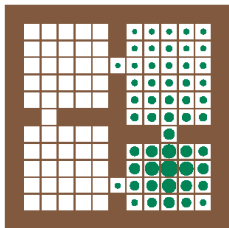
Initial values



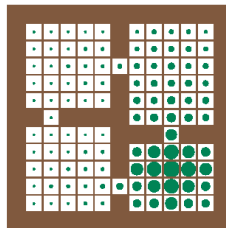
Iteration #1



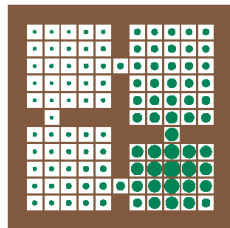
Iteration #2



Iteration #3



Iteration #4



Iteration #5



# Avantages

- 1 La vue SMDP permet de calculer les fonctions et les politiques plus rapidement sur les options.
- 2 La vue MDP permet de savoir quand déclencher une option pour un but donné.
- 3 On découpe le problèmes en sous buts (reste à apprendre la stratégie des options).
- 4 Les options sont améliorées.



# Amélioration de la terminaison

## Idée

Parfois, c'est une bonne idée de ne pas laisser une option aller jusqu'à son terme.

## Théorème

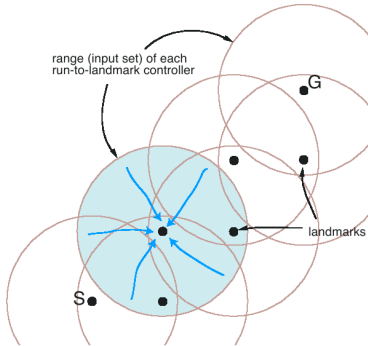
Pour toutes les politiques sur les option  $\mu : S \times O \rightarrow [0, 1]$ , supposons que l'on interrompe l'option, une ou plusieurs fois si :

$$Q^\mu(s, o) < Q^\mu(s, \mu(s)) \text{ ou } s \text{ est l'état courant}$$

On obtient alors  $\mu' > \mu$  (la récompense est meilleure partout)



# Exemple

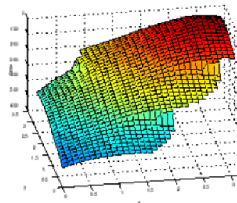
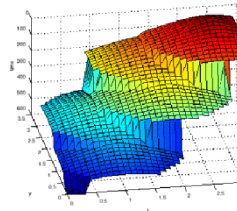
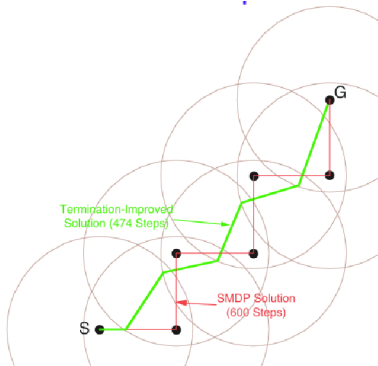


- But : aller de S à G,
- 4 actions élémentaires de petits déplacements,
- 7 contrôleurs permettant de se rendre au centre d'un cercle.

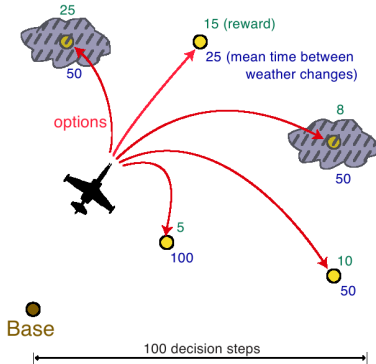
Apprendre au niveau primitif est impossible (coût trop élevé), on a besoin des contrôleurs



# Exemple



# Illustration : Planification de mission de reconnaissance



- But : survoler les sites les plus intéressants et rentrer à la base,
- La météo affecte les observations,
- Carburant limité,
- Échelles temporelles :
  - Actions : dans quelle direction voler
  - Options : Vers quel site aller
- Les Options réduisent l'espace de recherche :
  - Pour les étapes de  $\sim 600$  à  $\sim 6$
  - Pour les états possibles de  $\sim 10^1$  à  $\sim 10^6$



# Illustration : Planification de mission de reconnaissance

## SMDP

- Les actions sont suivies jusqu'au bout
- On cherche le SMDP optimal

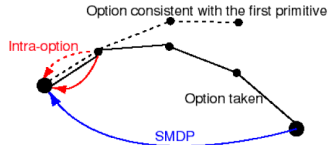
## SMDP réévalué

- La planification est faite comme si les actions allaient être suivies jusqu'à la fin,
- mais à tous les tours choisit une nouvelle option.



# Intra-Option Learning

Normalement pendant l'apprentissage du SMDP, on exécute l'option jusqu'à la fin et on met à jour uniquement l'option choisie.



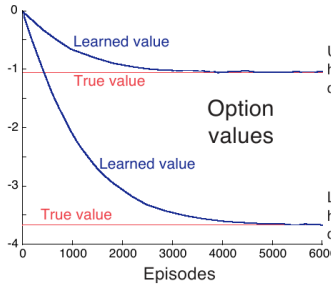
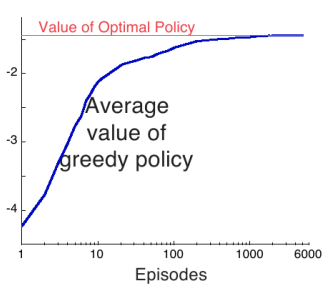
A

près chaque choix primitive, on met à jour toutes les options qui auraient pu choisir cette action

Sous les même hypothèses que le 1-Step Q-learning on a la convergence vers les valeurs correctes



# Exemple Intra-Options

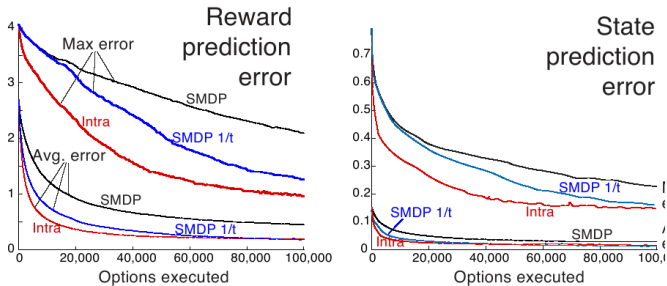


Random start, goal in right hallway, random actions

La technique intra-options apprend la bonne valeur sans prendre les options !



# Exemple Intra-Options

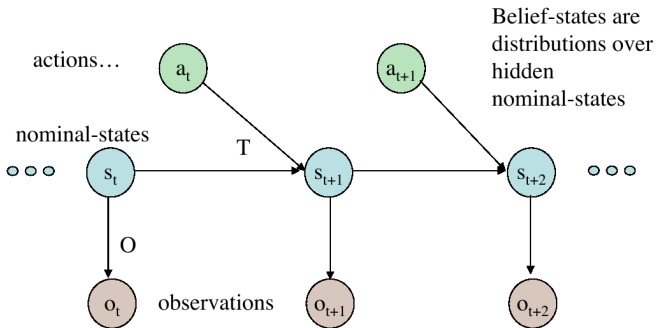


La technique intra-options apprend plus vite qu'un SMDP.



# Repenser l'État

Utiliser des POMDPs au lieu des MDPs :



# Repenser les récompenses

Je sais pas j'y connais rien (mais ca se fait)



# Mythes du Reinforcement Learning

- 1 Le RL c'est du TD ou peut-être du Q-learning,
- 2 Dans le RL, il n'y a pas de modèle,
- 3 Le RL fait du look-up table
- 4 Le RL c'est leeeeeent
- 5 Le RL ne marche pas bien avec l'approximation de fonctions
- 6 Le RL a du mal avec les POMDPs
- 7 Le RL c'est d'apprendre des stratégies optimales



# Références

- 1 **Reinforcement Learning : A Tutorial** - NIPS05 - Satinder Singh
- 2 **Thèse de Doctorat : Exploration guidée et induction de comportements génériques en apprentissage par renforcement**  
-IRISA- Pascal Garcia
- 3 **Reinforcement Learning : An Introduction** - Richard S. Sutton and Andrew G. Barto

