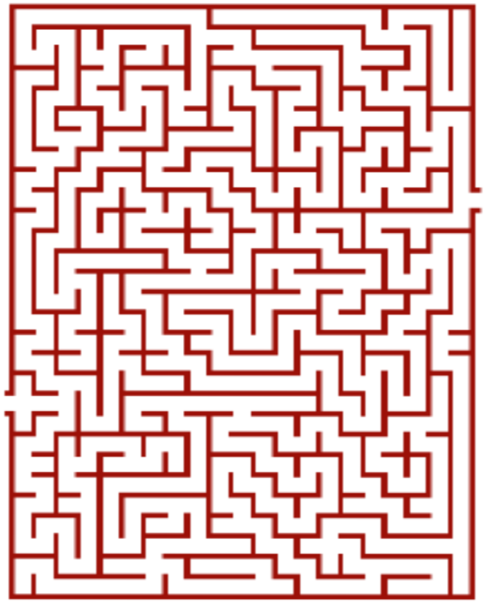
Processus de Décision Markoviens

Hélène Fargier / (Philippe Leray)

fargier@irit.fr

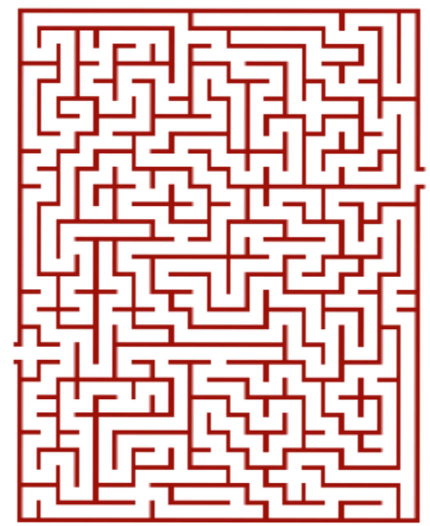
Introduction

- Un problème de décision est souvent dynamique...
- Comment prendre en compte l'aspect séquentiel ?
- Ex : robot dans un labyrinthe



Notations

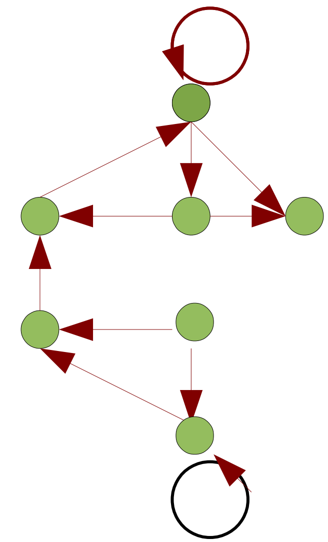
- **A l'instant t,**
  - **l'environnement est dans l'état** s_t
  - **l'agent décide d'une action** a_t
  - **pour cette action, l'agent recevra une récompense immédiate** r_{t+1}
    (cout de l'action + reward ou cout à etre dans l'etat suivant)
  - **l'action fait passer l'environnement dans l'état** s_{t+1}



A l'instant t,

- ***st = position du robot dans le labyrinthe***
- ***at = aller à l'Ouest / Est / Nord / Sud***
- ***rt = -1 si la case n'est pas la sortie, +1 sinon***

Processus de Décision Markovien (MDP)



- **Soient**
  - **un ensemble fini d'états S**
  - **un ensemble fini d'actions A**

- **A chaque instant t (discret), l'agent**
  - **observe un état** $s \in S$
  - **choisit une action** $a \in A$

- **Répercussions :**
  - **l'environnement passe dans l'état (cas deterministe)** $s' = T(s; a)$
  - **l'agent recoit la récompense** $r' = r(s; a)$ $(= r(s') - c(s,a))$

- **Hypothèse de Markov : T(*s*; a) et r(s; a) ne dépendent que de l'état et de l'action choisie**

Objectif (cas déterministe)

Cas déterministe : appliquée dans un état donné, une action ne peut conduire qu'à un seul nouvel état

- **Choisir une séquence d'actions τ = (s0,a0,s1,a1,s2,a2, …) - un plan, ou « trajectoire »- qui va maximiser une fonction fixée**
- **A l'instant t, essayer de recevoir de futures récompenses r_{t+i} les meilleures possibles**
- **Récompense cumulée (à horizon fini):** $R(t) = \sum_{i=0,h} r_{t+i+1}$
- **Récompense cumulée (à horizon infini) avec un terme d'oubli**

$$R(t) = \sum_{i=0,\infty} \gamma_i \,.\, r_{t+i+1}$$

γi représente la valeur t=i d'une unité de récompense reçue t=i+1

Representer les actions

- **Actions déterministes:**

$$T: S \times A \rightarrow S \;\;;$$

Pour tout état et toute action T spécifiait un nouvel etat

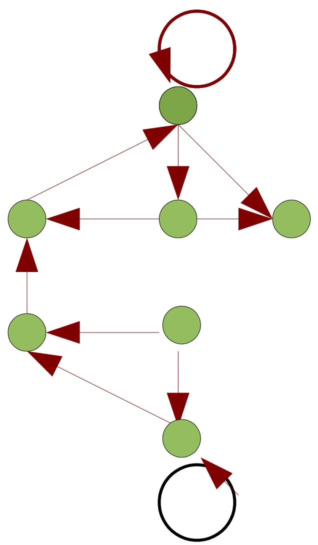
- **Actions stochastiques :**

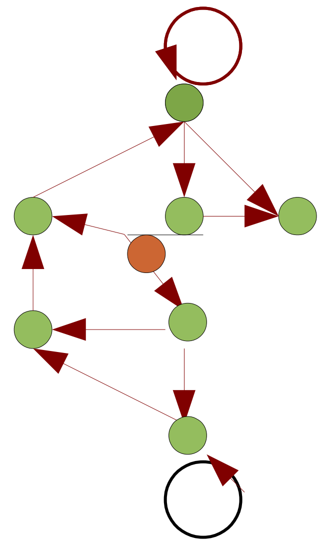
$$T: S \times A \rightarrow P(S)$$

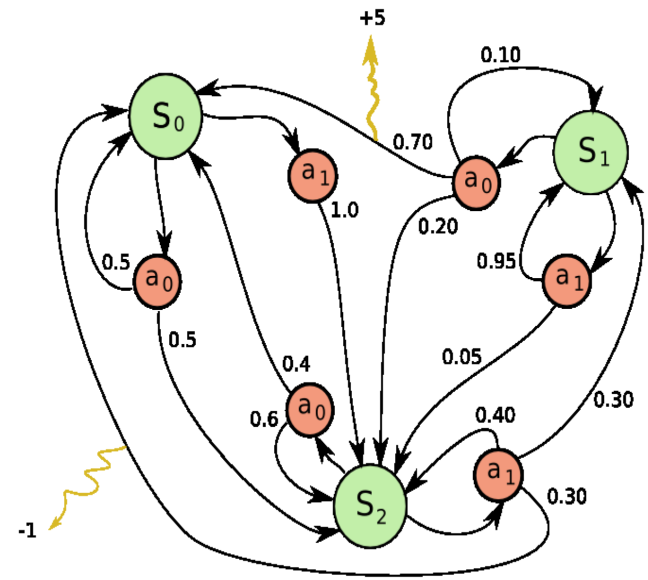
Pour tout etat et tout action,T spécifie une distribution de probabilité sur les états suivants P(s'|s,a).

Hypothèse de Markov : T(*s*; a) et r(s; a) ne dépendent que de l'état et de l'action choisie

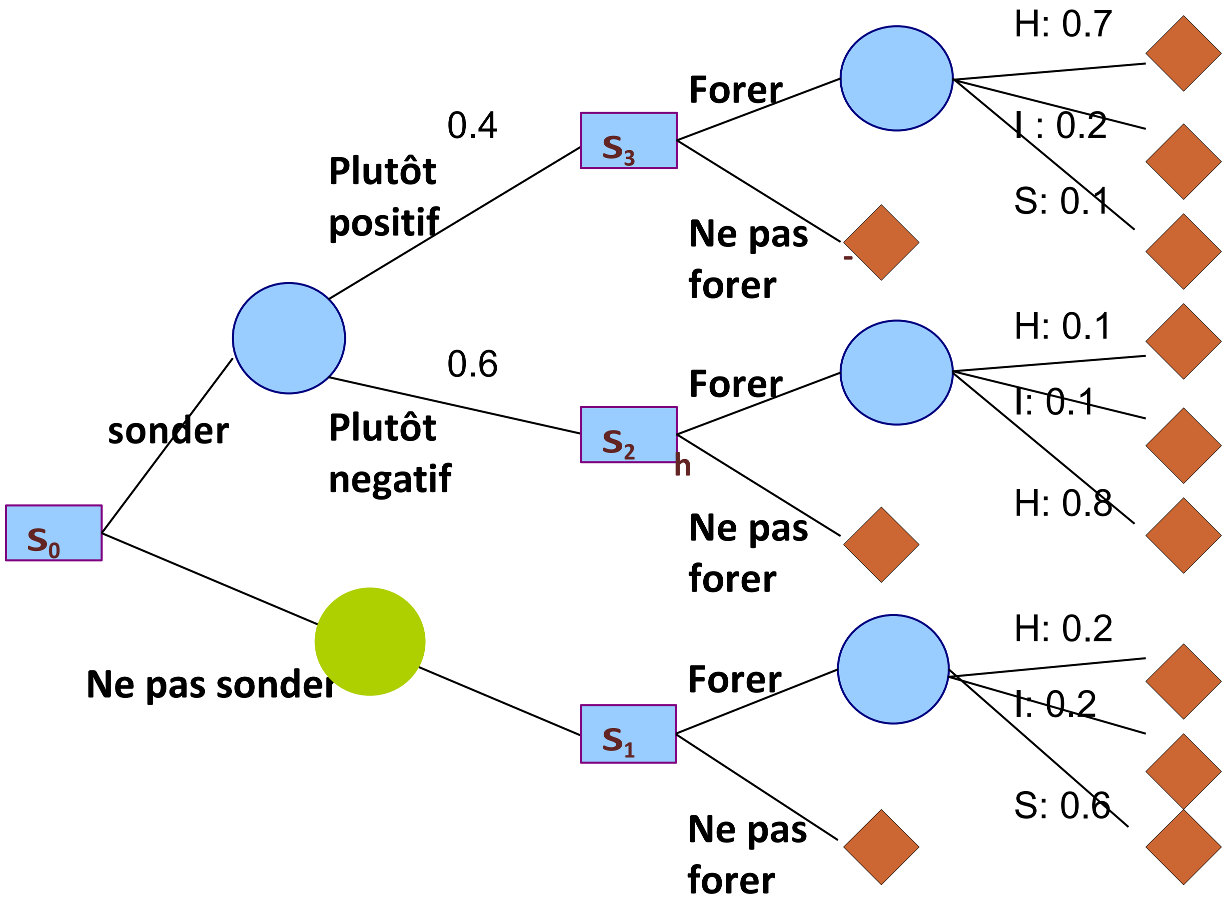
Automates déterministes, automates stochastiques







Arbre de décision (probabiliste)

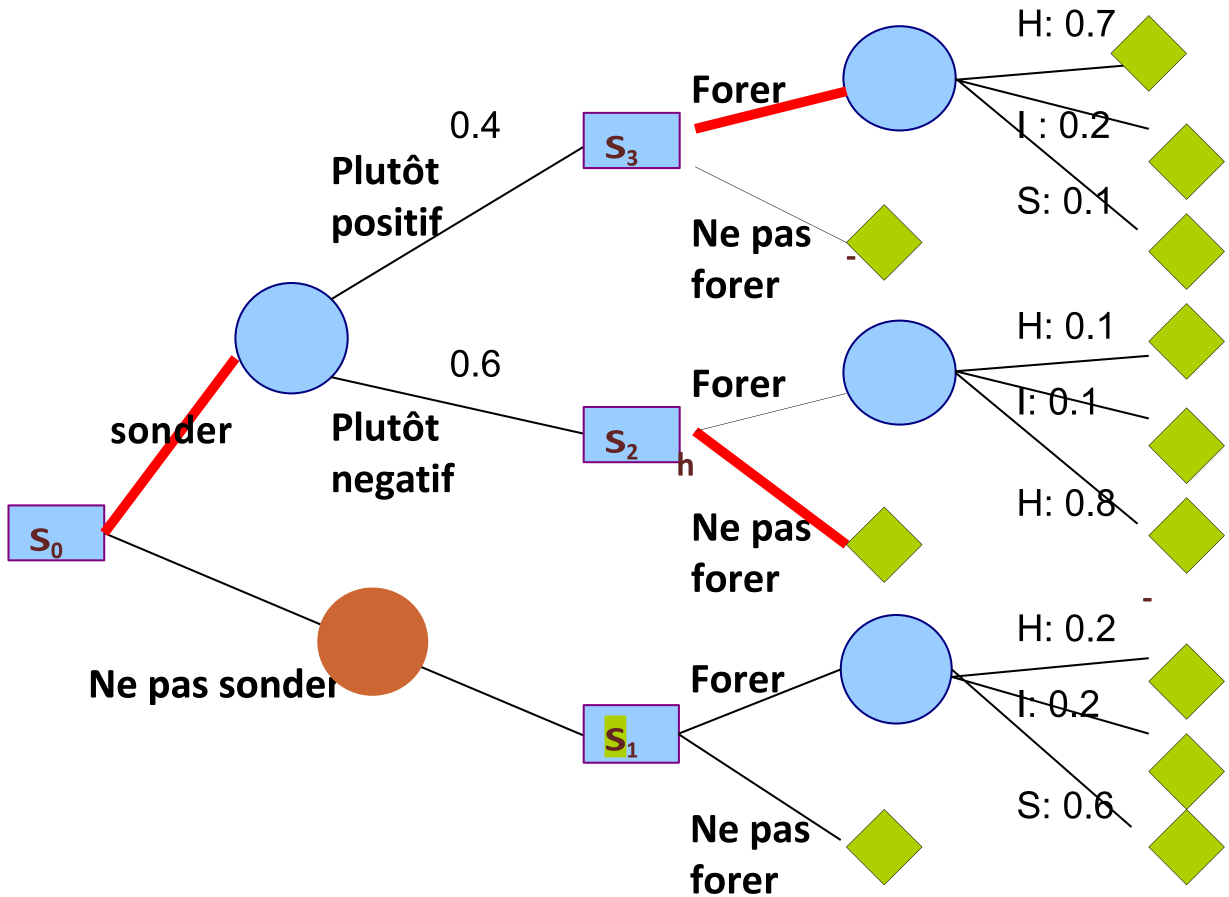


Sondage : -10 K
Forage : - 70 K

Humide: 270 K
Imbibé: 120 K
Sec: 0 K

Sur chaque arc (Ci,N) issu d'un noeud chance : P(N|Ci,PRED(Ci))

Politique

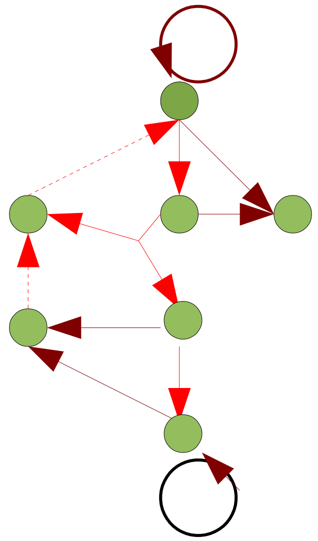


Sondage : -10 K
Forage : - 70 K

Humide: 270 K
Imbibé: 120 K
Sec: 0 K

Représenter une solution

- Une politique π est une fonction de S dans A.



- Suivre une politique
  1. Déterminer l'état courant s
  2. Exécuter l'action π (s)
  3. Retourner en 1

- Suppose l'observabilité totale du système (on sait toujours où on est)

Evaluer la qualité d'une politique (cas stochatique)

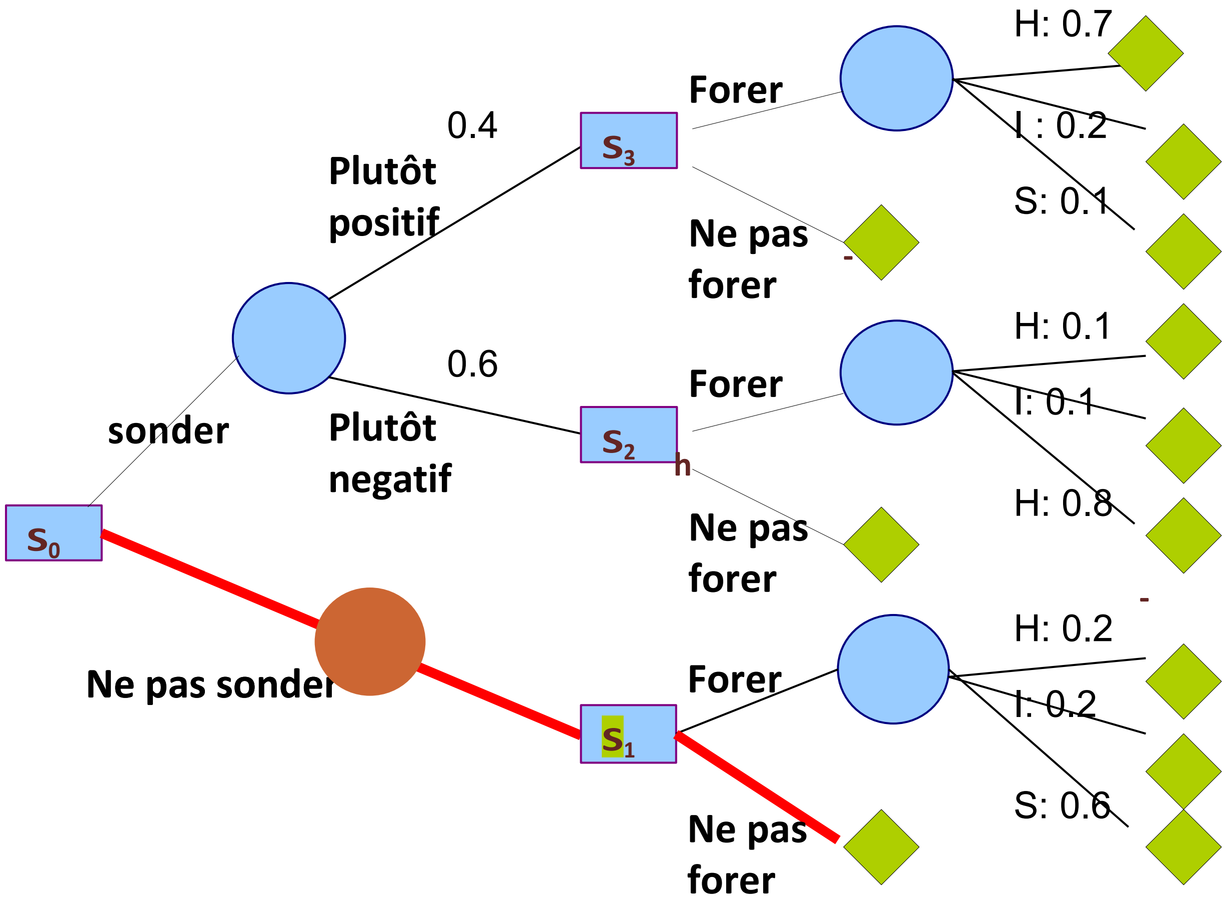
- Une politique d'action = Une fonction $\pi : S \rightarrow A$.

- Cas déterministe, π définit une trajectoire pour tout état initial s0 :

$$\tau\ (\pi, s0) = (s0, \pi(s0), s1, \pi(s1), s2, \pi(s2), \ldots)$$

On peut choisir la politique maximisant l'utilité $R(t) = \sum_{i=0,h} r_{t+i+1}$ pour $t=0$

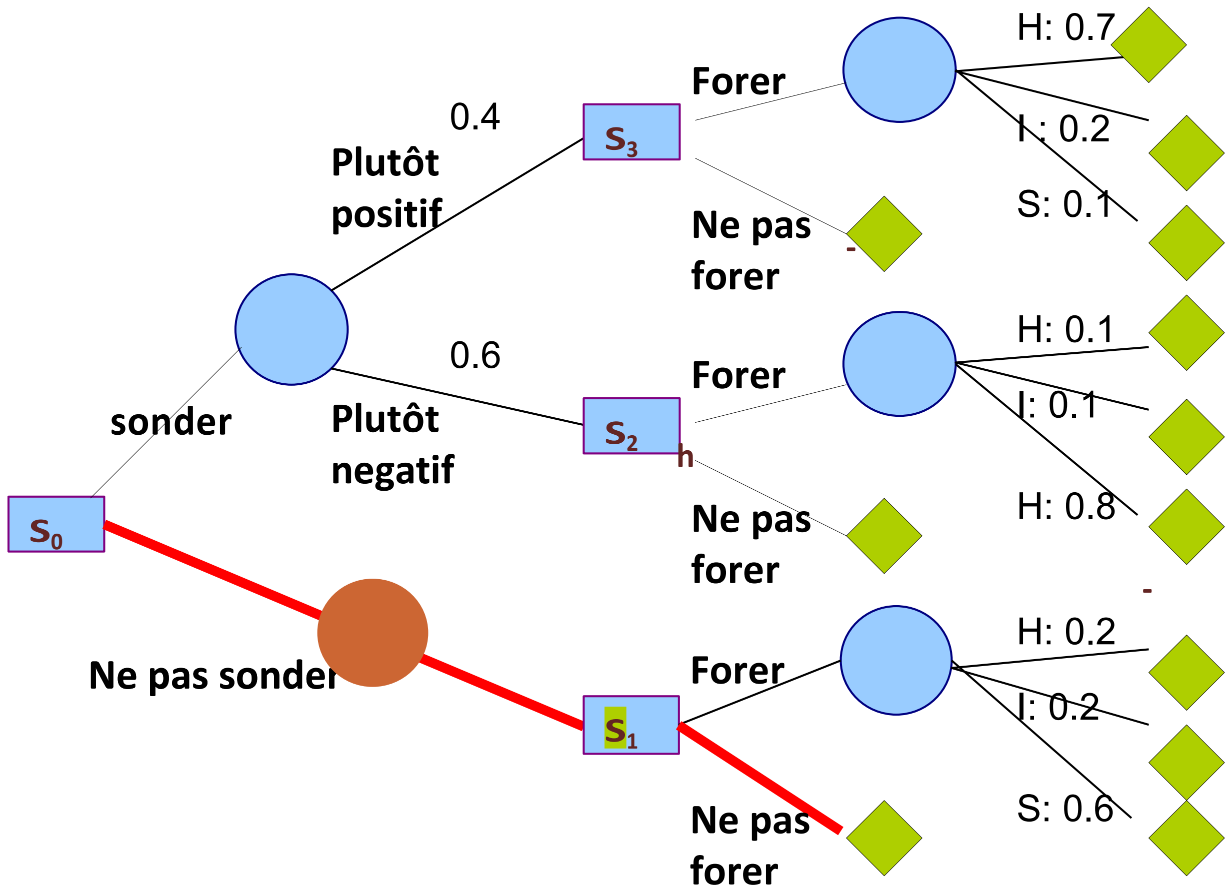
Politique



Sondage : -10 K
Forage : - 70 K

Humide: 270 K
Imbibé: 120 K
Sec: 0 K

Politique



Sondage : -10 K
Forage : - 70 K

Humide: 270 K
Imbibé: 120 K
Sec: 0 K

Evaluer la qualité d'une politique (cas stochatique)

- Une politique d'action = Une fonction $\pi : S \rightarrow A$.

- Cas déterministe, π définit une trajectoire pour tout état initial s0 :

$$\tau\ (\pi, s0) = (s0, \pi(s0), s1, \pi(s1), s2, \pi(s2), \ldots)$$

On peut choisir la politique maximisant l'utilité $R(t) = \sum_{i=0,h} r_{t+i+1}$ pour $t=0$

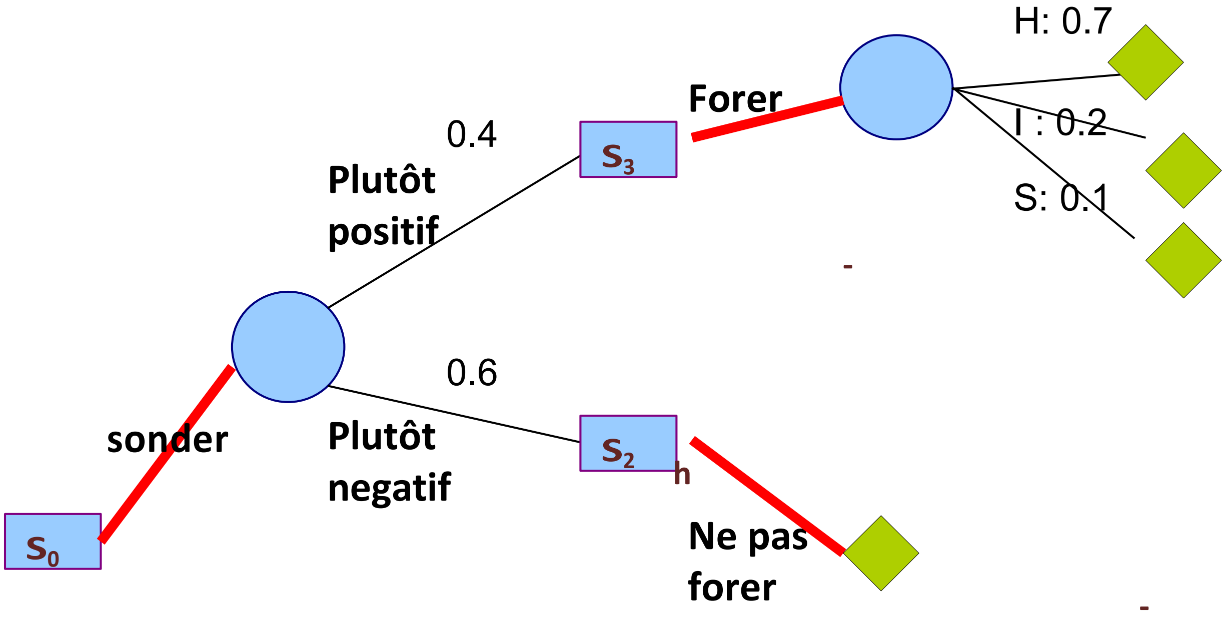
- Maintenant, π génère une distribution de probabilité sur les trajectoires

$$p(\tau|\pi, s0) = \prod_{t=0,\ h-1} p(s_{i+1} \mid s_i,\ \pi(t, s_i))$$

A chaque trajectoire une utilité

Politique optimale : π maximisant l'espérance mathématique de l'utilité

Politique → Distribution de Probabilité sur les utilités



Sondage : -10 K
Forage : - 70 K

Humide: 270 K
Imbibé: 120 K
Sec: 0 K

Equation de Bellman

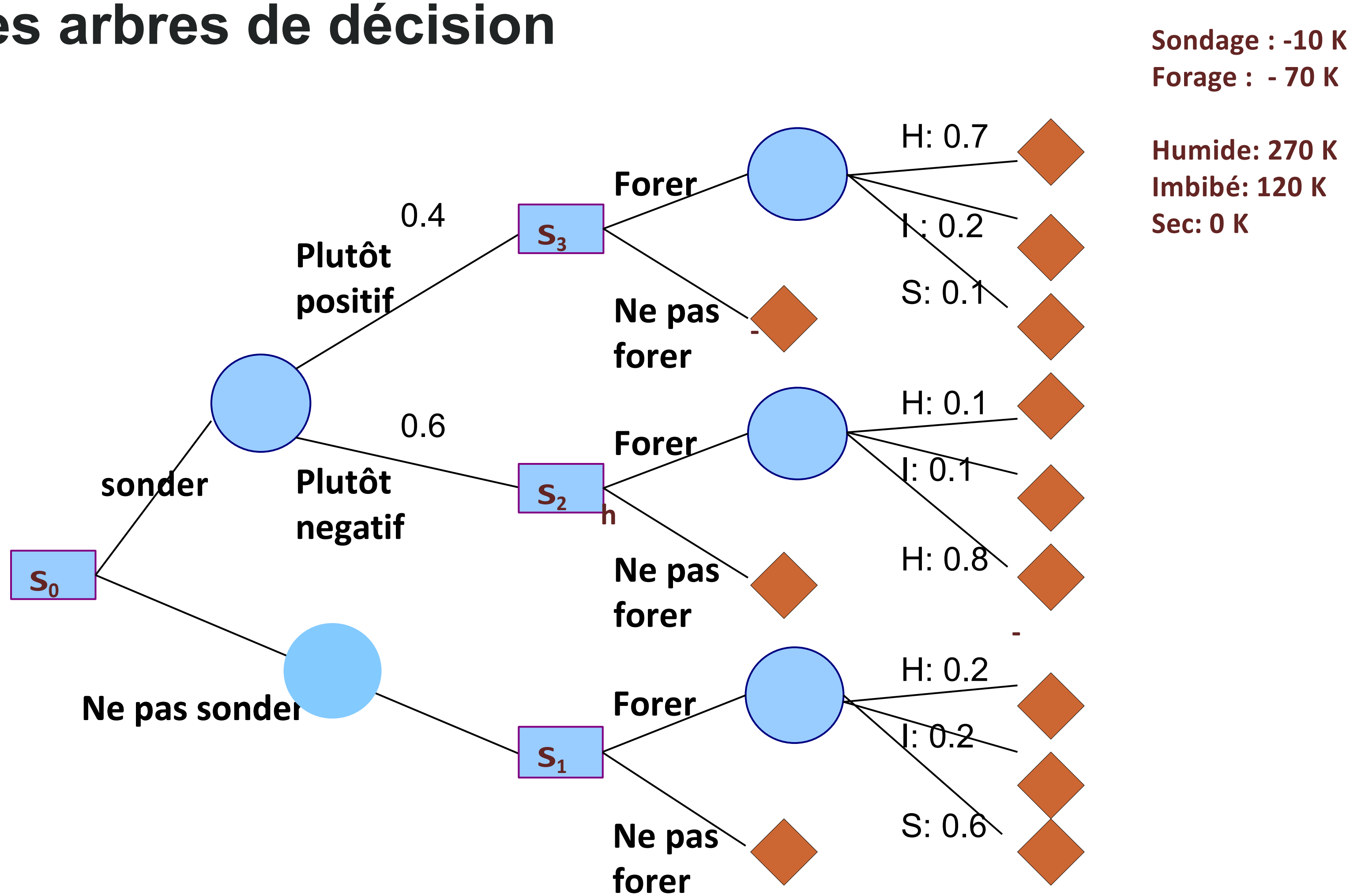
- Equations de Bellman: EU pour chaque etat selon la politique

$$V_\pi(s) = r(s,\pi(s)) + \sum_{s' \in S} P(s'|s,a).(\gamma) \, . \, V_\pi(s')$$

Maximiser $U(\pi) = V_\pi(s0)$; en fait, maximiser $V_\pi(s)$ pour chaque s;

- Comment résoudre l'équation de Bellman ?
  - Pour chaque s, 1 équation linéaire à |S| inconnues : |S| à |S| inconnues: Programmation linéaire (S petit)
  - Faire des simulations : methode de Monte Carlo
  - Programmation dynamique

Calcul d'une politique par programmation dynamique dans les arbres de décision



En partant de la fin, maximiser l'utilité esperée

Calcul d'une politique par programmation dynamique dans les arbres de décision

- Partir de la fin : pour chaque s, choisir l'action qui maximise EU

```
Pour t = N-1 à 0 faire
  Pour tout s de S faire
      Pour tout a de A(s) faire
      U(s,a) = r(s,a)+∑s'∈S P(s'|s,a).U(s')
       Fin pour
       π(s) = argmax a  U(s,a)
       U(s) = U(s,π(s))
  Fin pour
Fin pour
```

Algo fondé sur principe de monotonie + decomposition de l'EU

Calcul d'une politique par programmation dynamique dans les arbres de décision

- **Monotonie :**
  toute sous politique d'une politique optimale est optimale

$$EU(a) > EU(a') \Leftrightarrow EU(\, p \,.\, a + (1 - p)\, a'') > EU(\, p \,.\, a' + (1 - p)\, a'')$$

- **Decomposition :**
  Les utilités espérées se calculent à partir des utilités esperées

$$EU(\, p \,.\, a + (1 - p)\, a'') = p .\, EU(a) + (1 - p)\, EU(a'')$$

Calcul d'une politique par programmation dynamique dans MDP à Horizon fini

- **On peut faire comme pour les arbres de décision :-)**

```
Pour t = N-1 à 0 faire
  Pour tout s de St faire
      Pour tout a de A(s) faire
      U(s,a) = r(s,a)+∑s'∈S P(s'|s,a).U(s')
      Fin pour
      π(s) = argmax a  U(s,a)
      U(s) = U(s,π(s))
  Fin pour
Fin pour
```

Horizon Infini : Iterative *policy evaluation*

Pour une politique *fixée* π et un coéf d'oubli γ donné

```
Initialisation : pour tout s, U(s) = 0

Repeter:
 Δ = 0;
 Pour tout s de S:
     u = U(s);
     U(s) = Σ_{s'∈S} (P(s'|s,π(s))[r(s,π(s),s')+γ.U(s'))];
     Δ = max(Δ, u - U(s));
Jusqu'à Δ<ε

Résultat : U ≈ U_π
```

Iterative policy evaluation (exemple)

→	→	→	○ Fin (10)
↑	Obstacle (-10)	↑	↑
↑	←	↑	←

Cout du deplacement : -1 si nord, 0 pour les autres, discount 0.75

Politique:

→	→	→	○ Fin (10)
↑	Obstacle (-10)	↑	↑
↑	←	↑	←

Cas stochastique : Est : Est (0.5), reste sur place (0.5)

Policy iteration

→	→	→	○ Fin (1)
↑	obstacle	↑	↑
↑	←	↑	←

- Et si on allait à l'Est en (1,2)?

- **Idée:**

  - **on part d'une politique quelconque** **$\pi 0$**
  - **on calcule (policy evaluation)** **$U_{\pi 0}$**
  - **grace à $U_{\pi 0}$ on trouve une politique ameliorée** **$\pi 1$**
  - **et on continue jusqu'à stabilité ...**

$$\pi 0 \rightarrow U_{\pi 0} \rightarrow \pi 1 \rightarrow U_{\pi 1} \rightarrow \pi 2 \rightarrow U_{\pi 2} \rightarrow \ldots \rightarrow \pi^* \rightarrow U_{\pi^*}$$

Et ça converge !

Policy iteration

Pour coéf d'oubli donné ....

```
Initialisation : pour tout s, U(s) = 0 et π quelconque

Repeter:
  Policy Evaluation (c.f. algo prédédent)

  (Policy improvement)
  stable ← true
  Pour tout s de S:
    b ← U(s);
   π(s)← argmax_a Σ_{s'∈S} P(s'|s,T(a,s))[r(s,T(a,s))+ γ.U(s')];
   stable ← (b == π(s)) && stable ;
Tant que stable = false
```

Iteration de la valeur (Value Iteration)

- Chacune des itérations de Policy Iteration nécéssite l'exécution de toute l'évaluation …
- Principe de l'algorithme Value Iteration = effectuer seulement la première itération de l'algorithme Policy Evaluation
- ... et ca converge aussi !

Value Iteration

Pour coéf d'oubli donné ....

```
Initialisation : pour tout s, U(s) = 0 et π quelconque

 U ← Policy Evaluation (π)

Repeter:
 Δ = 0;
 Pour tout s de S:
  u ← U(s);
   U(s)  = max_a Σ_{s'∈S} P(s'|s,T(a,s))[r(s,T(a,s))+ γ.U(s')];
   Δ =  max(Δ, u − U(s)|;
Jusqu'à Δ <ε

π(s)← argmax_a Σ_{s'∈S} P(s'|s,T(a,s))[r(s,T(a,s))+ γ.U(s')];
```

Pour aller plus loin

- **POMDP : l'état (s') resultat de l'action (a) dans (s) est partiellement observable : de l'observation o on deduit une distribution de probabilité sur l'etat s' à partir de**
  - **p(o | s', a)**
  - **T(s ' | a, s)**
  - **p(s | ...), mon etat de croyance sur s**

- **Diagrammes d'influence :**
  Réseau Bayesien + variables de décision + utilités

- **D'autres critères que l'utilité esperée**
  - **Utilité possibiliste (optimiste, pessimiste) : MDP possibilistes**
  - **MDP algébriques**
  - **Choquet (RDU, Probas imprecises) ? Aie ! → Complexité accrue !**

Réferences

- *Decision Theory - An introduction to Dynamic Programming and Sequential Decisions,* J. Bather, Wiley, 2000.

- *Reinforcement Learning - an introduction*, R. Sutton & A. Barto, MIT Press, 1998.

- *Learning to solve Markovian Decision Processes (http ://www.cs.colorado.edu/baveja/)*, S.P. Singh, PhD Dissertation, Department on Computer Science, MIT, 1994.

- *Algorithms for Sequential Decision Making* (ftp://ftp.cs.brown.edu/pub/techreports/96/cs96-09.pdf), M. Littman, Department of Computer Science, Brown University, 1996.

- Algebraic Markov Decision Processes. Perny, Patrice and Spanjaard, Olivier and Weng, Paul (2005). In 19th International Joint Conference on Artificial Intelligence, pp. 1372--1377.