

COMP0271: Inteligência Artificial

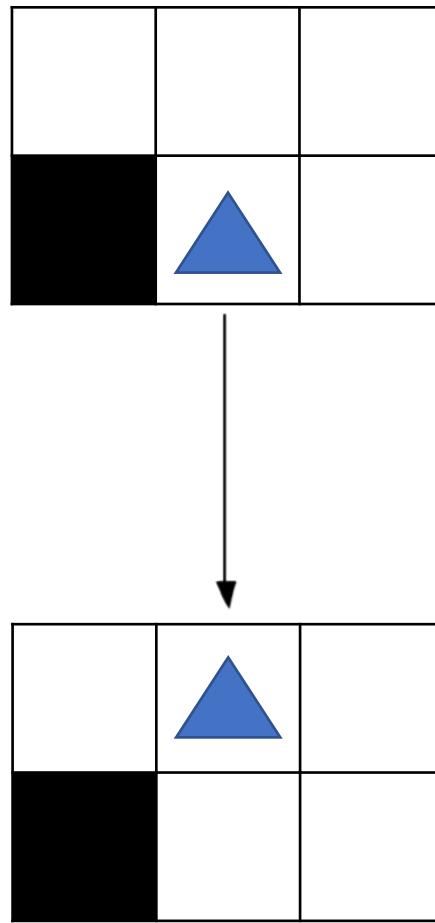
Processos de Decisão Markoviana (MDP)



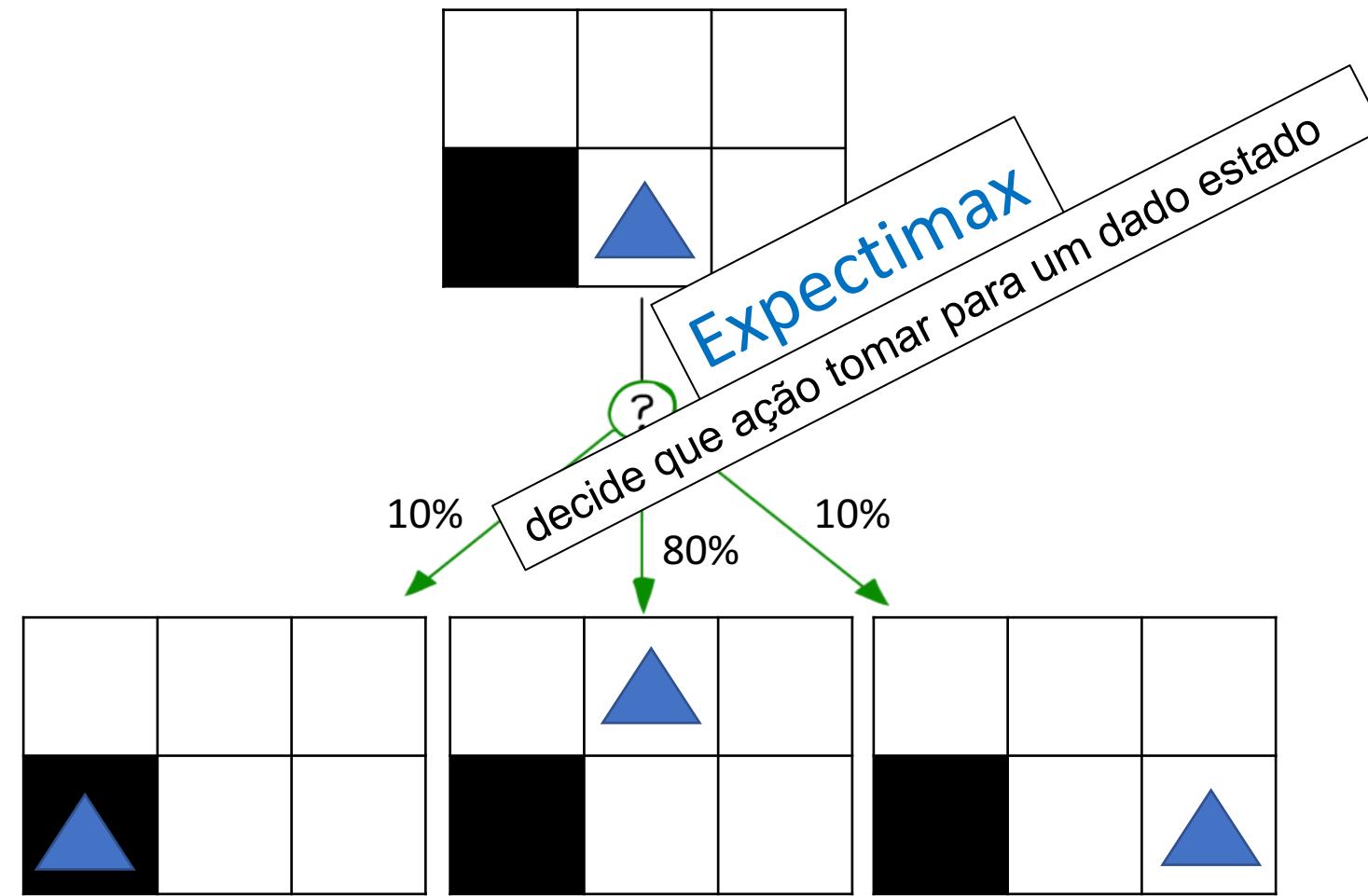
Professor: Hendrik Macedo

Universidade Federal de Sergipe, Brasil

Ambiente Determinístico



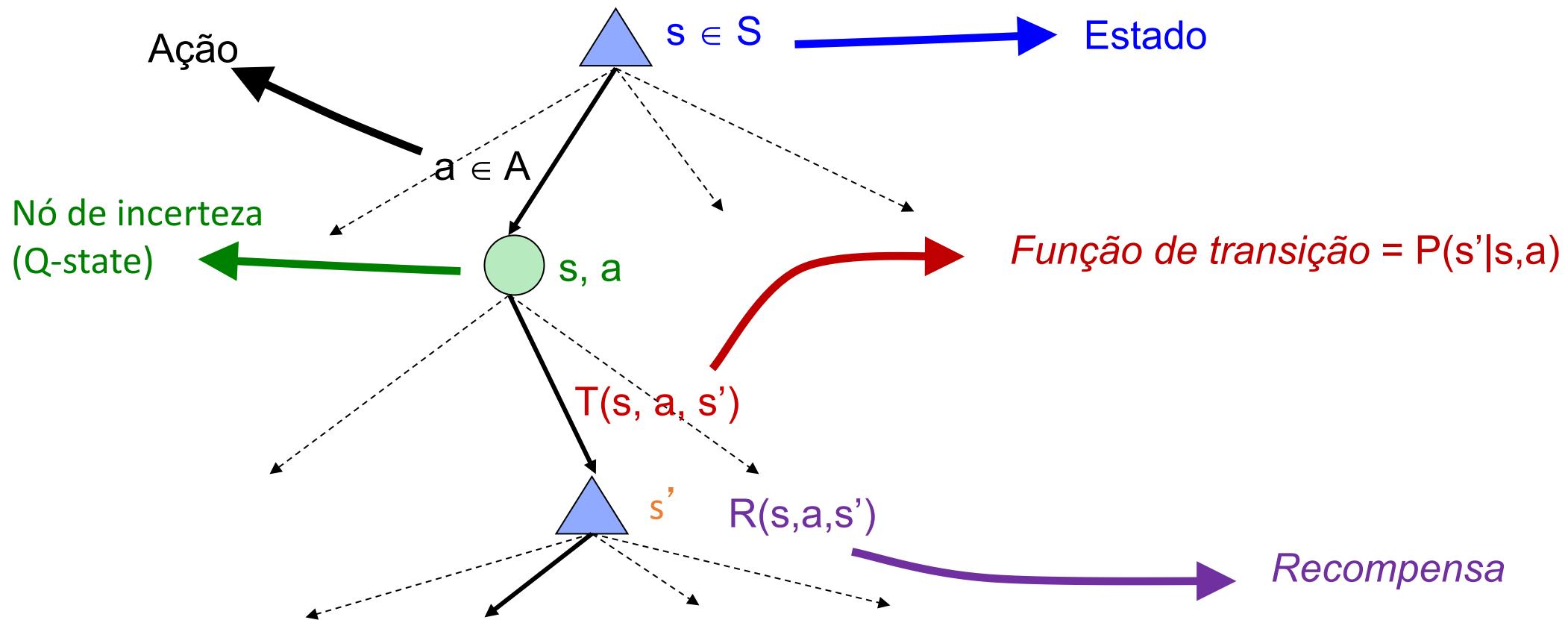
Ambiente Estocástico



E se ao invés da escolha de uma ação singular, precisarmos decidir por toda uma **sequência de ações**?

Processos de Decisão Markoviana

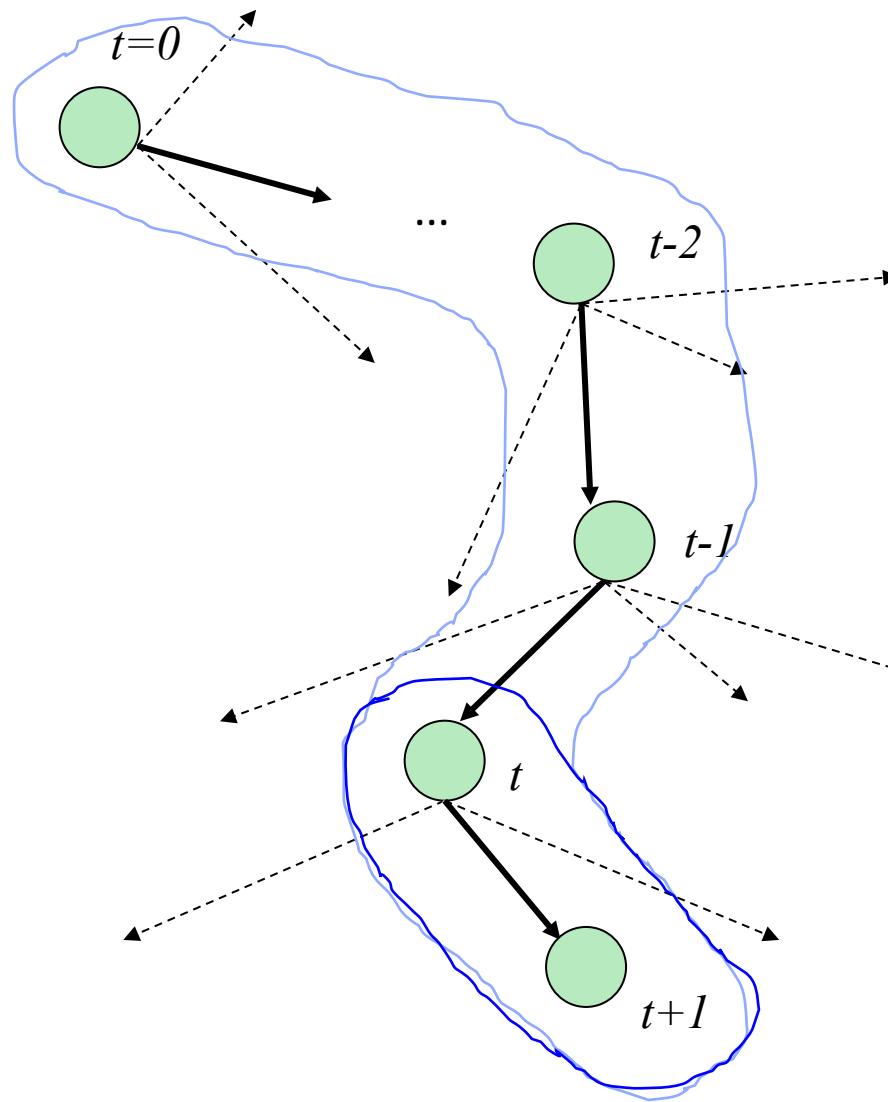
- Cada estado MDP projeta uma árvore de busca expectimax



Exemplo



Do que se trata Markov?



"Dado o estado presente, o futuro e o passado são independentes."

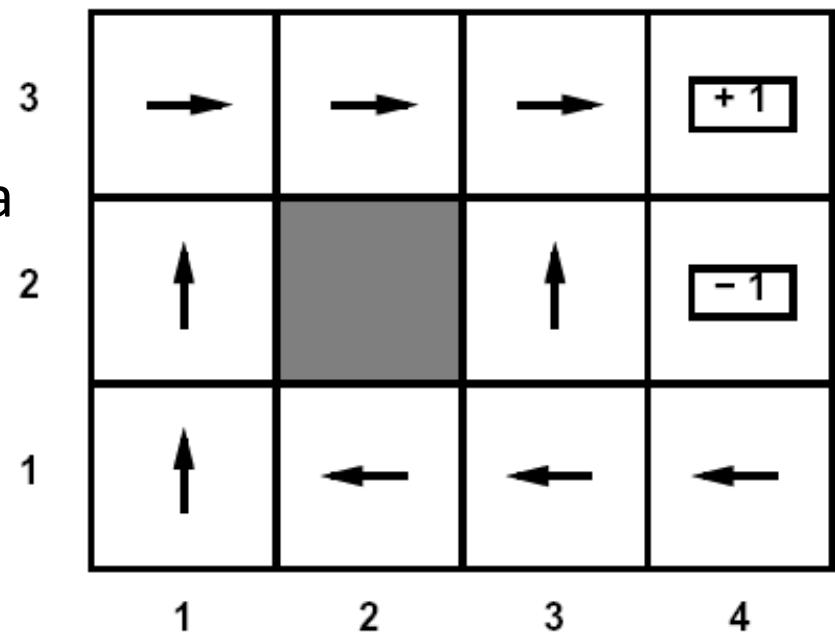


Andrey Markov
(1856-1922)

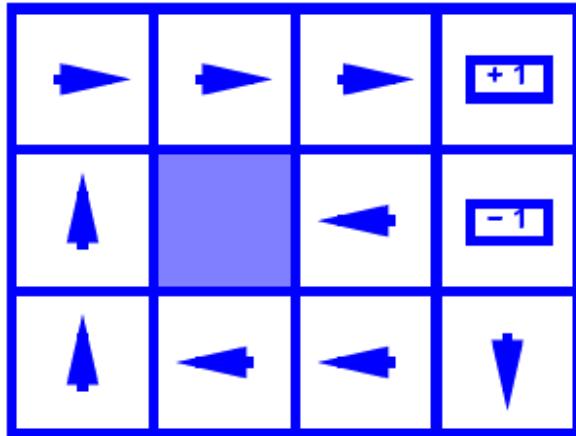
$$\begin{aligned} &P(S_{t+1} = s' | S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1}, \dots, S_0 = s_0) \\ &= \\ &P(S_{t+1} = s' | S_t = s_t, A_t = a_t) \end{aligned}$$

Plano vs. Política (Policy)

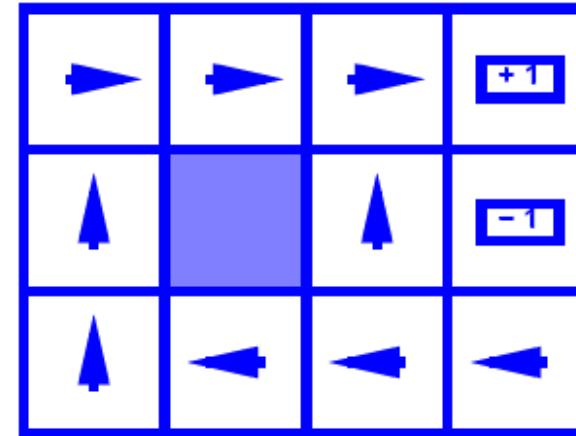
- Ambiente determinístico e monoagente → Plano
- MDP → Política π^* : $S \rightarrow A$
 - π define uma **ação para cada estado**
 - Política ótima π^* : maximiza a utilidade esperada
 - Política explícita define um agente reativo



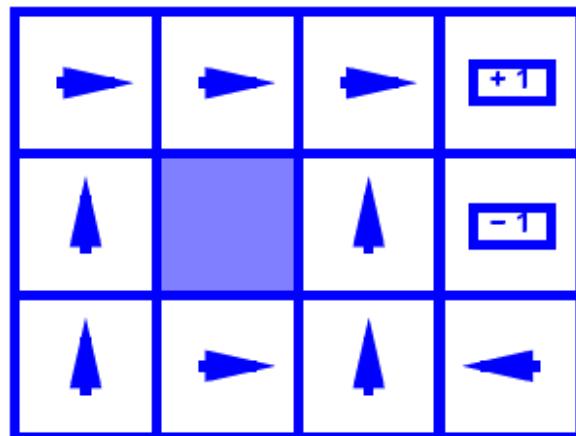
Políticas



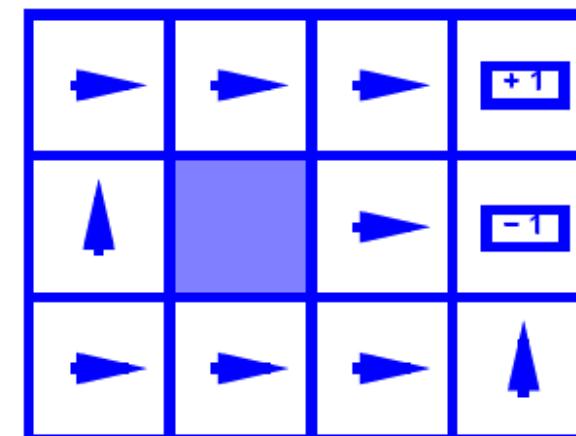
$$R(s) = -0.01$$



$$R(s) = -0.03$$



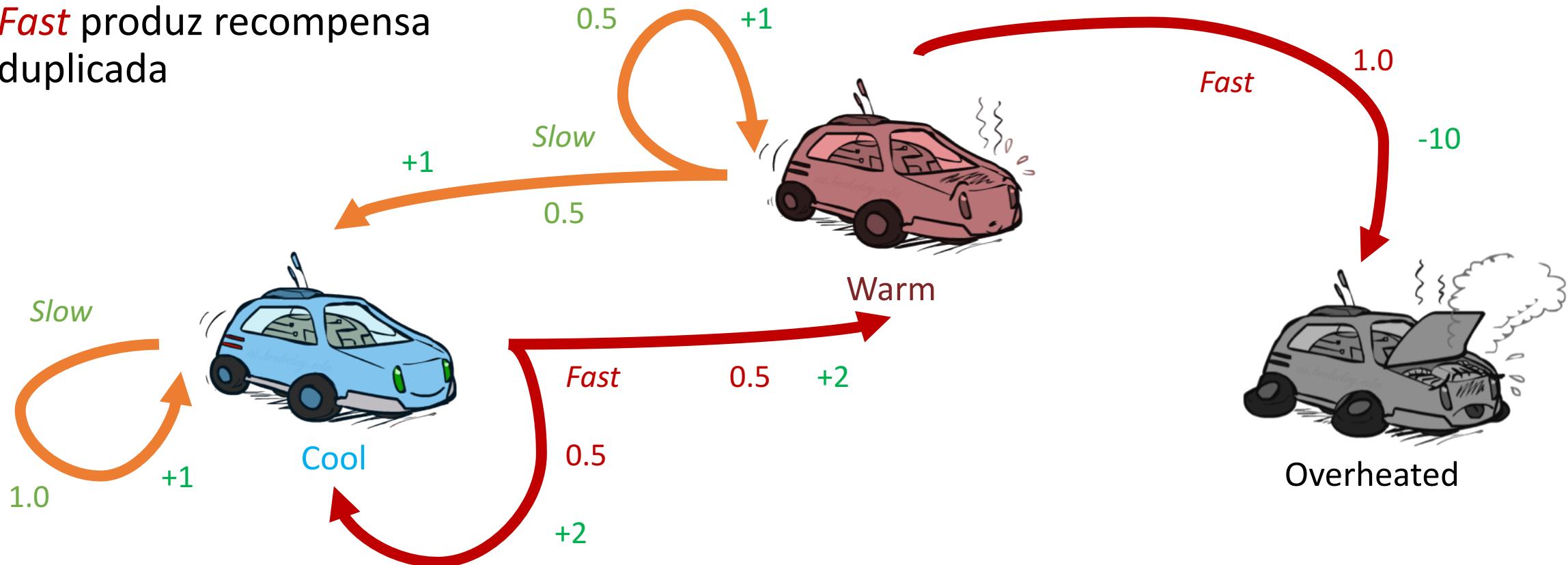
$$R(s) = -0.4$$



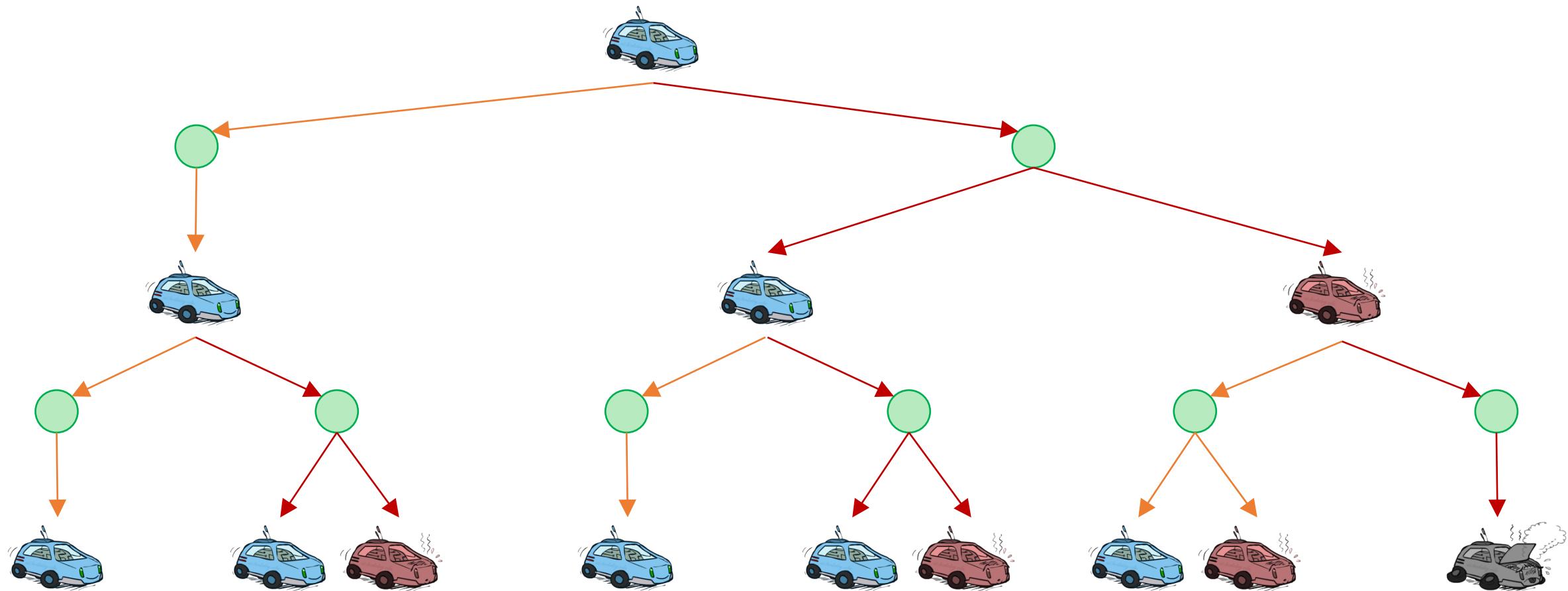
$$R(s) = -2.0$$

Carro autônomo

- 3 estados: **Cool**, **Warm**, Overheated
- 2 ações: **Slow**, **Fast**
- **Fast** produz recompensa duplicada



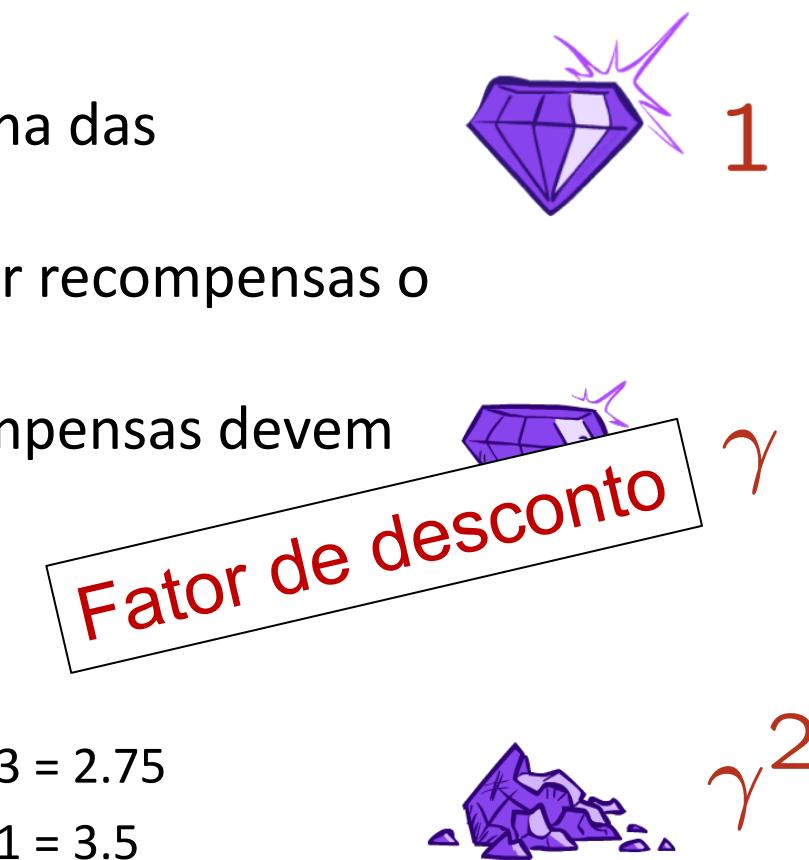
Carro autônomo .: Árvore de busca



Utilidades de sequência de ações

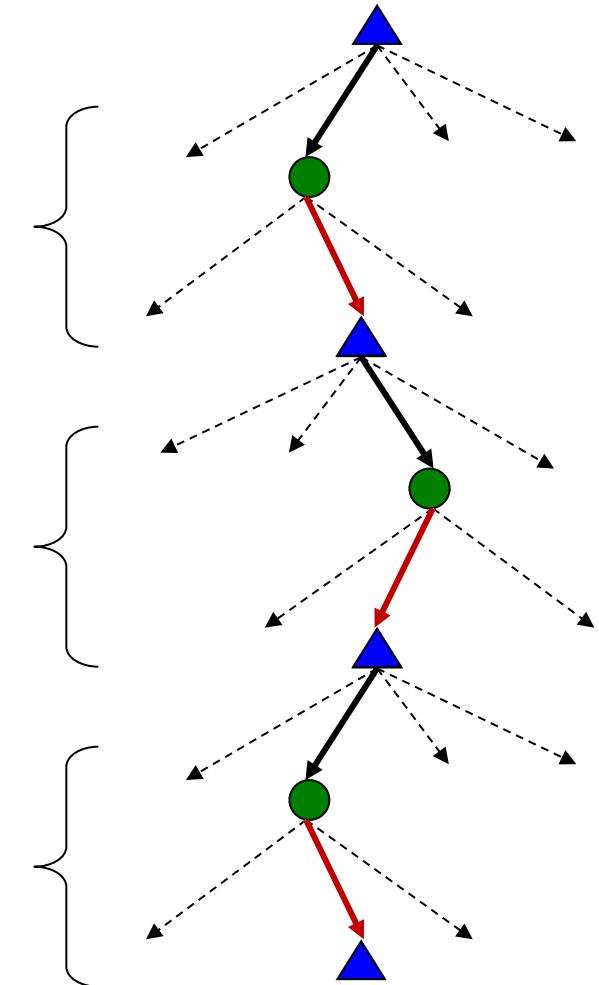
- Como computar?

- É racional maximizar a soma das recompensas
- Também é racional preferir recompensas o quanto antes
- Solução: valores das recompensas devem cair exponencialmente



- Ex: Fator de desconto de 0.5

- $U([1,2,3]) = 1*1 + 0.5*2 + 0.25*3 = 2.75$
- $U([3,2,1]) = 1*3 + 0.5*2 + 0.25*1 = 3.5$
- $U([1,2,3]) < U([3,2,1])$



Q: Desconto

- Dado:



- Ações: *Leste*, *Oeste*, e *Sair* (apenas disponível nos estados de saída a e e)

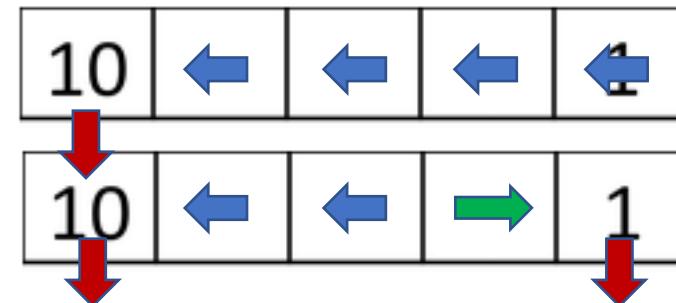


- Ambiente determinístico

- Q1: Para $\gamma = 1$, qual é a política ótima?

- Q2: Para $\gamma = 0.1$, qual é a política ótima?

- Q3: Para qual valor de γ Oeste e Leste são equivalentes em $s = d$?



$$1\gamma = 10\gamma^3 \rightarrow \gamma^3 = 1/10 \rightarrow \gamma \approx 0.46$$

Solução dos MDPs

Valores ótimos

- Utilidade de um estado s :

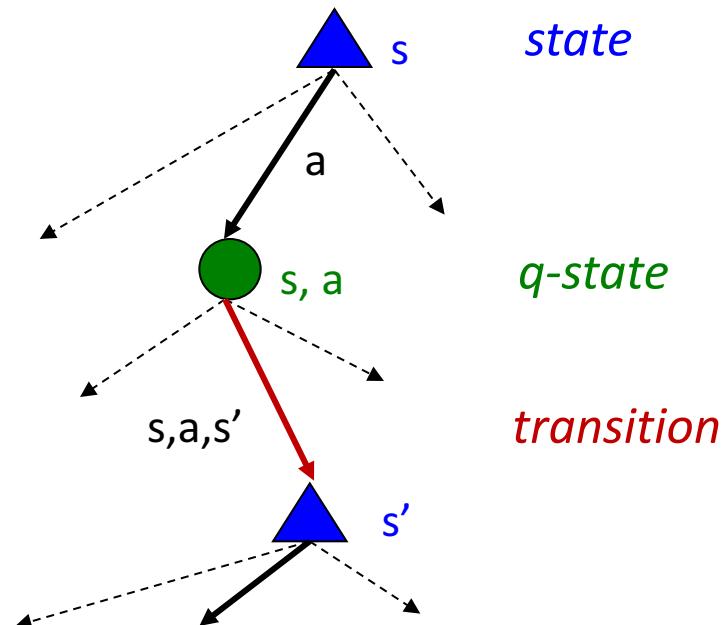
$V^*(s)$ = utilidade esperada ao começar em s e agir de forma ótima

- Utilidade de q-state (s,a) :

$Q^*(s,a)$ = utilidade esperada ao ter tomado ação a a partir do estado s e depois disso agir de forma ótima

- Política ótima:

$\pi^*(s)$ = ação ótima a partir do estado s

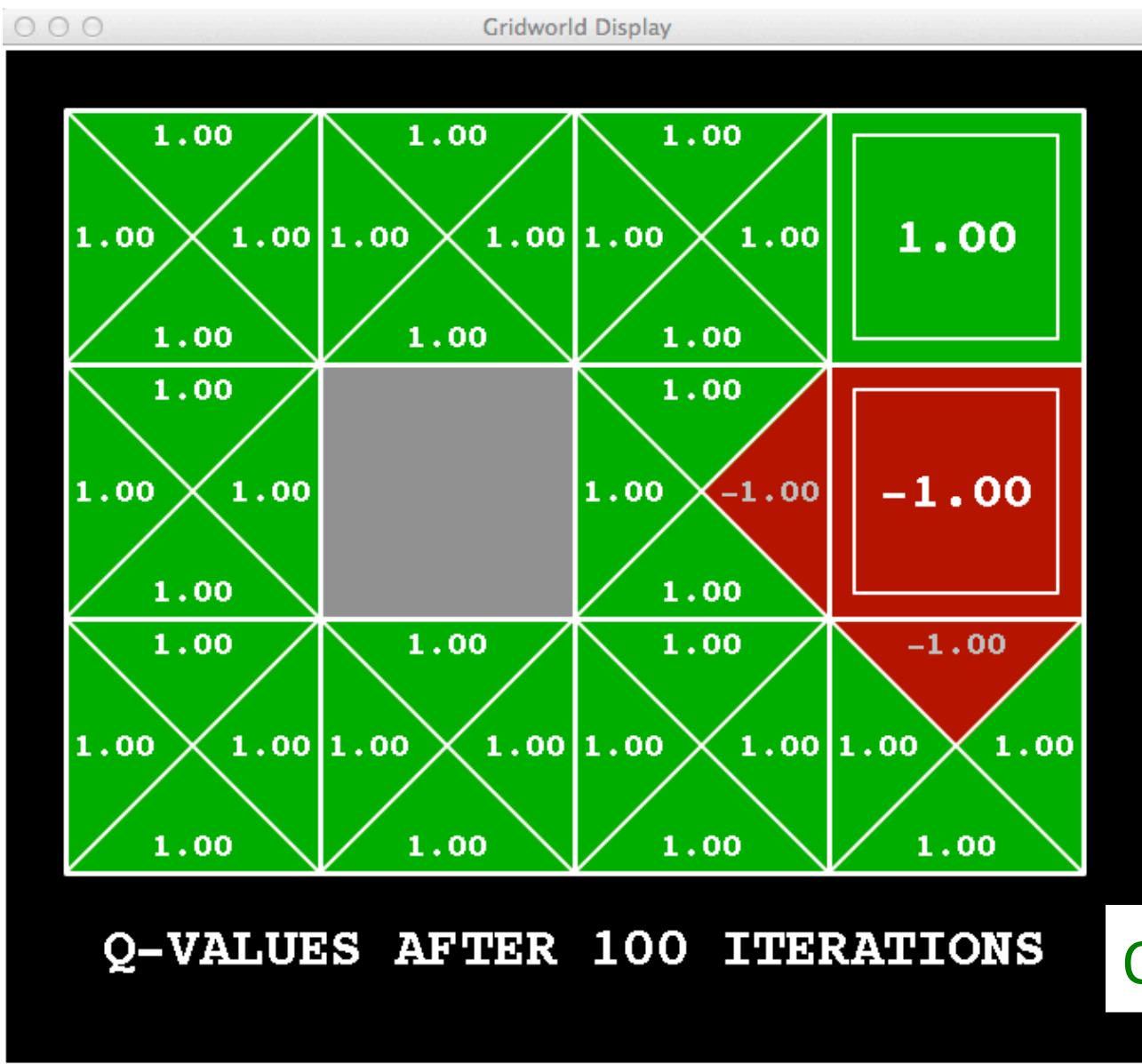




VALUES AFTER 100 ITERATIONS

$V^*(s)$

Ruído = 0
Desconto = 1
Recompensa = 0



Ruído = 0
Desconto = 1
Recompensa = 0



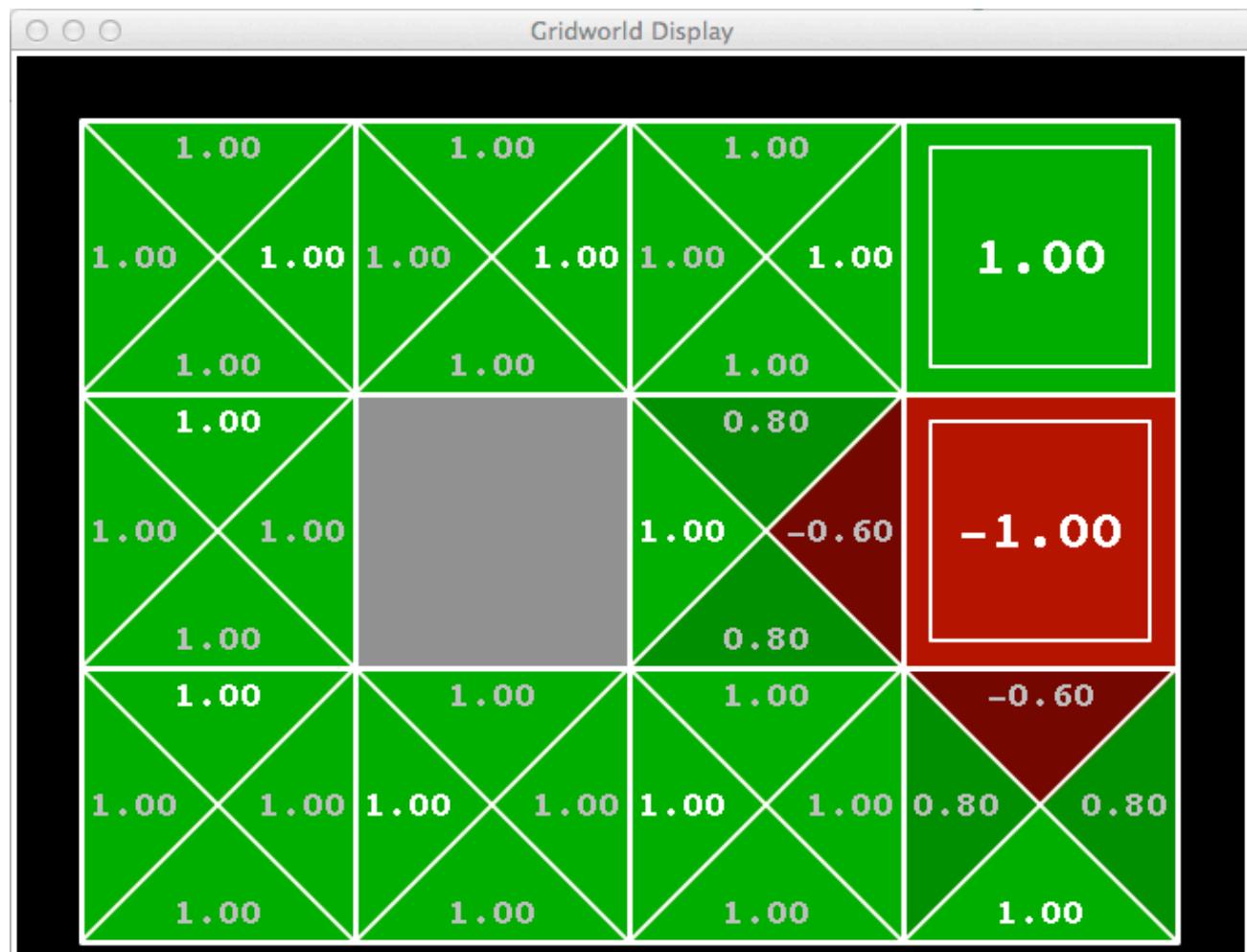
VALUES AFTER 100 ITERATIONS

$V^*(s)$

Ruído = 0.2

Desconto = 1

Recompensa = 0



Ruído = 0.2

Desconto = 1

Recompensa = 0

$Q^*(s,a)$

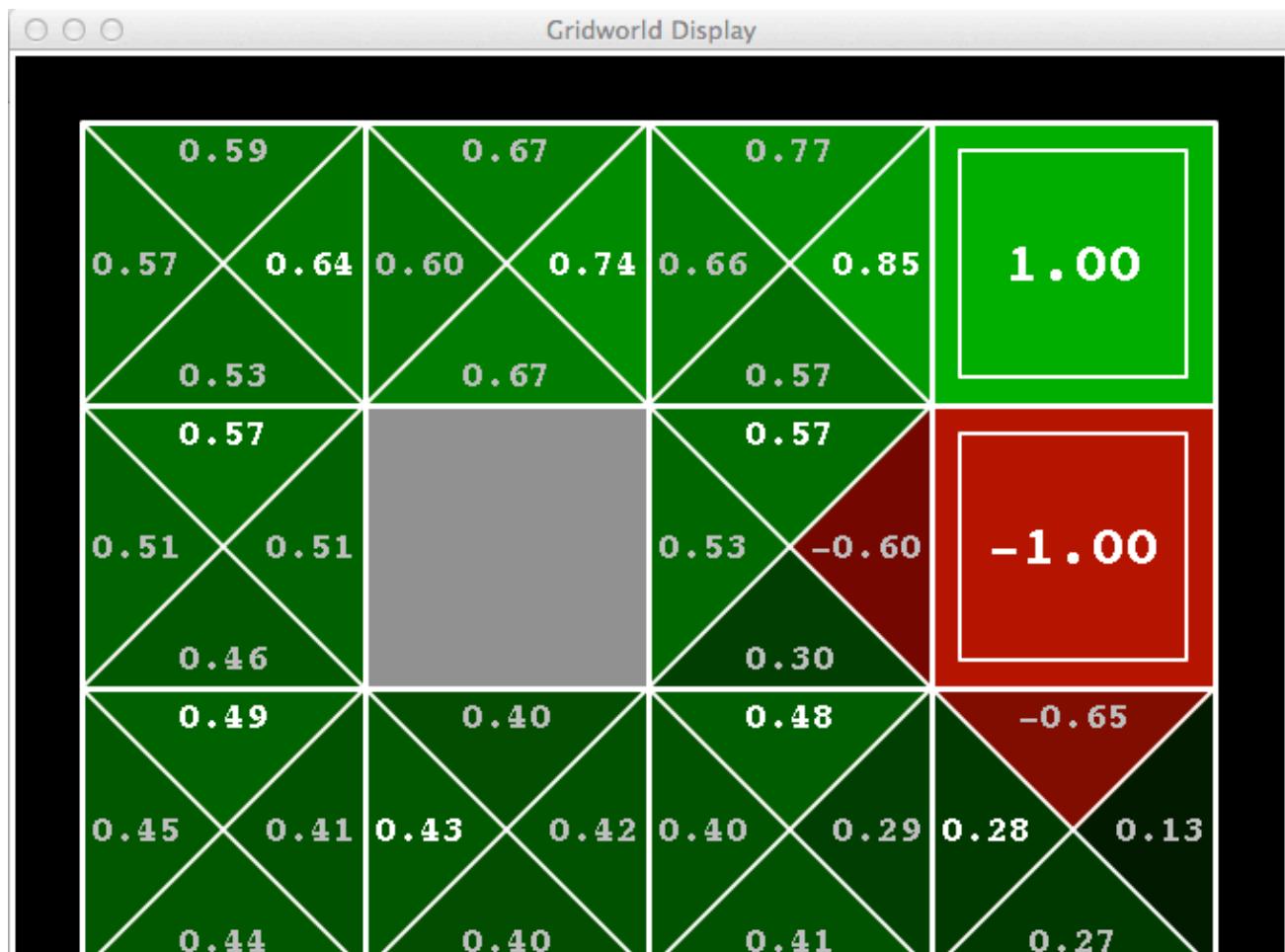


VALUES AFTER 100 ITERATIONS

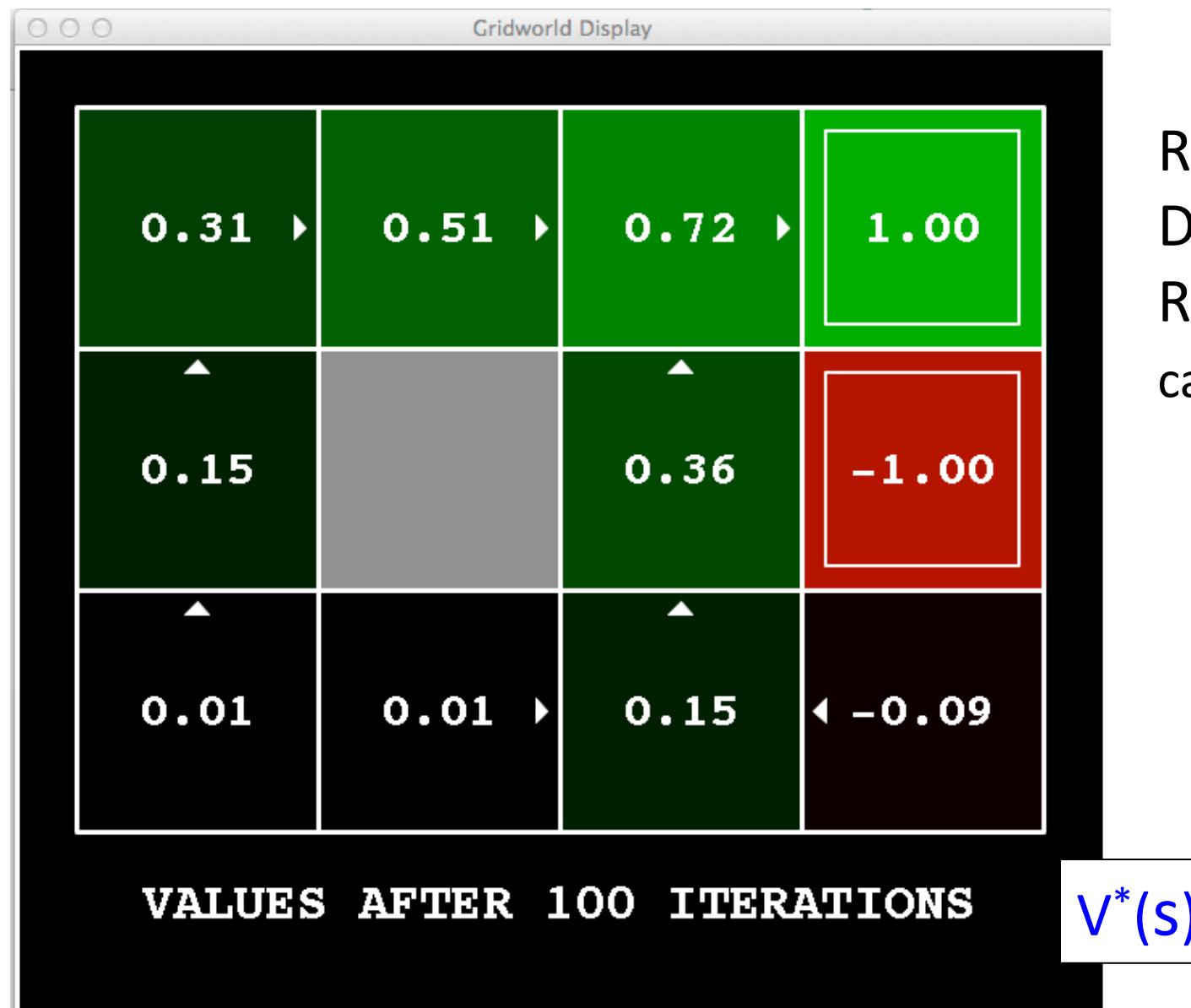
$V^*(s)$

Ruído = 0.2
Desconto = 0.9
Recompensa = 0

Ruído = 0.2
Desconto = 0.9
Recompensa = 0



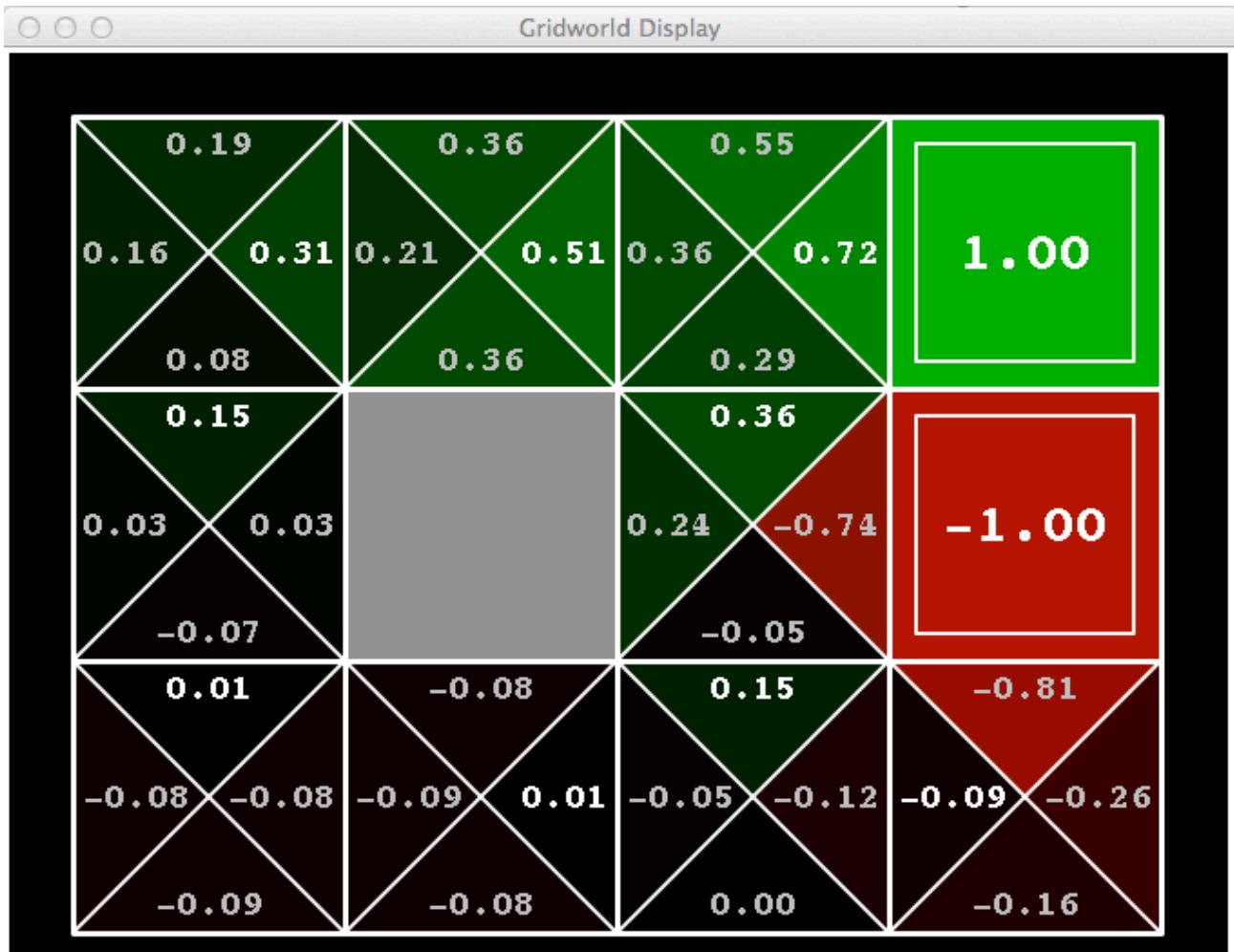
$Q^*(s,a)$



Ruído = 0.2

Desconto = 0.9

Recompensa (em
cada estado)= **-0.1**



Q-VALUES AFTER 100 ITERATIONS

$Q^*(s,a)$

Ruído = 0.2

Desconto = 0.9

Recompensa (em
cada estado)= -0.1

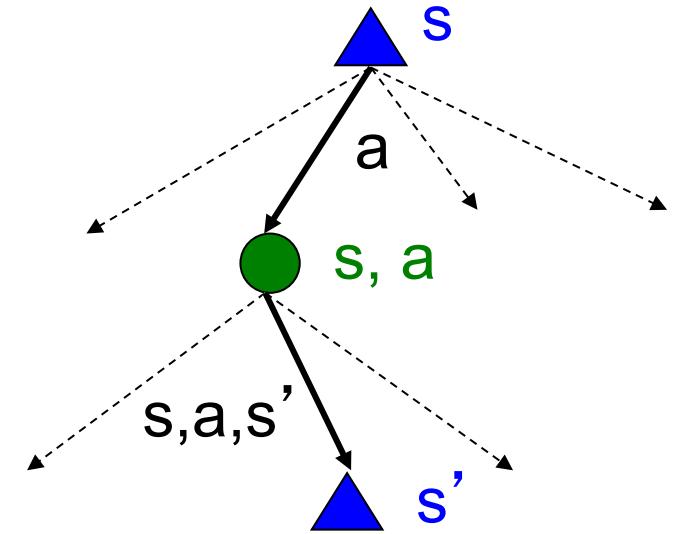
Valores dos estados

- Computar o (expectimax) valor de um estado
 - Utilidade esperada considerando ação ótima
 - Soma média das recompensas descontadas
- Definição recursiva do valor

$$V^*(s) = \max_a Q^*(s, a)$$

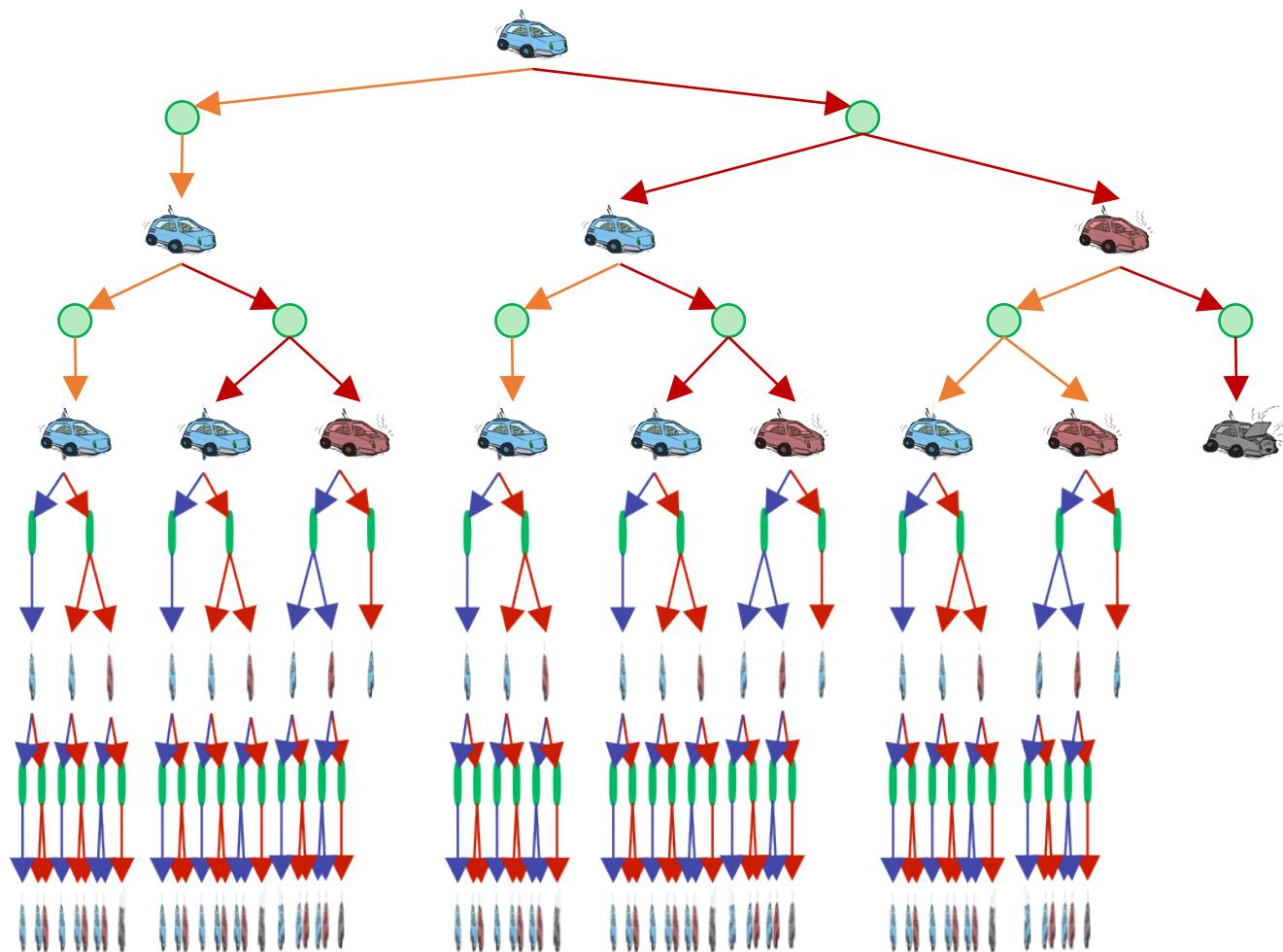
$$Q^*(s, a) = \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

$$V^*(s) = \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$



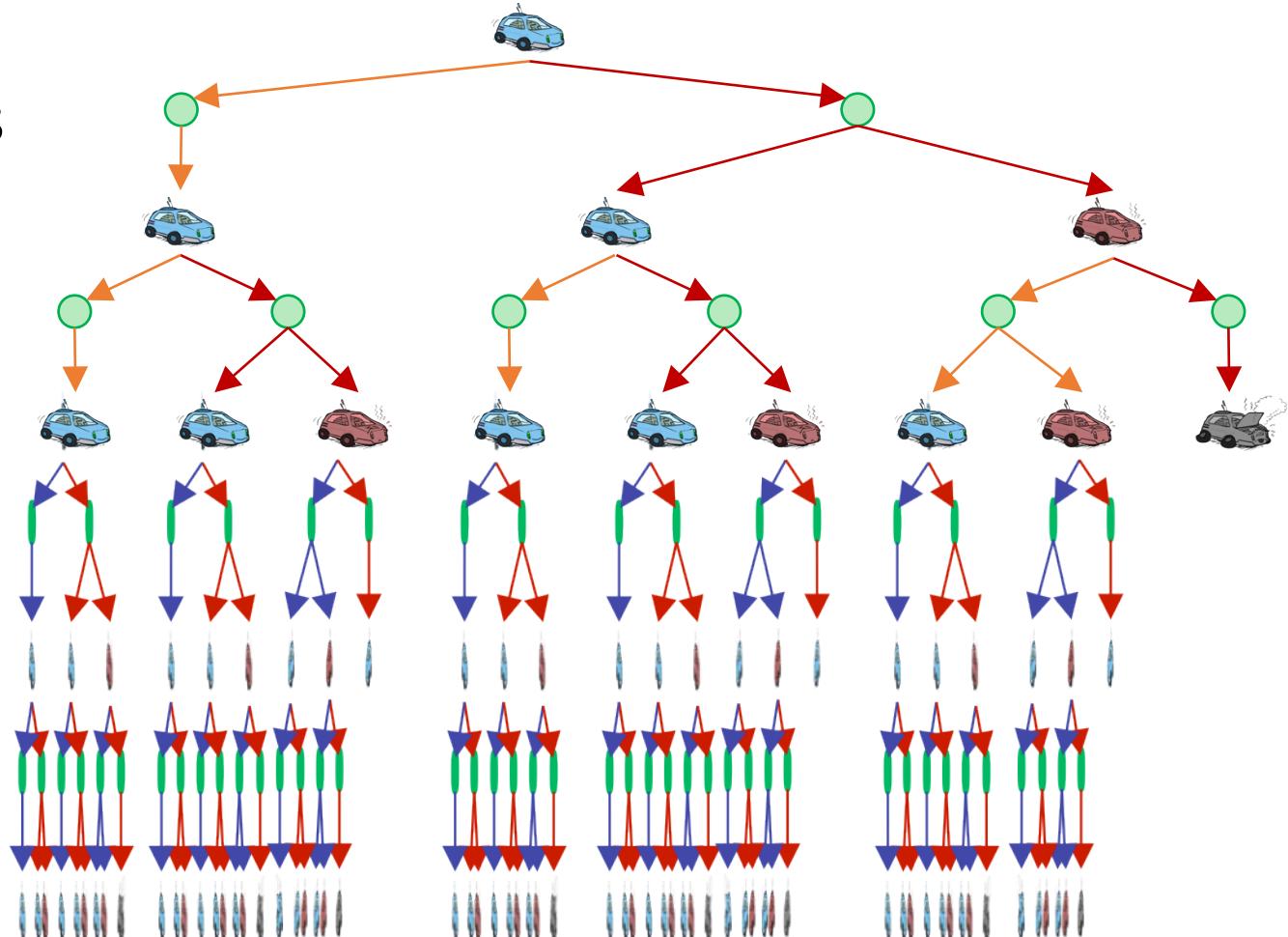
Equações de Bellman

Carro autônomo .: Árvore de busca



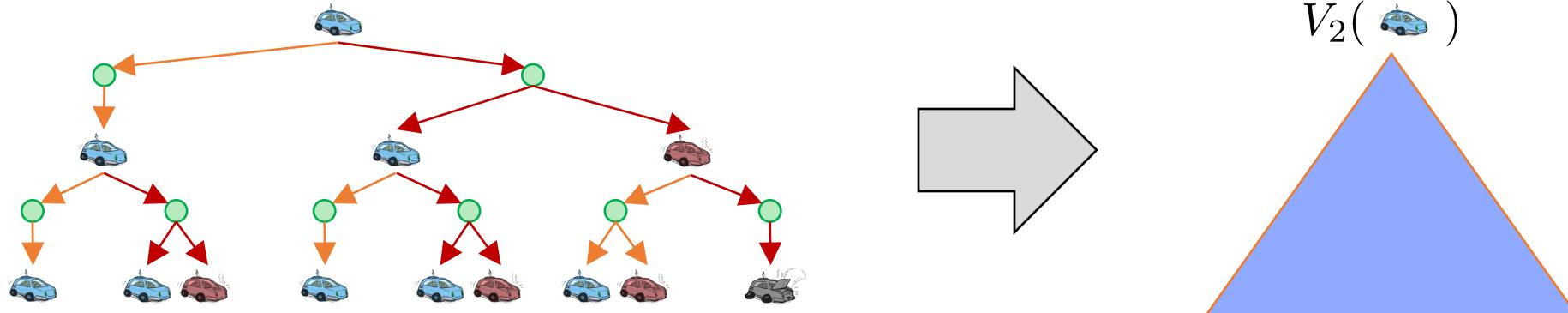
Expectimax: muito trabalhoso para MDPs

- Problema: estados são repetidos
 - Idéia: computar valores necessários apenas uma vez
- Problema: profundidade indefinida
 - Idéia: busca com limite de profundidade, mas com profundidades crescentes até que mudança seja pequena
 - Se $\gamma < 1$, níveis profundos da árvore são irrelevantes

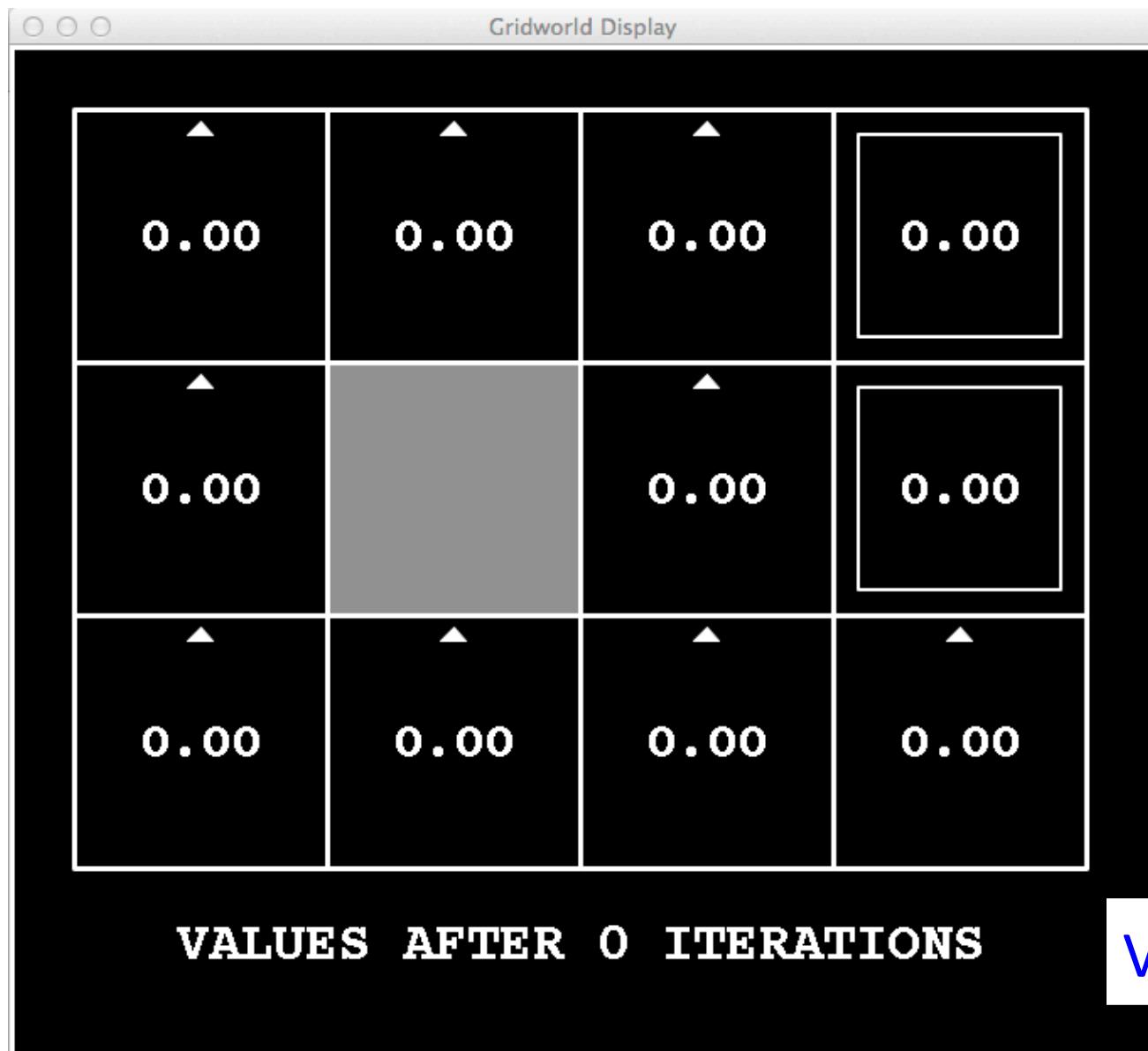


Idéia chave: Valores limitados no tempo

- Define-se $V_k(s)$ como valor ótimo de s se o jogo terminar em k mais passos de tempo
 - Equivalente a o que uma busca expectimax com profundidade limitada a k retornaria a partir de s



$k=0$



Ruído = 0.2
Desconto = 0.9
Recompensa (em
cada estado)= 0

$k=1$



Ruído = 0.2

Desconto = 0.9

Recompensa (em
cada estado)= 0

$k=2$

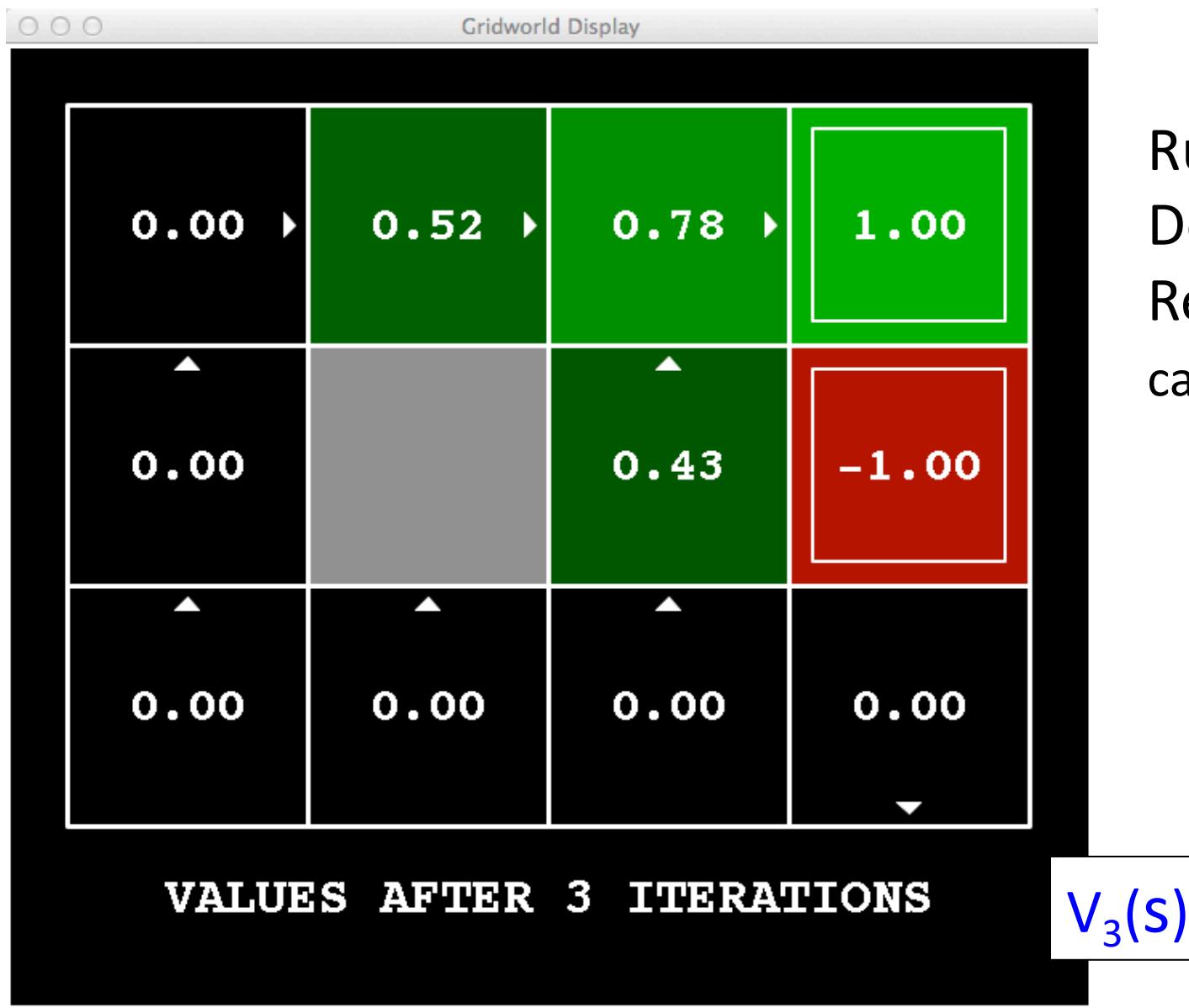


VALUES AFTER 2 ITERATIONS

$V_2(s)$

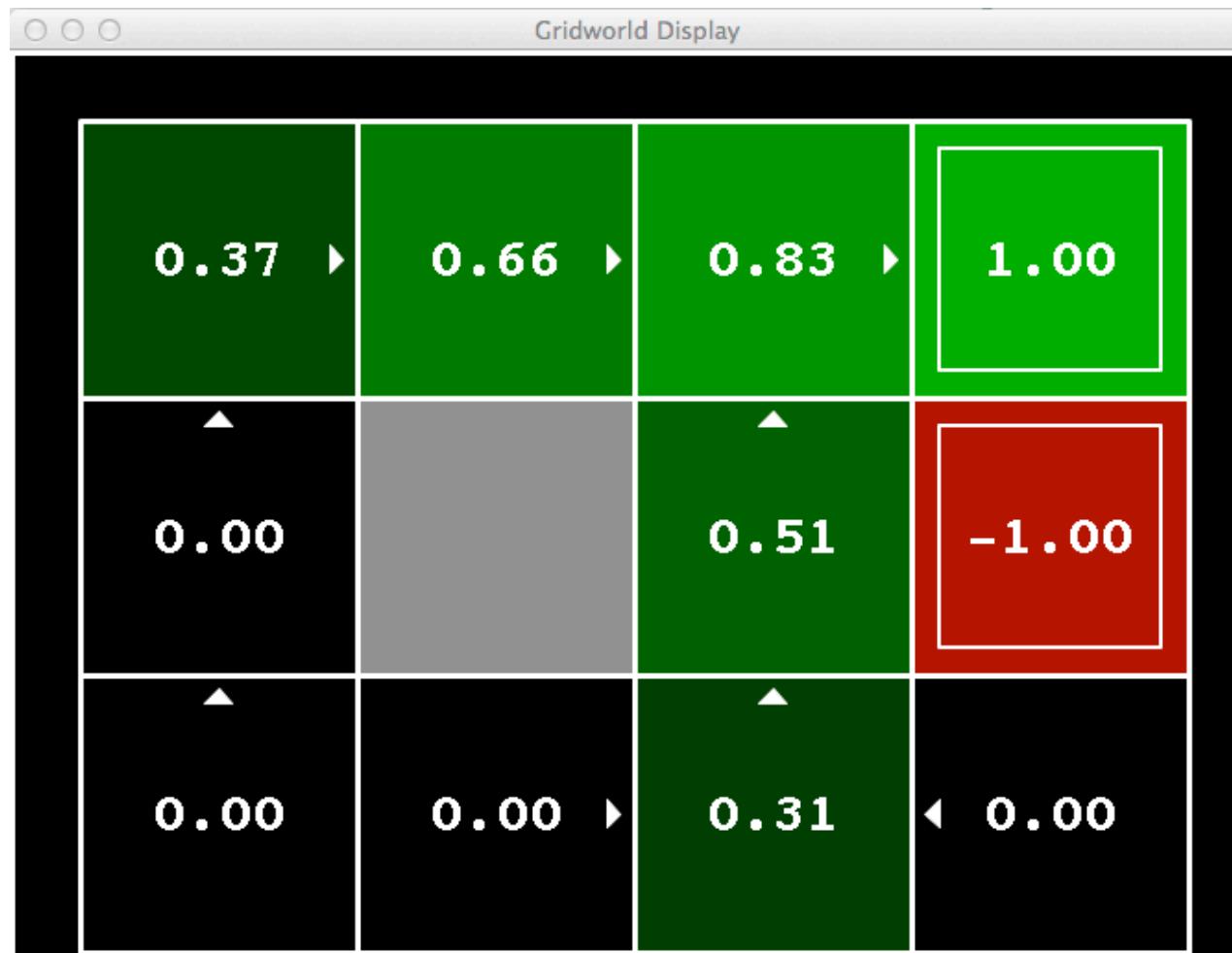
Ruído = 0.2
Desconto = 0.9
Recompensa (em
cada estado)= 0

$k=3$



Ruído = 0.2
Desconto = 0.9
Recompensa (em
cada estado)= 0

$k=4$



VALUES AFTER 4 ITERATIONS

$V_4(s)$

Ruído = 0.2
Desconto = 0.9
Recompensa (em
cada estado)= 0

$k=5$



Ruído = 0.2

Desconto = 0.9

Recompensa (em
cada estado)= 0

$k=6$

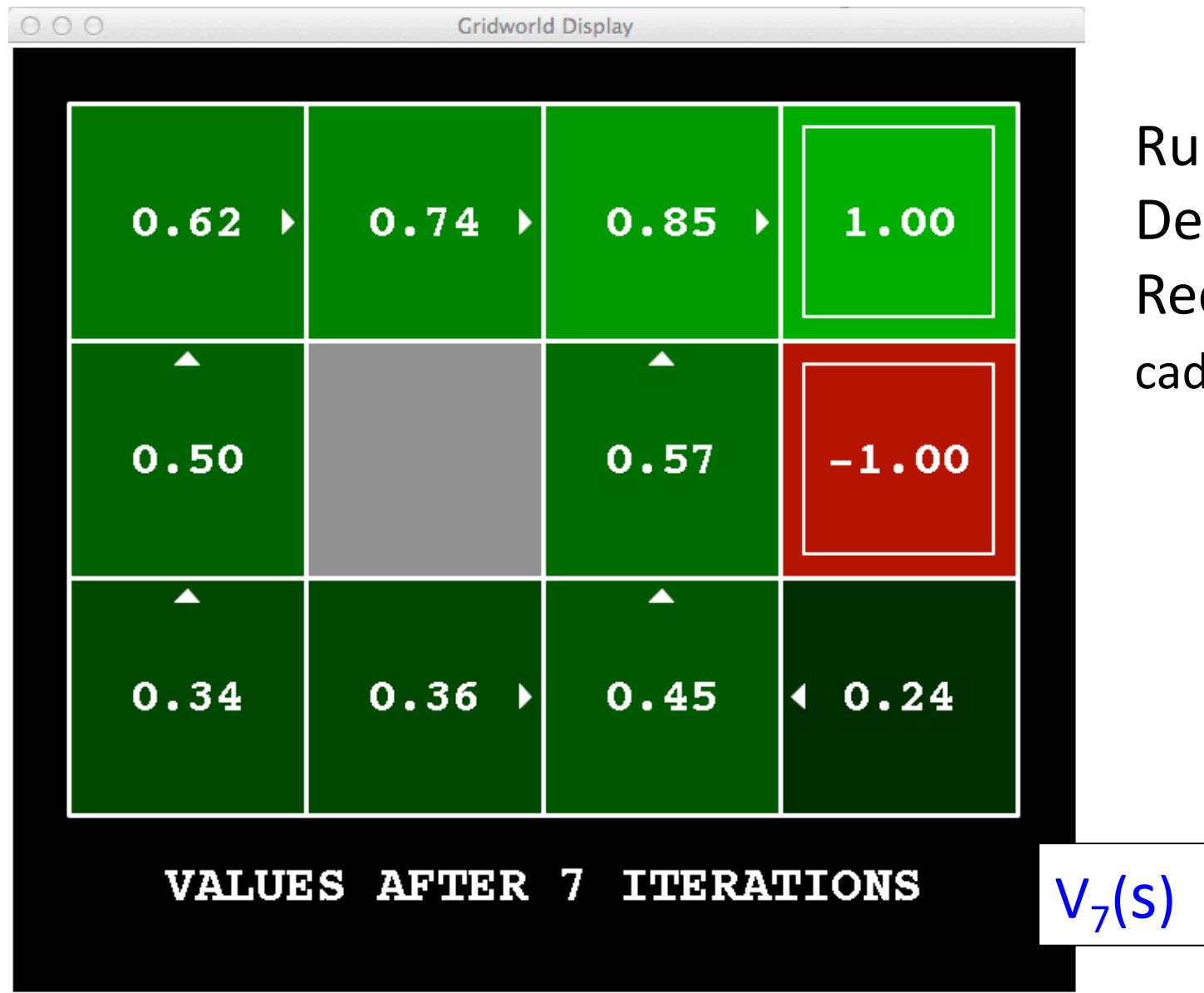


Ruído = 0.2

Desconto = 0.9

Recompensa (em
cada estado)= 0

$k=7$

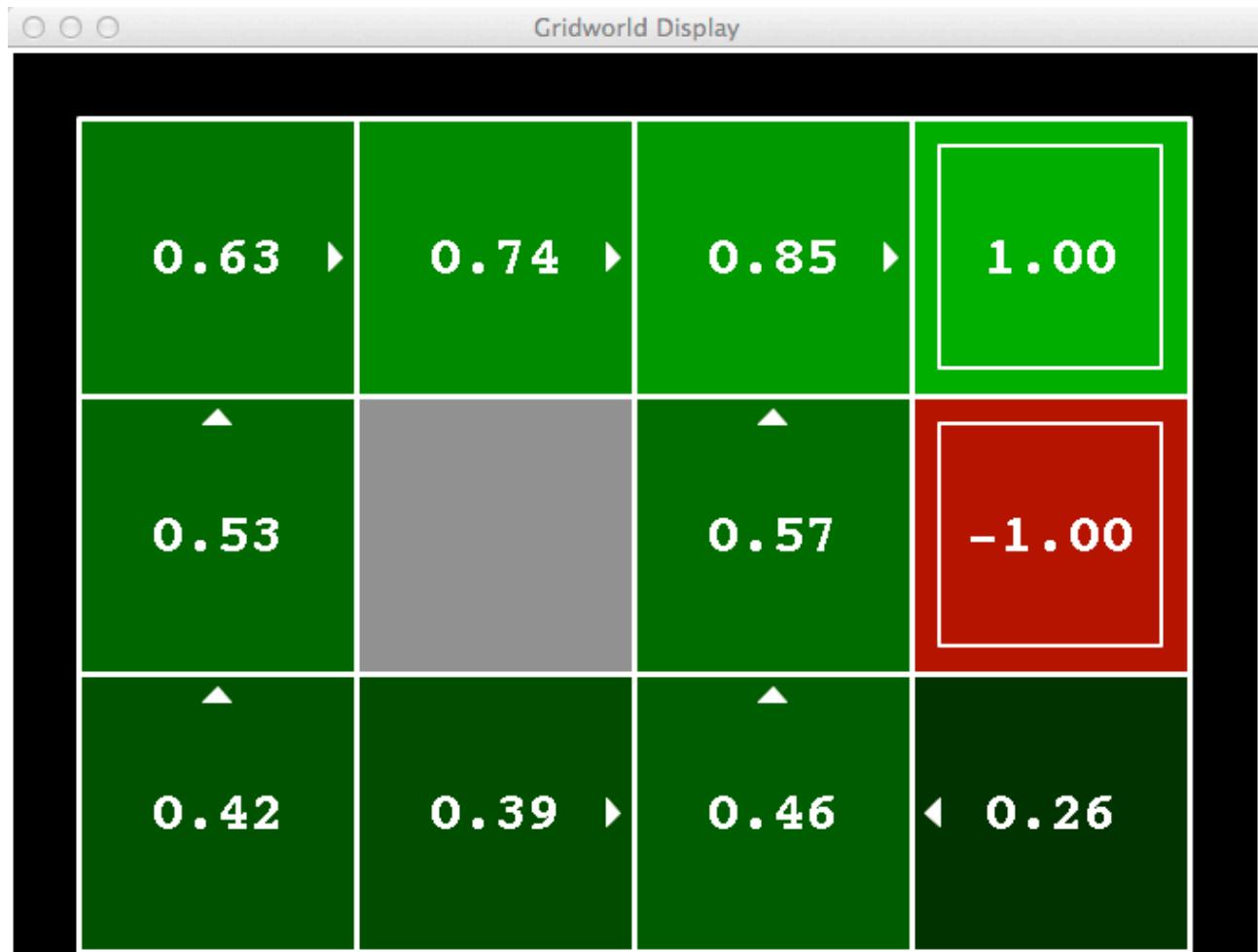


Ruído = 0.2

Desconto = 0.9

Recompensa (em
cada estado)= 0

$k=8$



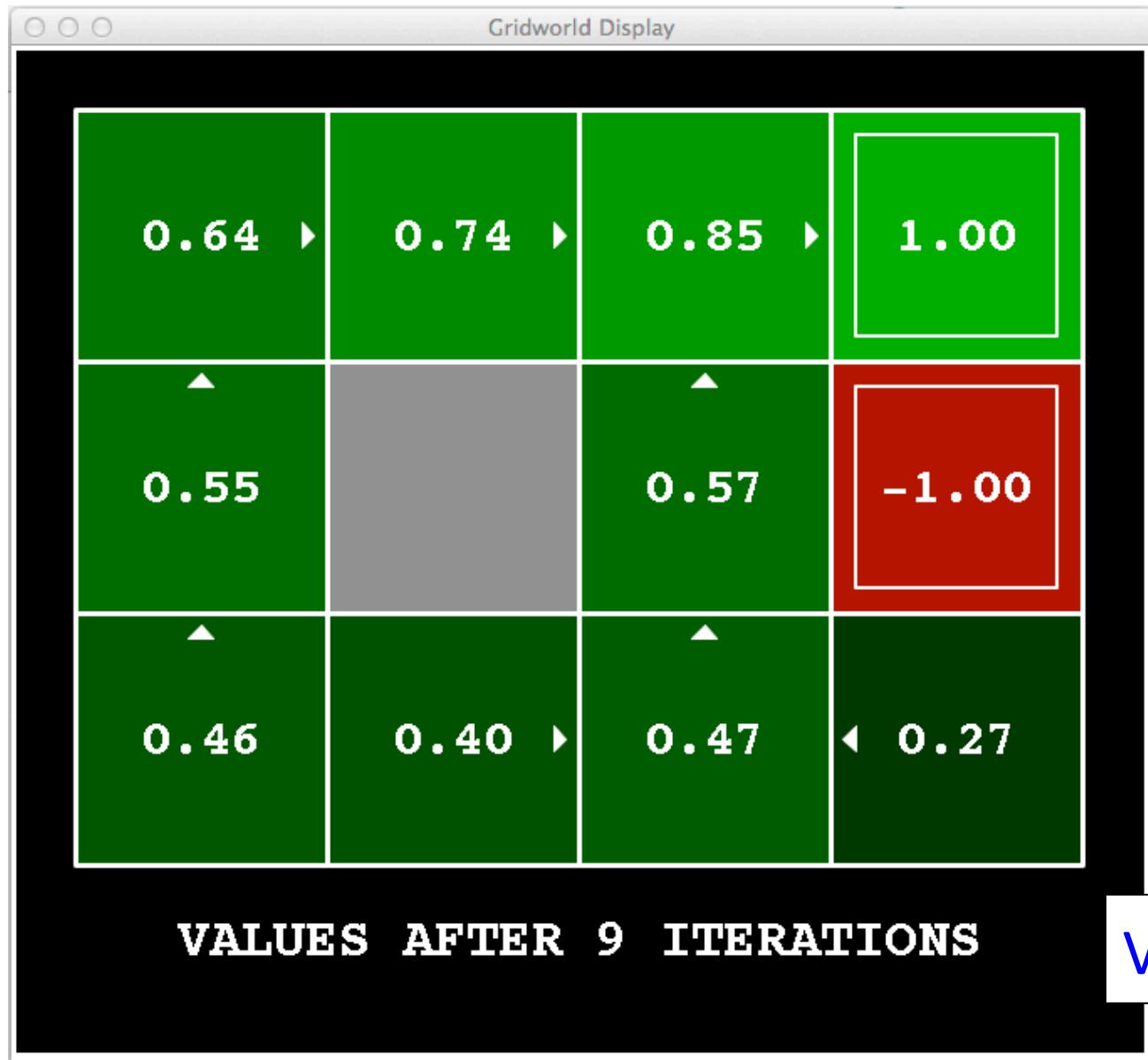
$V_8(s)$

Ruído = 0.2

Desconto = 0.9

Recompensa (em
cada estado)= 0

$k=9$

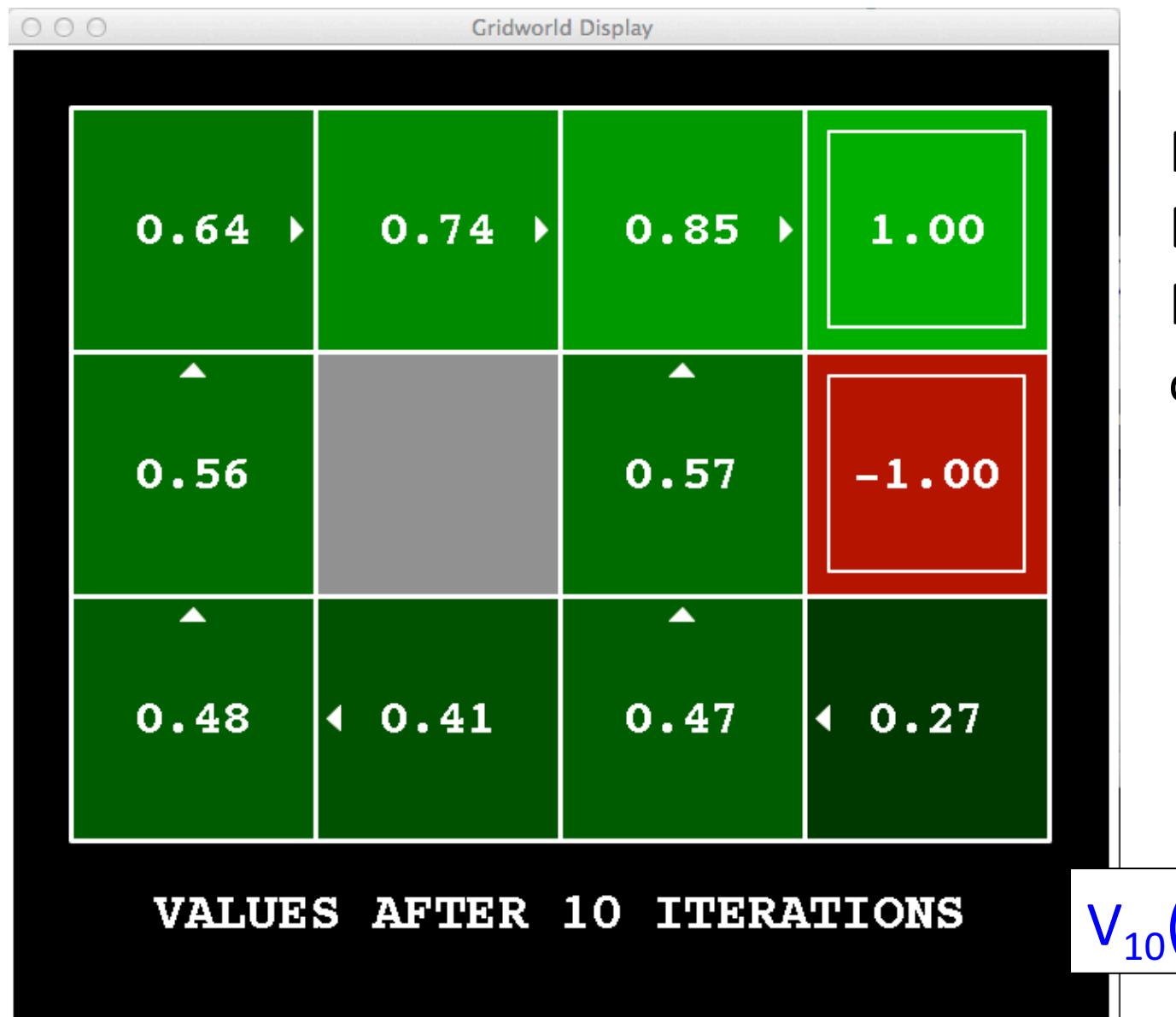


Ruído = 0.2

Desconto = 0.9

Recompensa (em
cada estado)= 0

$k=10$



Ruído = 0.2

Desconto = 0.9

Recompensa (em
cada estado)= 0

$k=11$



VALUES AFTER 11 ITERATIONS

$V_{11}(s)$

Ruído = 0.2
Desconto = 0.9
Recompensa (em
cada estado)= 0

$k=12$



VALUES AFTER 12 ITERATIONS

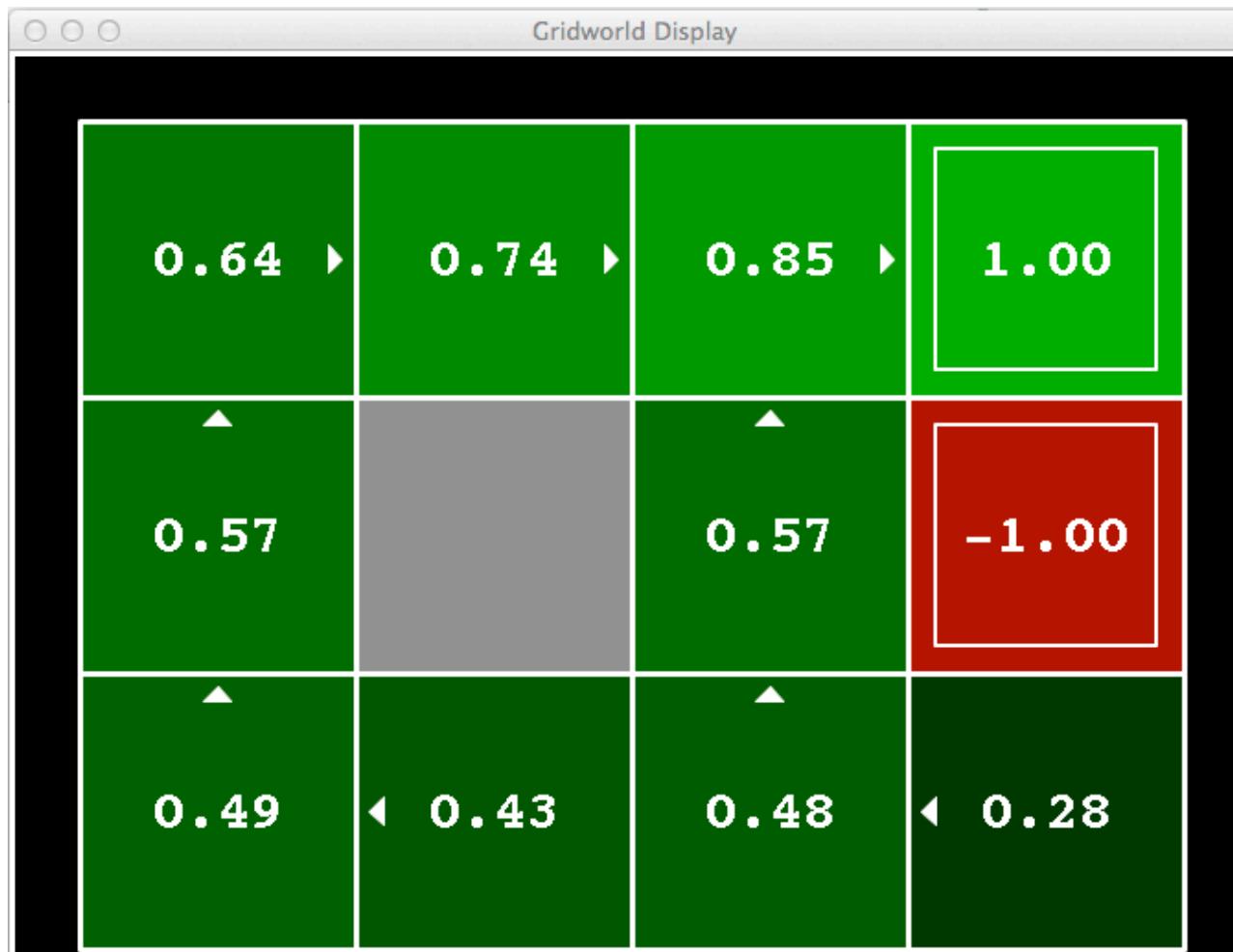
$V_{12}(s)$

Ruído = 0.2

Desconto = 0.9

Recompensa (em
cada estado)= 0

$k=100$



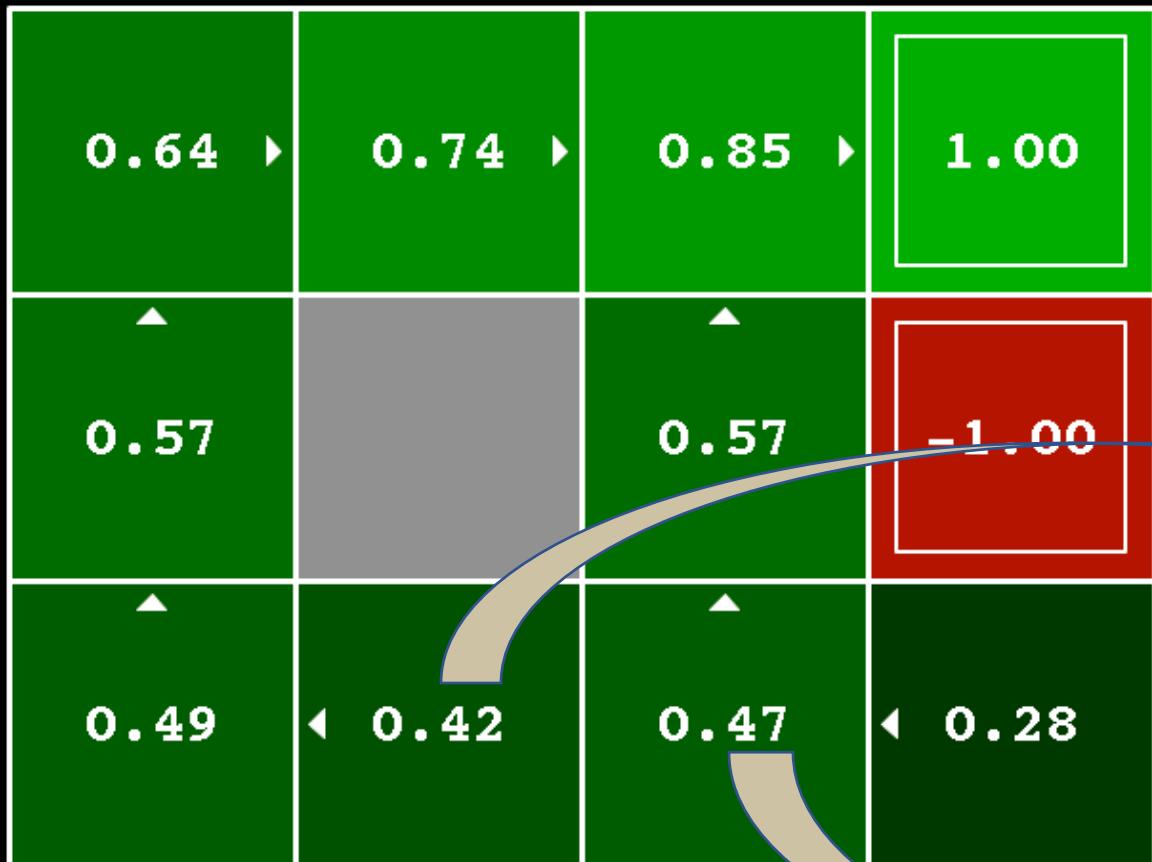
VALUES AFTER 100 ITERATIONS

$V_{100}(s)$

Ruído = 0.2
Desconto = 0.9
Recompensa (em
cada estado)= 0

$k=12$

Gridworld Display

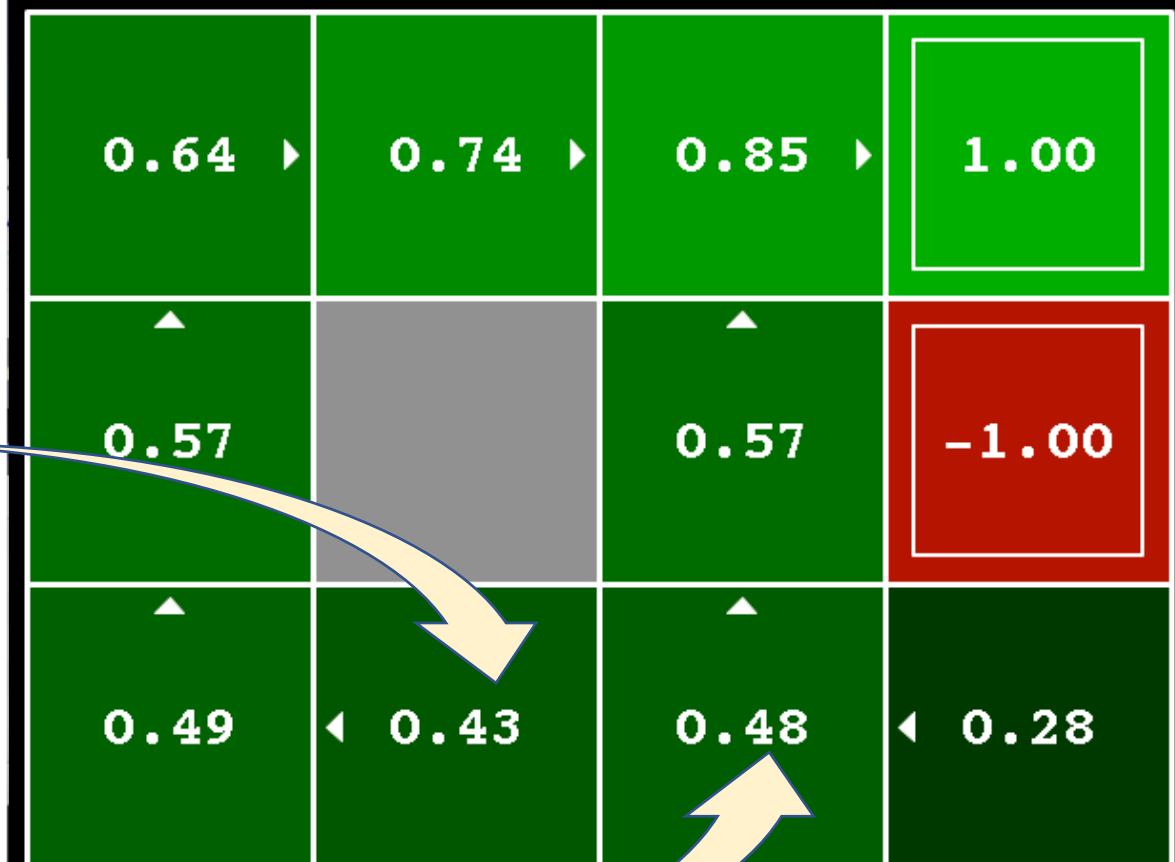


VALUES AFTER 12 ITERATIONS

$V_{12}(s)$

$k=100$

Gridworld Display

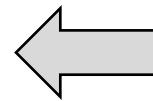


VALUES AFTER 100 ITERATIONS

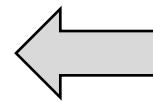
$V_{100}(s)$

Idéia chave: Valores limitados no tempo

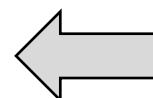
$$V_4(\text{blue car}) \quad V_4(\text{red car}) \quad V_4(\text{crashed car})$$



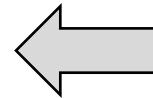
$$V_3(\text{blue car}) \quad V_3(\text{red car}) \quad V_3(\text{crashed car})$$



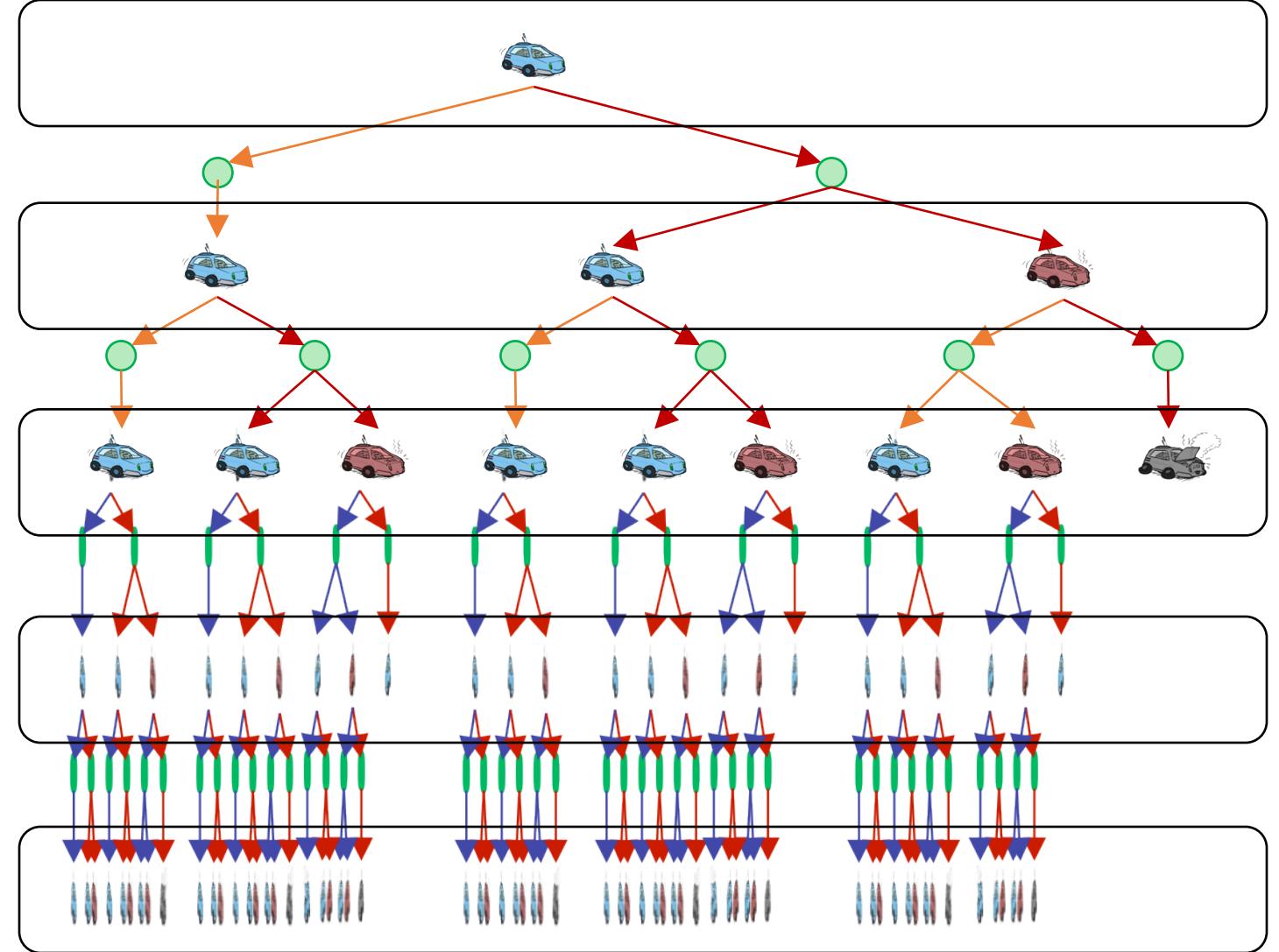
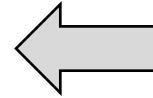
$$V_2(\text{blue car}) \quad V_2(\text{red car}) \quad V_2(\text{crashed car})$$



$$V_1(\text{blue car}) \quad V_1(\text{red car}) \quad V_1(\text{crashed car})$$

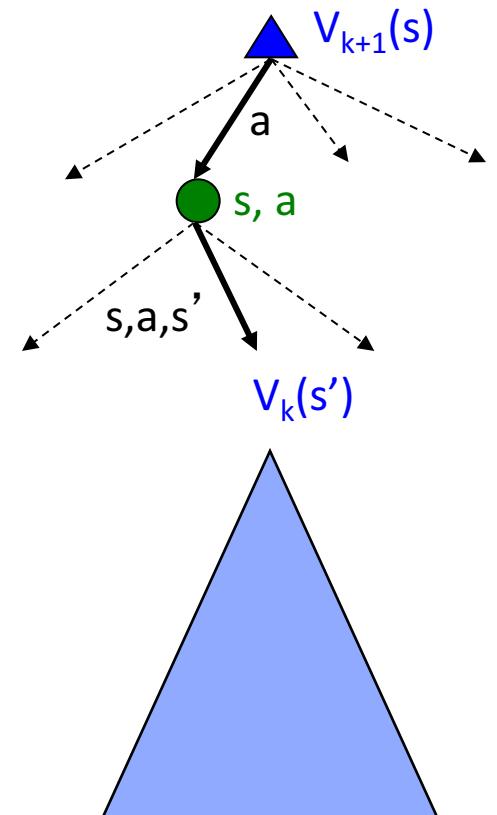


$$V_0(\text{blue car}) \quad V_0(\text{red car}) \quad V_0(\text{crashed car})$$



Algoritmo *Value Iteration*

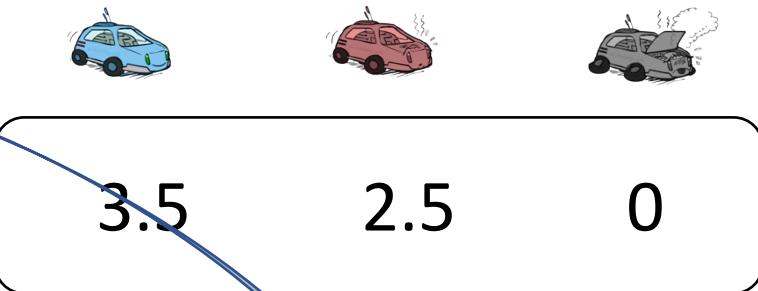
1. Começa $V_0(s) = 0 \rightarrow$ soma de recompensa esperada = zero
2. $V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$
3. Repete até convergir



$$V_2(\text{Cool}) = 1 * [+1 + 1 * 2] \text{ vs. } 0.5 * [+2 + 1 * 2] + 0.5 * [+2 + 1 * 1] = 3.5$$

$$V_2(\text{Warm}) = 0.5 * [+1 + 1 * 2] + 0.5 * [+1 + 1 * 1] \text{ vs. } 1 * [-10 + 1 * 0] = 2.5$$

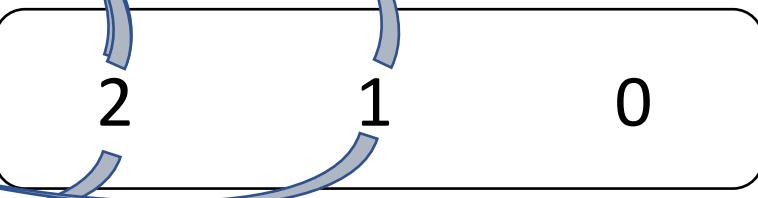
$$V_2(\text{Overheated}) = 0$$



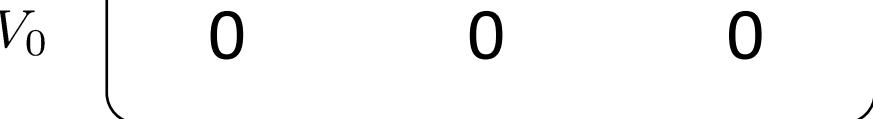
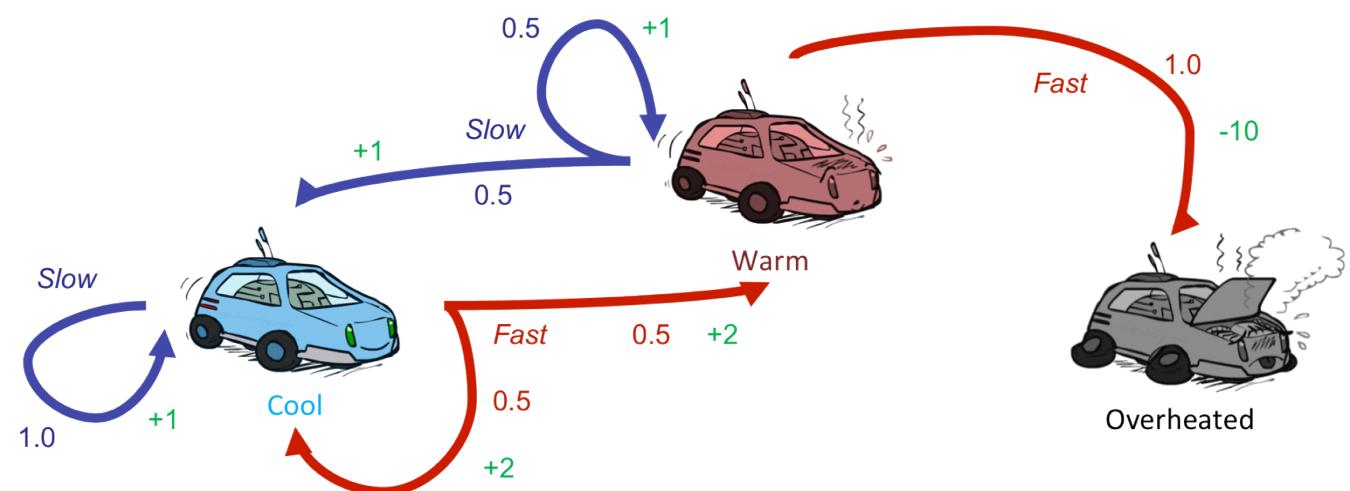
$$V_1(\text{Cool}) = 1 * [+1 + 1 * 0] \text{ vs. } 0.5 * [+2 + 1 * 0] + 0.5 * [+2 + 1 * 0] = 2$$

$$V_1(\text{Warm}) = 0.5 * [+1 + 1 * 0] + 0.5 * [+1 + 1 * 0] \text{ vs. } 1 * [-10 + 1 * 0] = 1$$

$$V_1(\text{Overheated}) = 0$$



$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') \left[R(s, a, s') + \gamma V_k(s') \right]$$



Considere $\gamma = 1$
(sem desconto)

Extração de política

- Considere que já computamos os valores ótimos $V^*(s)$
- Como devemos agir?



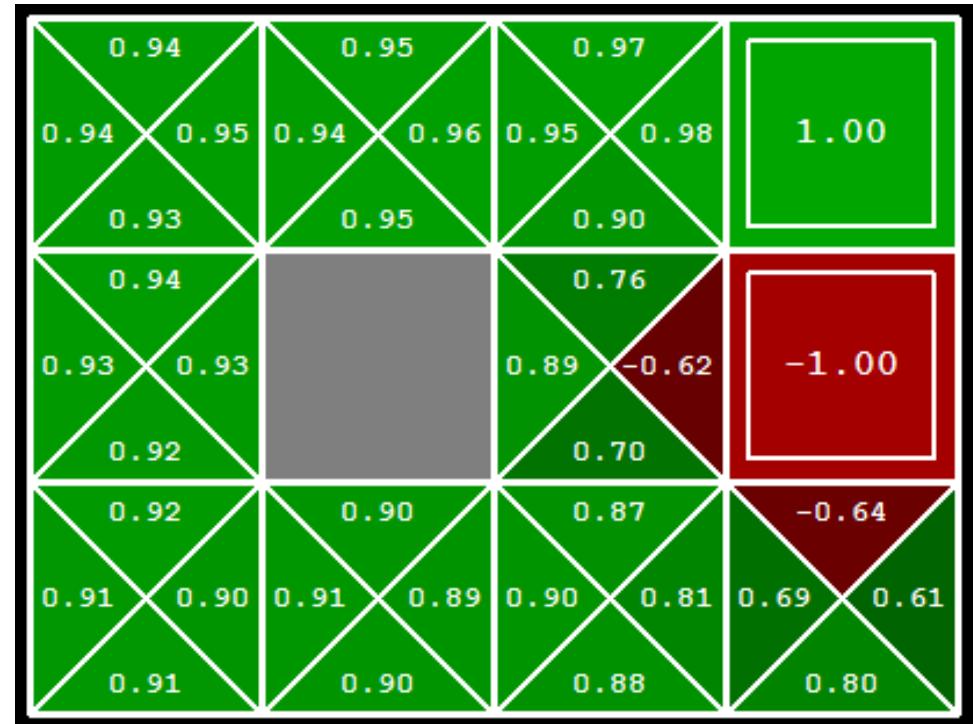
$$\pi^*(s) = \arg \max_a \sum_{s'} T(s, a, s')[R(s, a, s') + \gamma V^*(s')]$$

Calculando ações a partir de Q-Values

- Considere que temos os $Q^*(s,a)$ ótimos
- Como devemos agir?

$$\pi^*(s) = \arg \max_a Q^*(s, a)$$

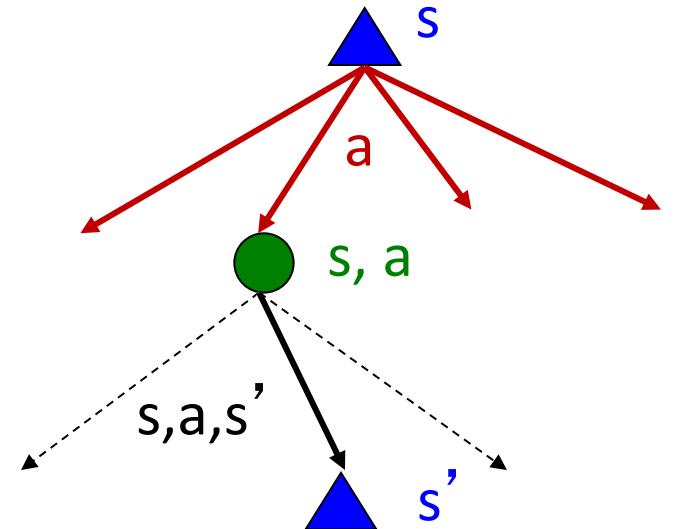
- **Importante:** é mais fácil computar ações a partir de Q^* do que de V^*



Problemas com *Value Iteration*

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

- Problema 1: Lentidão – $O(S^2A)$ por iteração
- Problema 2: A política usualmente converge bem antes dos valores



k=1



k=2



$k=12$



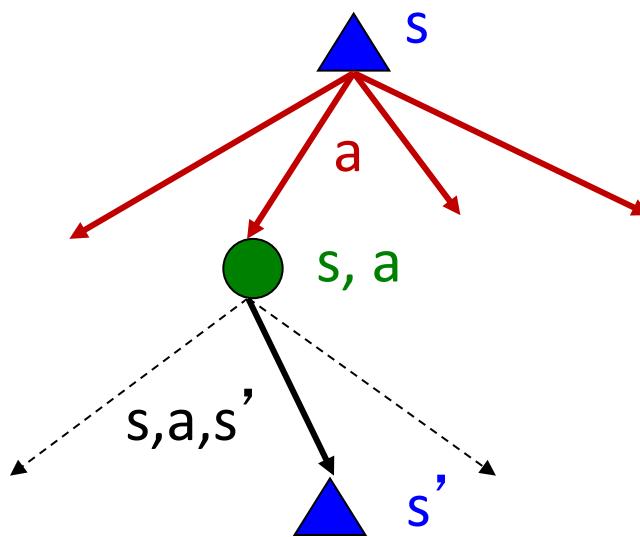
$k=100$



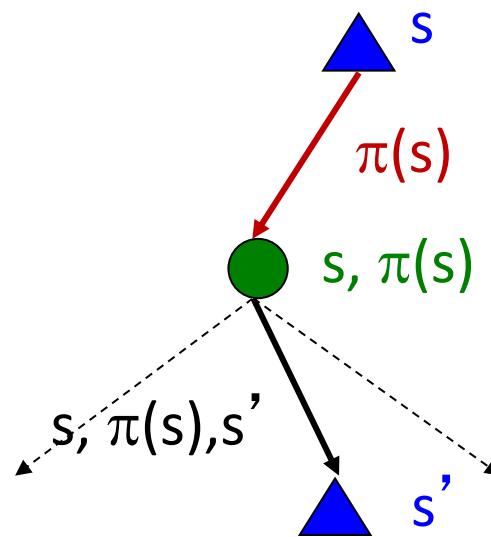
Uma saída: *Policy Iteration*

Preâmbulo: políticas fixas

Execute ação ótima



Faça o que diz π

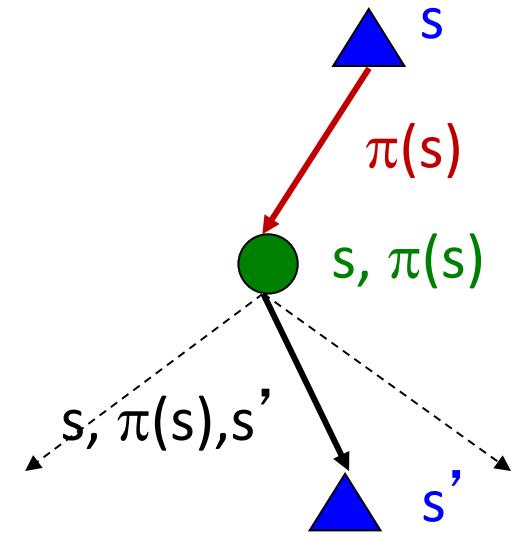


- Expectimax executa max sobre todas as ações para definir valores ótimos
- Se fixarmos alguma política $\pi(s)$, então a árvore seria mais simples – apenas uma única ação por estado

Preâmbulo: políticas fixas

- Recompensa total esperada partindo de s e seguindo política fixa π :

$$V^\pi(s) = \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V^\pi(s')]$$

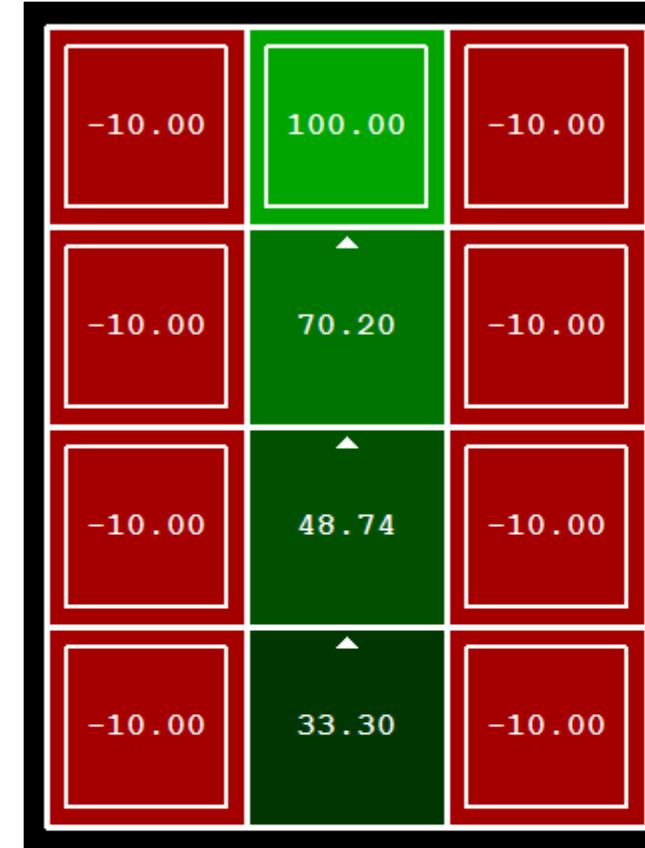


Avaliação de política

Sempre siga para o leste



Sempre siga pra o norte

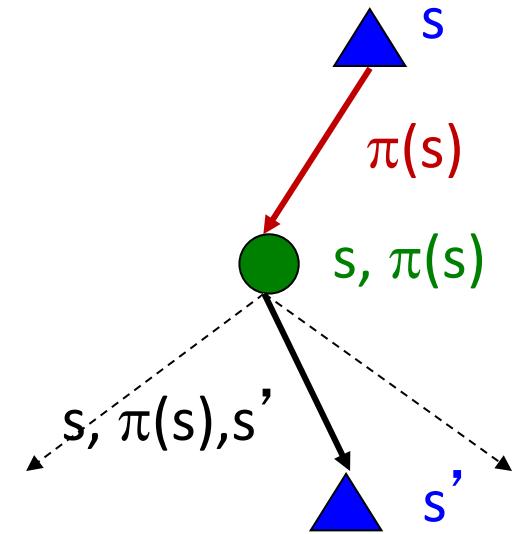


Avaliação de política

- Opção 1: Usar Bellman recursivamente para atualizações

$$V_0^\pi(s) = 0$$

$$V_{k+1}^\pi(s) \leftarrow \sum_{s'} T(s, \pi(s), s')[R(s, \pi(s), s') + \gamma V_k^\pi(s')]$$



- Opção 2: Sem os max, Bellman é apenas um sistema linear, que pode ser resolvido por frameworks mais otimizados

Algoritmo *Policy Iteration*

- 1. Avaliação: usa política fixa autal π e encontra V :
 - Itera até valores convergirem:

$$V_{k+1}^{\pi_i}(s) \leftarrow \sum_{s'} T(s, \pi_i(s), s') [R(s, \pi_i(s), s') + \gamma V_k^{\pi_i}(s')]$$

- 2. Melhoria: com os V fixos, computa uma melhor política usando extração
 - Um passo adiante:
- $\pi_{i+1}(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^{\pi_i}(s')]$
- 3. Repete passos 1 e 2 até convergência

Value Iteration vs. Policy Iteration

- Ambos computam a mesma coisa: valores ótimos!
- No **value iteration**:
 - Toda iteração atualiza V e (implicitamente) π
 - Não foca no cálculo da política, mas aplicando max sobre as ações implicitamente ela é recalculada
- No **policy iteration**:
 - Passos atualizam as utilidades com política fixa (cada passo é rápido porque apenas uma ação é considerada)
 - Após a política ser avaliada, uma nova é escolhida (lento como value iteration)
 - A nova política será melhor ou então encerra-se

Resumo

- Para...
 - Computar valores ótimos: usar value iteration OU policy iteration
 - Computar valores para uma política específica: usar policy evaluation
 - Transformar valores numa política: usar policy extraction

Q: Micro-Blackjack

Apenas três cartas em jogo: 2, 3 e 4. Retira uma carta do monte, guarda o valor e reembaralha. Se o valor acumulado for menor que 6 (seis), você pode pedir nova carta (Nova) ou parar (Parar). Caso contrário, você é obrigado a parar. Ao parar, sua utilidade é igual à soma das cartas somadas (até 5), ou 0 (zero) se você atingir 6 (seis) ou mais. Não há desconto ($\gamma = 1$).

1. Quais são os estados e ações desse MDP?
2. Qual é a função de transição (T) e de recompensa (R) desse MDP?
3. Qual a política ótima?

Q: Micro-Blackjack

1. Quais são os estados e ações desse MDP?
Estados = {0, 2, 3, 4, 5, Fim} e Ações = {Nova, Parar}
2. Qual é a função de transição (T) e de recompensa (R) desse MDP?
 $T(s, Parar, Fim) = 1$
 $T(0, Nova, 2) = 1/3, T(0, Nova, 3) = 1/3, T(0, Nova, 4) = 1/3$
 $T(2, Nova, 4) = 1/3, T(2, Nova, 5) = 1/3, T(2, Nova, Fim) = 1/3$
 $T(3, Nova, 5) = 1/3, T(3, Nova, Fim) = 2/3$
 $T(4, Nova, Fim) = 1$
 $T(5, Nova, Fim) = 1$
 $R(s, Parar, s') = s$ (para $s \leq 5$)
 $R(s, Parar, s') = 0$ (caso contrário)

Q: Micro-Blackjack

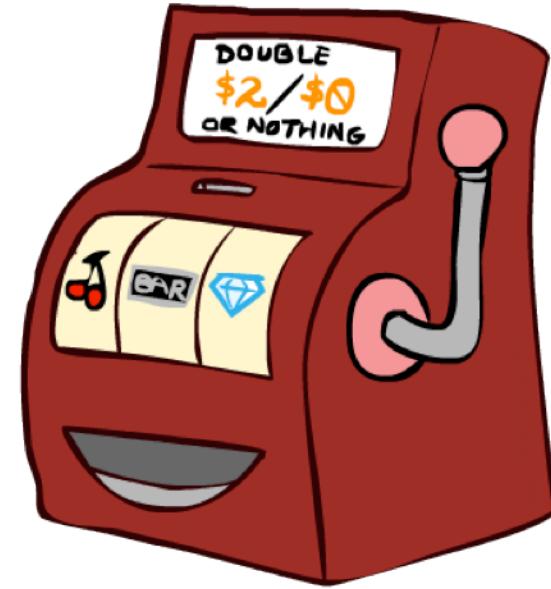
3. Qual a política ótima?

$$V_{k+1}(s) \leftarrow \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V_k(s')]$$

$$\pi^*(s) = \arg \max_a \sum_{s'} T(s, a, s') [R(s, a, s') + \gamma V^*(s')]$$

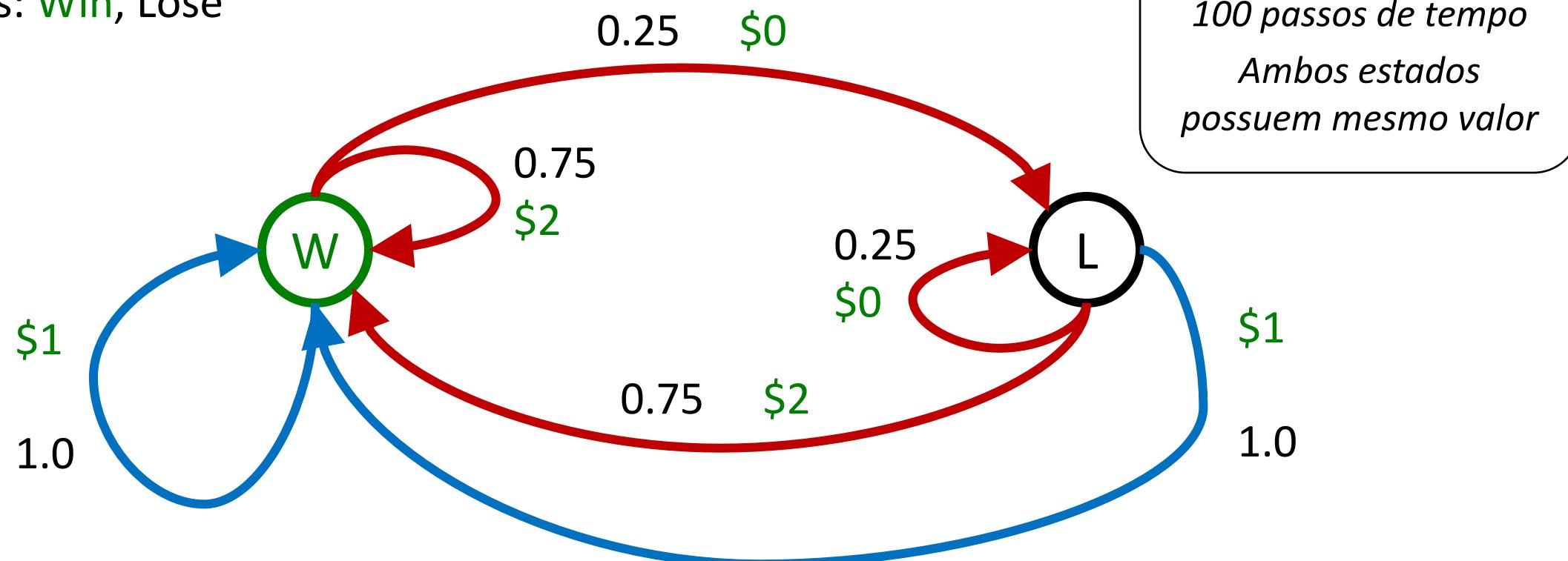
V	0	2	3	4	5	Fim
V_0	0	0	0	0	0	0
V_1	0	2	3	4	5	0
V_2	3	3	3	4	5	0
V_3	10/3	3	3	4	5	0
Extração de π	Nova	Nova	Parar	Parar	Parar	Parar

Finalmentes...



MDP

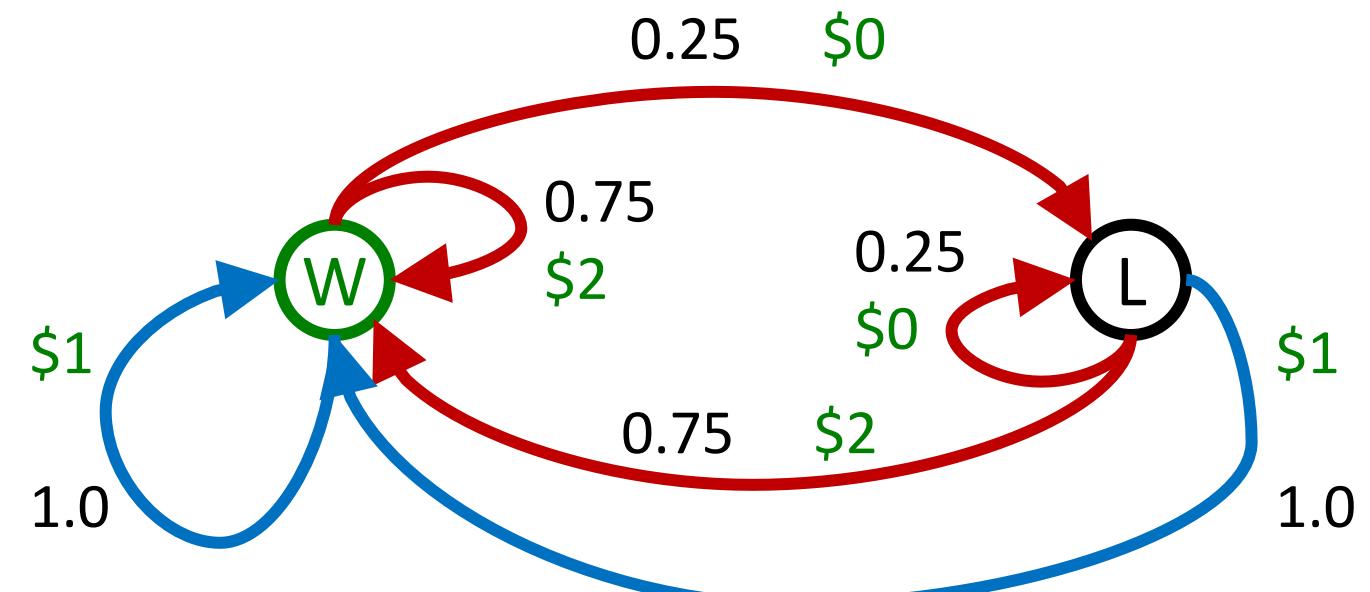
- Ações: *Blue, Red*
- Estados: *Win, Lose*



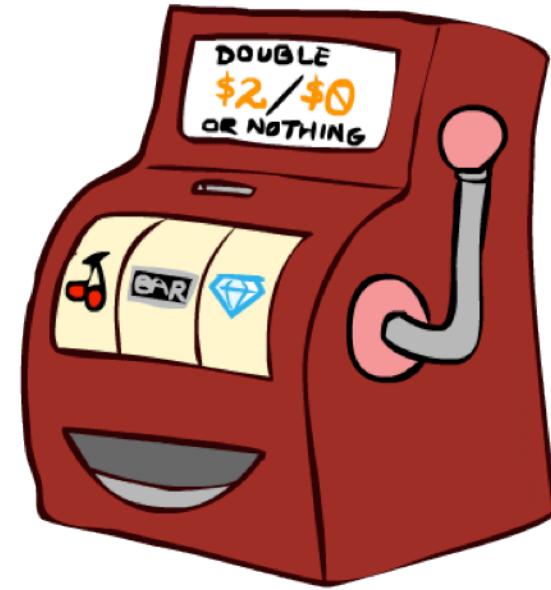
Solucionar MDPs → Planejamento *offline*

- Determina-se todos os valores através de computação
- Você precisa conhecer os detalhes do MDP
- Você não precisa realmente jogar o jogo!

	Valor
Play Red	150
Play Blue	100



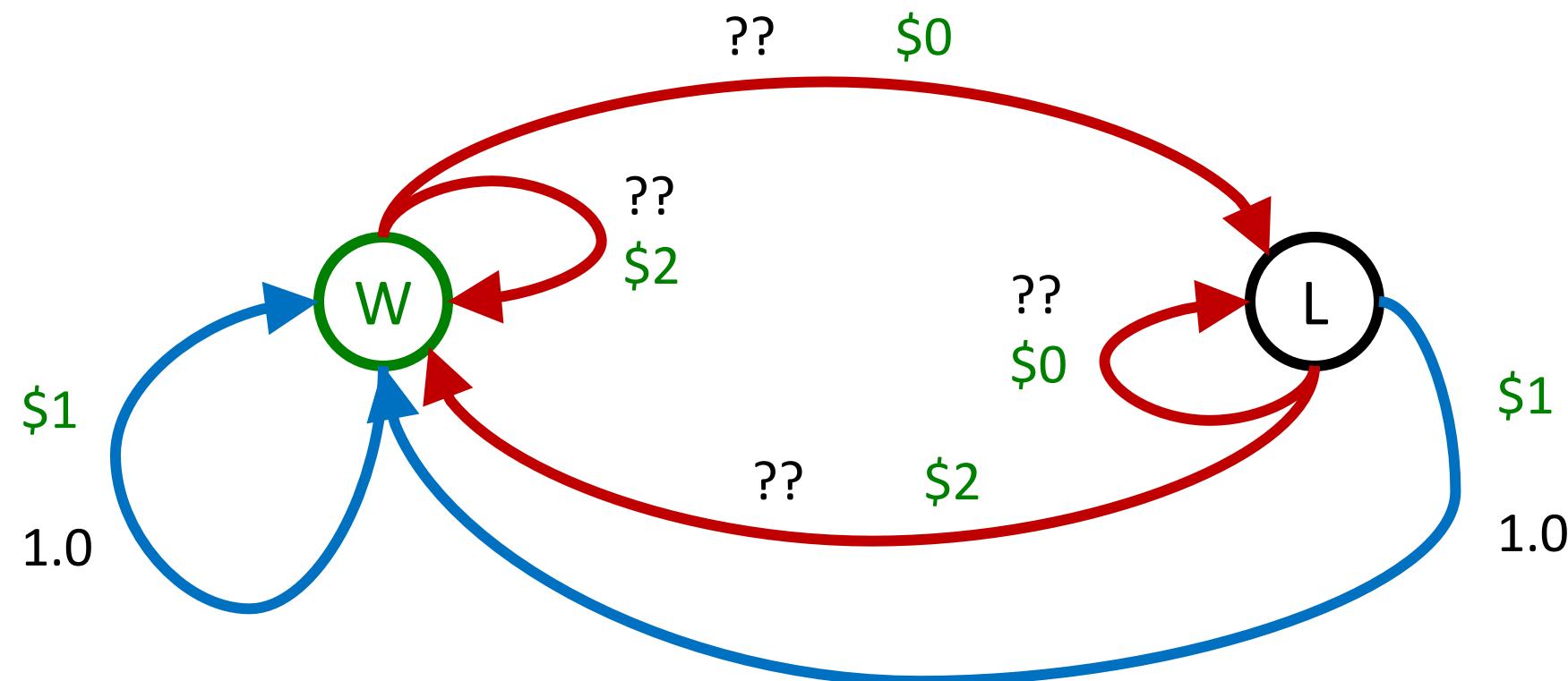
Jogando de fato...



\$2 \$2 \$0 \$2 \$2

\$2 \$2 \$0 \$0 \$0

Não se conhece probs. → Planejamento *online*



Jogando....



\$0	\$0	\$0	\$2	\$0
\$2	\$0	\$0	\$0	\$0

Planejamento vs. Aprendizado...

... por reforço!

- Tratava-se de um MDP, mas impossível de se resolver apenas com computação
- Necessário interagir para conhecê-lo melhor!
- Exploration vs. Exploitation





Hendrik Macedo

Escreve sobre Inteligência Artificial no Saense.

<http://www.saense.com.br/autores/artigos-publicados-por-hendrik-macedo/>