

# **APRENDIZAGEM POR REFORÇO**

Luís Morgado  
ISEL-ADEETC

# APRENDIZAGEM POR REFORÇO

- **PROBLEMAS**

- Eficiência do processo de aprendizagem
- Complexidade dos espaços de estados

- **SOLUÇÕES**

- Memória de experiência
- Utilização de modelos do mundo
- Generalização / Abstracção
  - Redes neuronais
  - Modelos hierárquicos
- Arquitecturas híbridas

# APRENDIZAGEM COM MEMÓRIA DE EXPERIÊNCIA

## *EXPERIENCE REPLAY*

- As experiências do agente são memorizadas numa **memória de experiência** (*replay memory*)
- Num determinado instante  $t$ , a experiência do agente é definida por um tuplo:

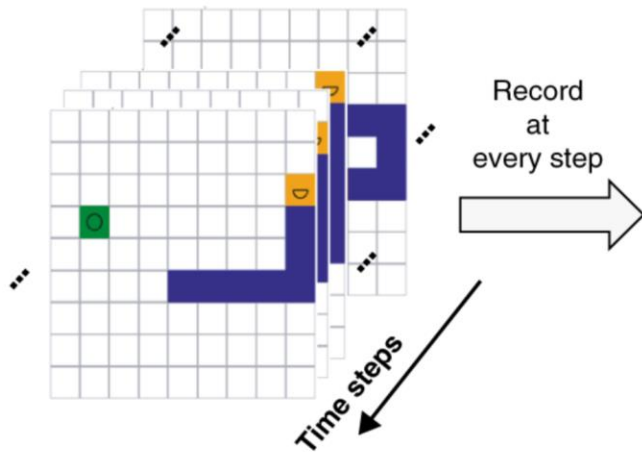
$$e_t = (s_t, a_t, r_{t+1}, s_{t+1})$$

- A memória de experiência é utilizada para simular a experiência real
- O objectivo é acelerar o processo de aprendizagem

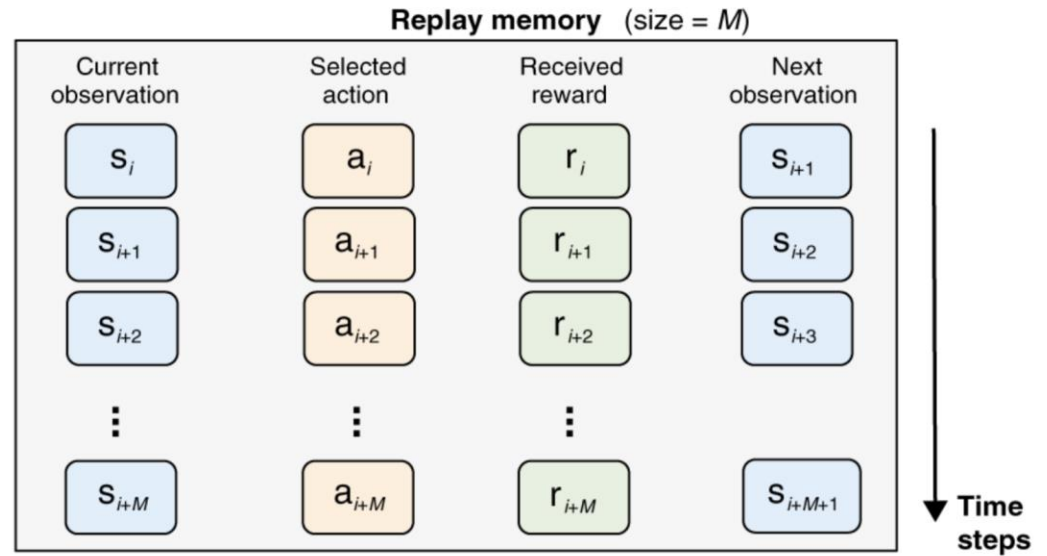
# MEMÓRIA DE EXPERIÊNCIA

## REPLAY MEMORY

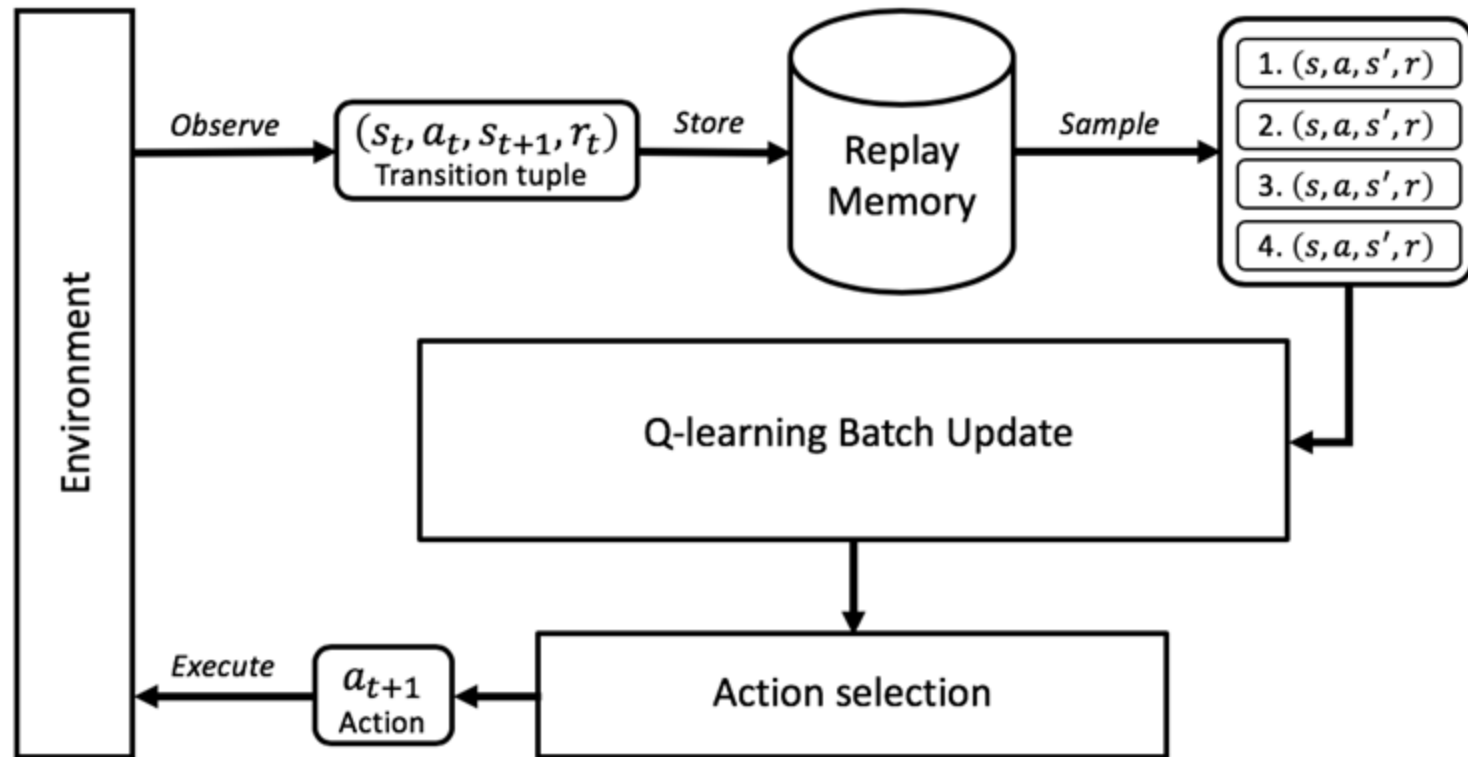
Experiência real



Memória de experiência



# APRENDIZAGEM COM MEMÓRIA DE EXPERIÊNCIA



# Q-LEARNING COM MEMÓRIA DE EXPERIÊNCIA

**Initialize**  $Q(s, a)$  for all  $s \in S$  and  $a \in A(s)$

Experience Memory  $\leftarrow \{\}$

**Do forever:**

$s \leftarrow$  current (nonterminal) state

$a \leftarrow \epsilon$ -greedy( $s, Q$ )

Take action  $a$ ; observe resultant reward  $r$  and state  $s'$

$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$

$e_t = (s, a, r, s')$

Update Experience Memory for experience  $e_t$

**Repeat  $n$  times:**

$s, a, r, s' \leftarrow$  sample from Experience Memory

$Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$

# APRENDIZAGEM COM MEMÓRIA DE EXPERIÊNCIA

- Aspectos a considerar
  - **Dimensão da memória de experiência**
    - Necessário limitar
      - Remover experiências mais antigas
      - Remover por critério de menor relevância
  - **Seleção de amostras**
    - Aleatória (distribuição uniforme)
    - Por critério de relevância
      - Amostragem com prioridade

# APRENDIZAGEM COM MODELOS

## *MODEL-BASED REINFORCEMENT LEARNING*

- Aprendizagem de um modelo do mundo
  - **Modelo do mundo:** Representação das características relevantes do domínio do problema
    - **Estrutura** (estado)
    - **Dinâmica** (transição de estado)
    - **Valor** (recompensa)
- Modelo do mundo
  - Utilizado para simular experiência
  - Suporta planeamento
- Necessidade de manter o modelo do mundo actualizado e consistente com a experiência



# MODELO DO MUNDO

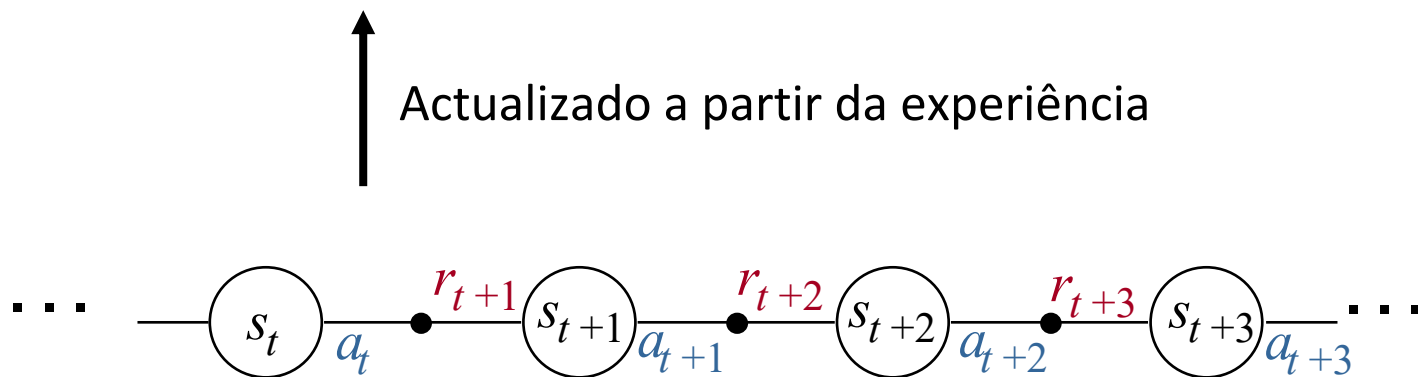
## REPRESENTAÇÃO DO MUNDO

$S$  – conjunto de estados do mundo

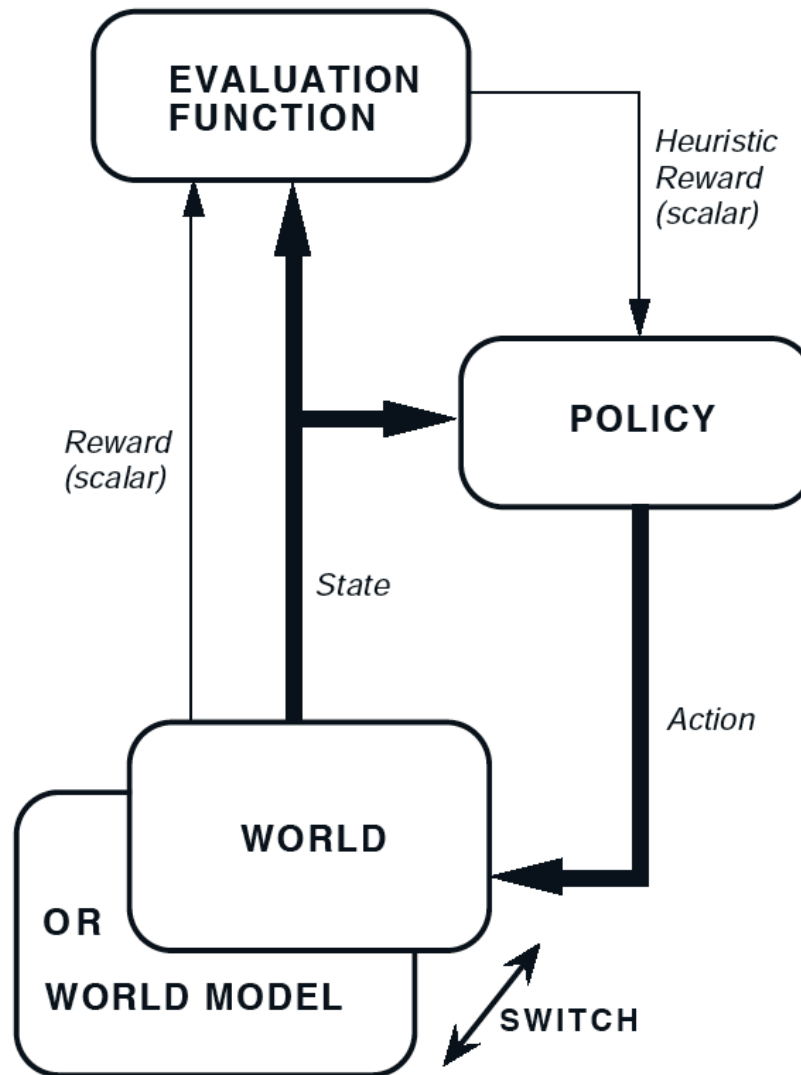
$A(s)$  – conjunto de acções possíveis no estado  $s \in S$

$T(s, a, s')$  – probabilidade de transição de  $s$  para  $s'$  através de  $a$

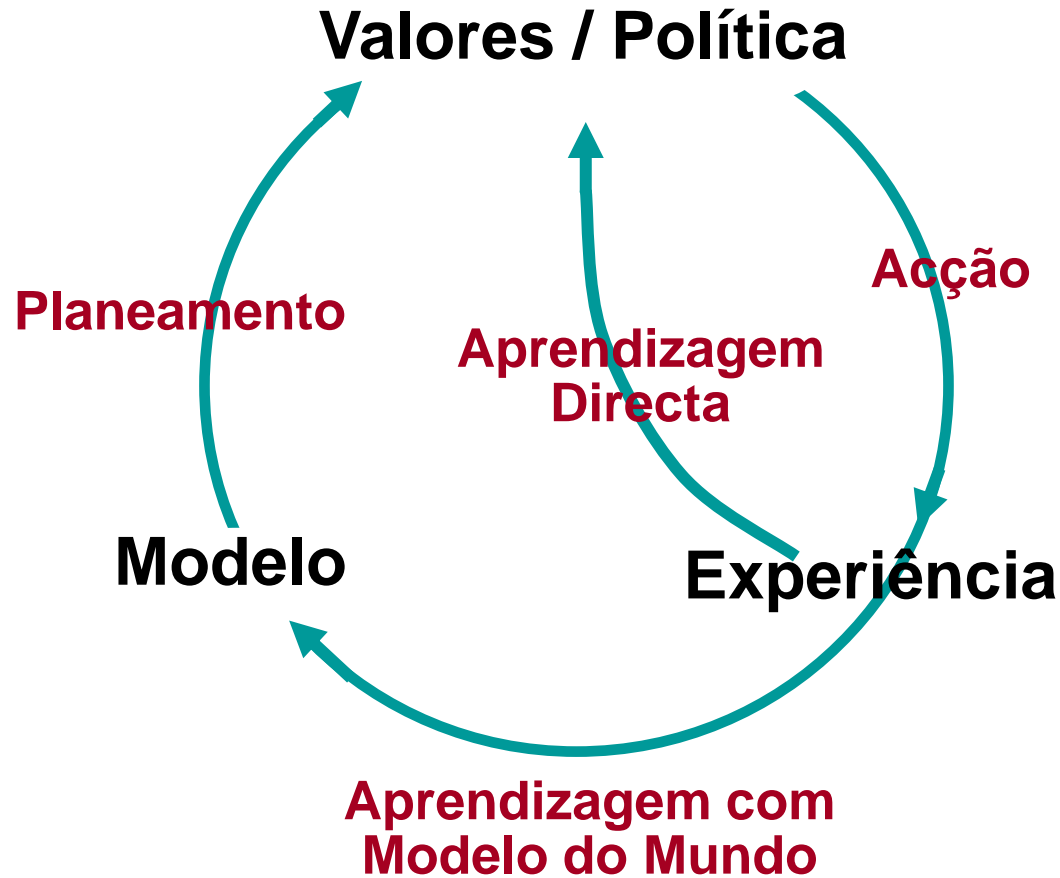
$R(s, a, s')$  – retorno esperado na transição de  $s$  para  $s'$  através de  $a$



# APRENDIZAGEM COM MODELO DO MUNDO



# APRENDIZAGEM E PLANEAMENTO



# APRENDIZAGEM E PLANEAMENTO

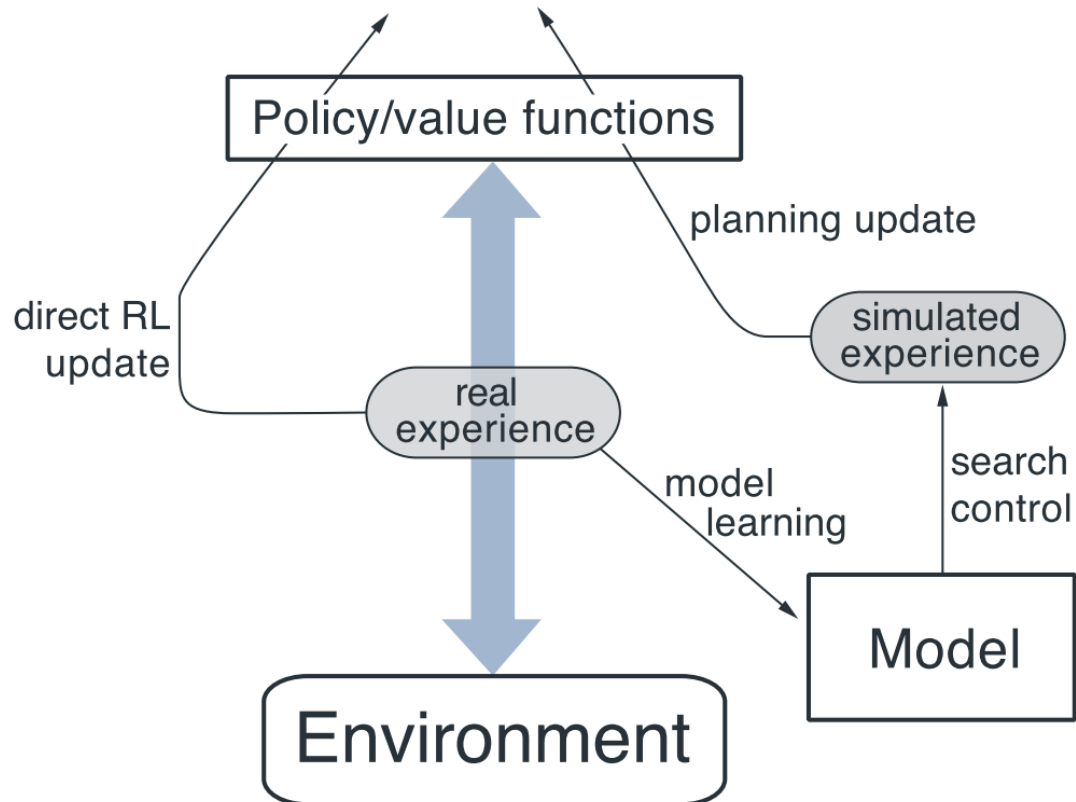
- **EXPERIÊNCIA**

- Aprendizagem de modelos do ambiente
- Aprendizagem de valores e políticas de acção

- **PLANEAMENTO**

- Simulação da experiência
- Aprendizagem indirecta a partir de modelos internos
- Iteração de comportamentos, não possível directamente no ambiente

# ARQUITECTURA DYNA



[Sutton & Barto, 2020]

# ALGORITMO DYNA-Q

## AMBIENTES DETERMINISTAS

$$Model(s,a) = \begin{cases} S \\ A \\ \hat{T}(s,a) \\ \hat{R}(s,a) \end{cases}$$
$$T: S \times A \rightarrow S$$
$$R: S \times A \rightarrow \mathbb{R}$$
$$\hat{T}(s,a) = s'$$
$$\hat{R}(s,a) = r$$

## AMBIENTES NÃO DETERMINISTAS

$$Model(s,a) = \begin{cases} S \\ A \\ \hat{T}(s,a,s') \\ \hat{R}(s,a) \end{cases}$$
$$T: S \times A \times S \rightarrow [0,1]$$
$$\hat{T}(s,a,s') = \frac{\text{Number of } (s, a, s') \text{ experiences}}{\text{Number of } (s, a) \text{ experiences}}$$
$$\hat{R}(s,a) = \frac{\text{Sum of rewards received by taking action } a \text{ in state } s}{\text{Number of } (s, a) \text{ experiences}}$$

# ALGORITMO DYNA-Q

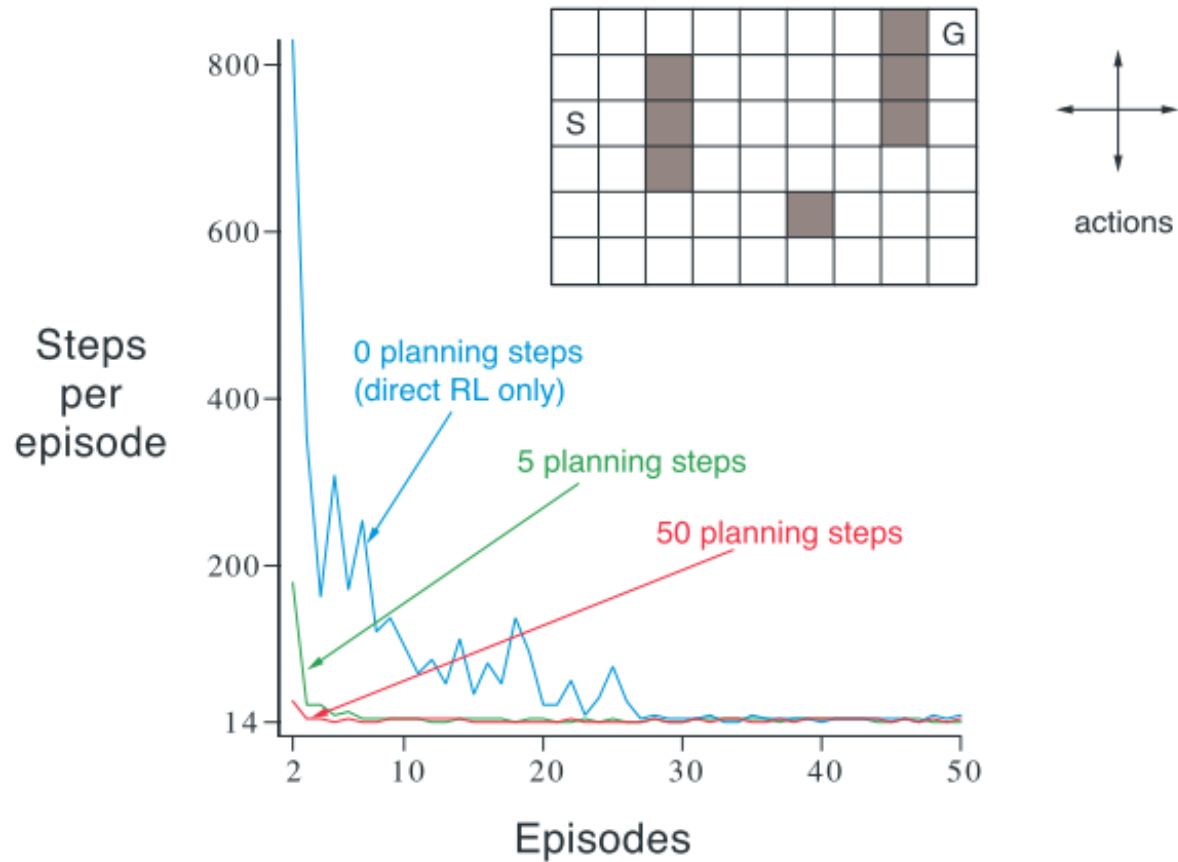
Initialize  $Q(s, a)$  and  $Model(s, a)$  for all  $s \in \mathcal{S}$  and  $a \in \mathcal{A}(s)$

Do forever:

- (a)  $s \leftarrow$  current (nonterminal) state
- (b)  $a \leftarrow \varepsilon$ -greedy( $s, Q$ )
- (c) Execute action  $a$ ; observe resultant state,  $s'$ , and reward,  $r$
- (d)  $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$
- (e)  $Model(s, a) \leftarrow s', r$  (assuming deterministic environment)
- (f) Repeat  $N$  times:
  - $s \leftarrow$  random previously observed state
  - $a \leftarrow$  random action previously taken in  $s$
  - $s', r \leftarrow Model(s, a)$
  - $Q(s, a) \leftarrow Q(s, a) + \alpha[r + \gamma \max_{a'} Q(s', a') - Q(s, a)]$

# ALGORITMO DYNA-Q

## EXEMPLO

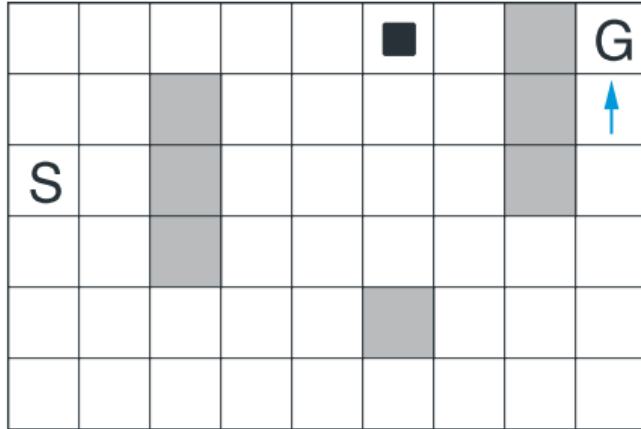




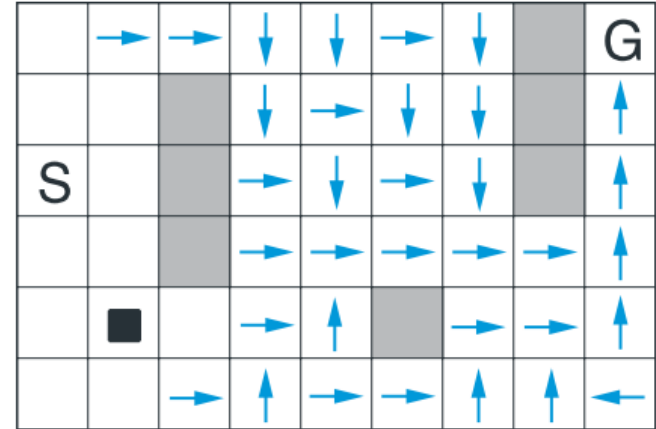
# ALGORITMO DYNA-Q

## EXEMPLO

WITHOUT PLANNING ( $n=0$ )



WITH PLANNING ( $n=50$ )



# AMOSTRAGEM POR PRIORIDADE

Varrimento com prioridade (*Prioritized Sweeping*)

- **Aumentar a eficiência do planeamento**
  - Simulações e actualizações de valor de pares estado-acção específicos
  - Prioridade a pares estado-acção com grande variação de valor
  - Actualização de valor retrospectiva
    - Pares estado-acção que levam a outros pares estado-acção com grande variação de valor
    - Varrimento de encadeamentos de transições

# AMOSTRAGEM POR PRIORIDADE

## Varrimento com prioridade (*Prioritized Sweeping*)

### Prioritized sweeping for a deterministic environment

Initialize  $Q(s, a)$ ,  $Model(s, a)$ , for all  $s, a$ , and  $PQueue$  to empty

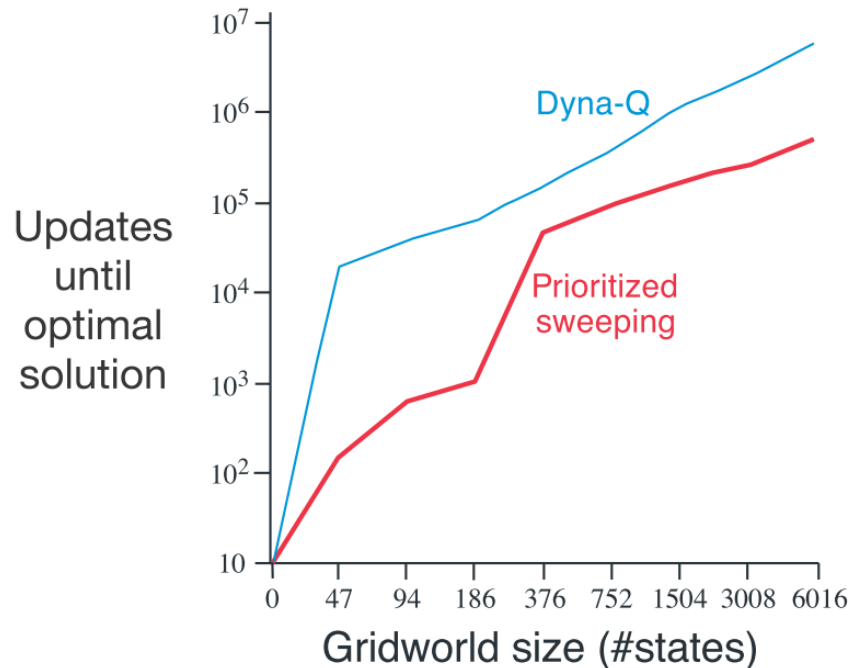
Loop forever:

- (a)  $S \leftarrow$  current (nonterminal) state
- (b)  $A \leftarrow policy(S, Q)$
- (c) Take action  $A$ ; observe resultant reward,  $R$ , and state,  $S'$
- (d)  $Model(S, A) \leftarrow R, S'$
- (e)  $P \leftarrow |R + \gamma \max_a Q(S', a) - Q(S, A)|$ .
- (f) if  $P > \theta$ , then insert  $S, A$  into  $PQueue$  with priority  $P$
- (g) Loop repeat  $n$  times, while  $PQueue$  is not empty:
  - $S, A \leftarrow first(PQueue)$
  - $R, S' \leftarrow Model(S, A)$
  - $Q(S, A) \leftarrow Q(S, A) + \alpha [R + \gamma \max_a Q(S', a) - Q(S, A)]$
  - Loop for all  $\bar{S}, \bar{A}$  predicted to lead to  $S$ :
    - $\bar{R} \leftarrow$  predicted reward for  $\bar{S}, \bar{A}, S$
    - $P \leftarrow |\bar{R} + \gamma \max_a Q(S, a) - Q(\bar{S}, \bar{A})|$ .
    - if  $P > \theta$  then insert  $\bar{S}, \bar{A}$  into  $PQueue$  with priority  $P$

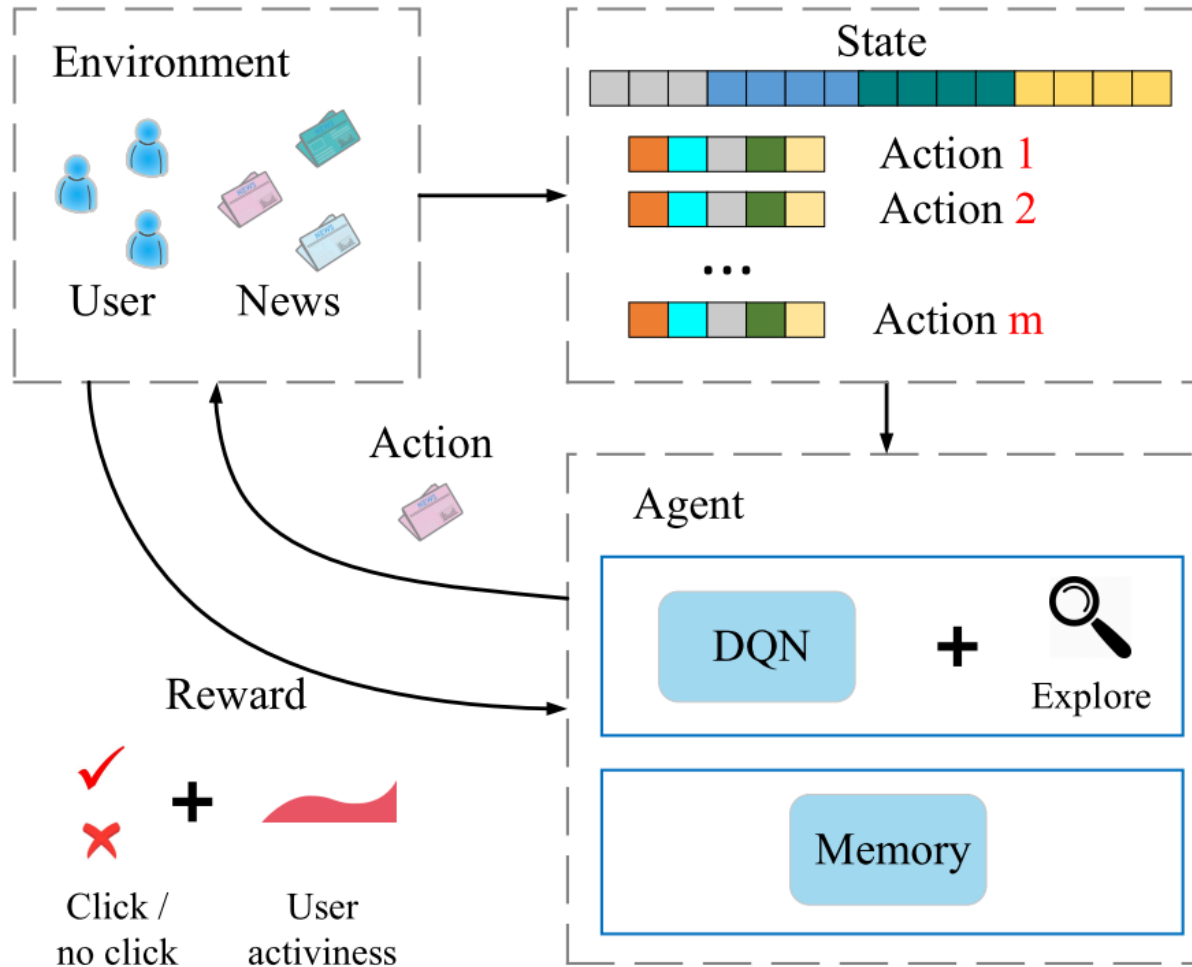
# AMOSTRAGEM POR PRIORIDADE

## Varrimento com prioridade (*Prioritized Sweeping*)

### Exemplo



# Sistema de recomendação de notícias



# REFERÊNCIAS

[Russel & Norvig, 2003]

S. Russell, P. Norvig, “Artificial Intelligence: A Modern Approach”, 2nd Edition, Prentice Hall, 2003

[Sutton & Barto, 1998]

R. Sutton, A. Barto, “Reinforcement Learning: An Introduction”, MIT Press, 1998

[Poole & Mackworth, 2010]

D. Poole, A. Mackworth, “Artificial Intelligence: Foundations of Computational Agents”, Cambridge University Press, 2010

[Raghavan *et al.*, 2019]

A. Raghavan, J. Hostetler, S. Chai, “Generative Memory for Lifelong Reinforcement Learning”, 2019

[Cai *et al.*, 2020]

S. Cai, S. Bileschi, E. Nielsen, F. Chollet, “Deep Learning with JavaScript: Neural networks in TensorFlow.js”, Manning Publications, 2020

[Zheng *et al.*, 2018]

G. Zheng, F. Zhang, Z. Zheng, Y. Xiang, N. Yuan, X. Xie, Z. Li, “DRN: A Deep Reinforcement Learning Framework for News Recommendation”, WWW '18: Proceedings of the World Wide Web Conference, 2018