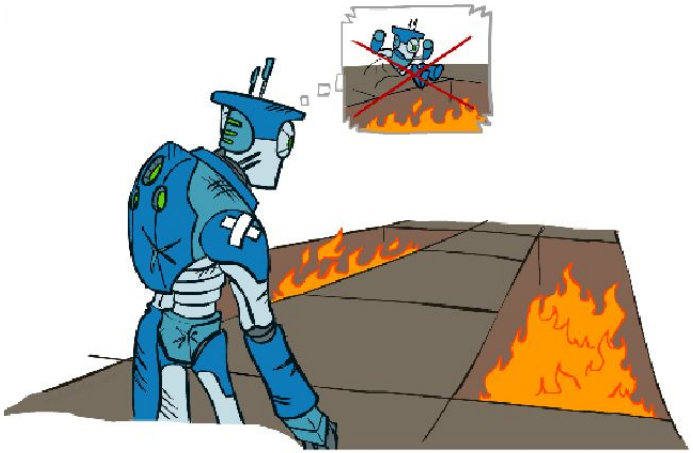
Reinforcement Learning using Parallel Computing



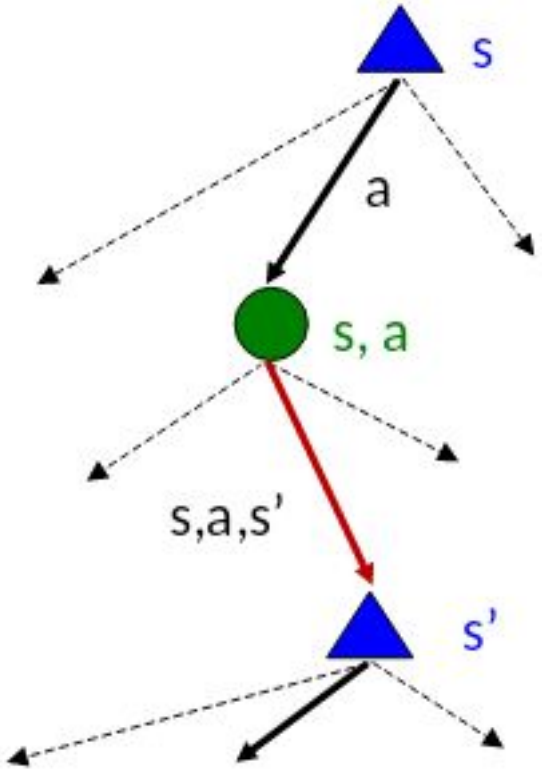
Gabriel Baptista - nUSP: 8941300
Hélio Assakura - nUSP: 8941064

Introdução

- MDP
- RL
- Parallel RL

Markov Decision Process - MDP

- Ambiente probabilístico(estocástico)
- Conjunto de estados S
- Conjunto de ações A
- Função de transição P
- (s, a, s') $\longrightarrow$ R(s, a, s'), função de Recompensa
- Estado atual depende apenas do anterior - Markov



s is a
state

(s, a) is a
q-state

(s,a,s') is a
transition

Política (π)

Sequência de ações

Política Ótima (π^*)

Política que maximiza o valor de recompensa recebido a cada estado

Valor de Iteração

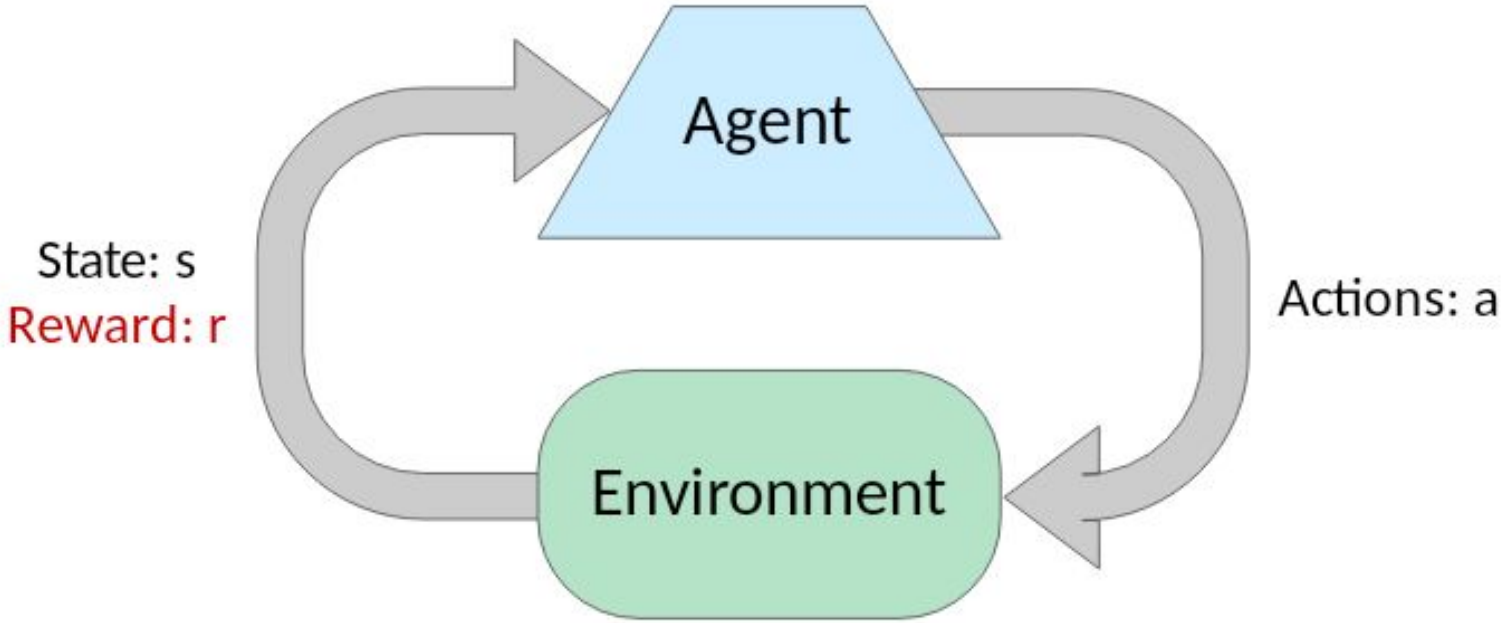
$$V^{\pi}(S) = R(S) + \gamma \sum_{S'} \mathbb{P}(S'|S, \pi(S)) V^{\pi}(S')$$

Equação de Bellman

$$V^{*}(S) = R(S) + \gamma \max_{A} \sum_{S'} \mathbb{P}(S'|S, A) V^{*}(S')$$

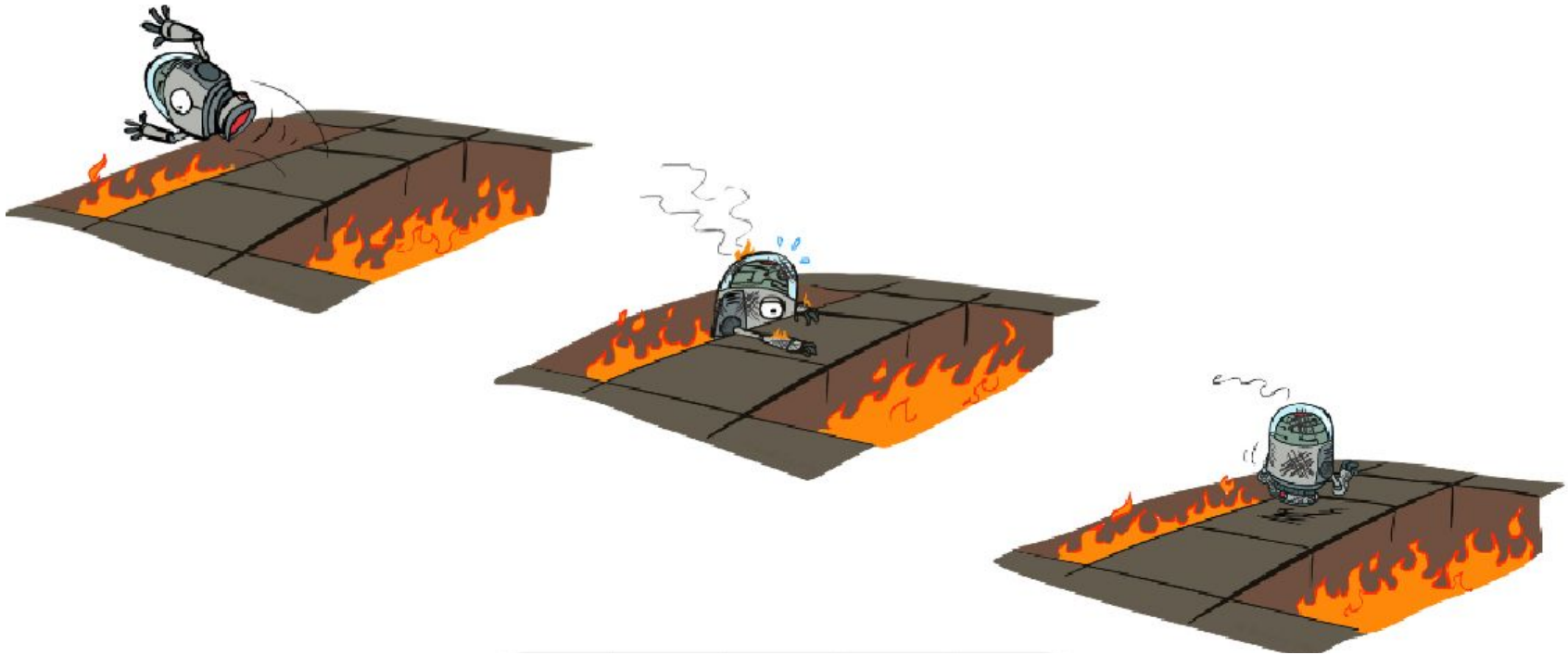
Reinforcement Learning - RL

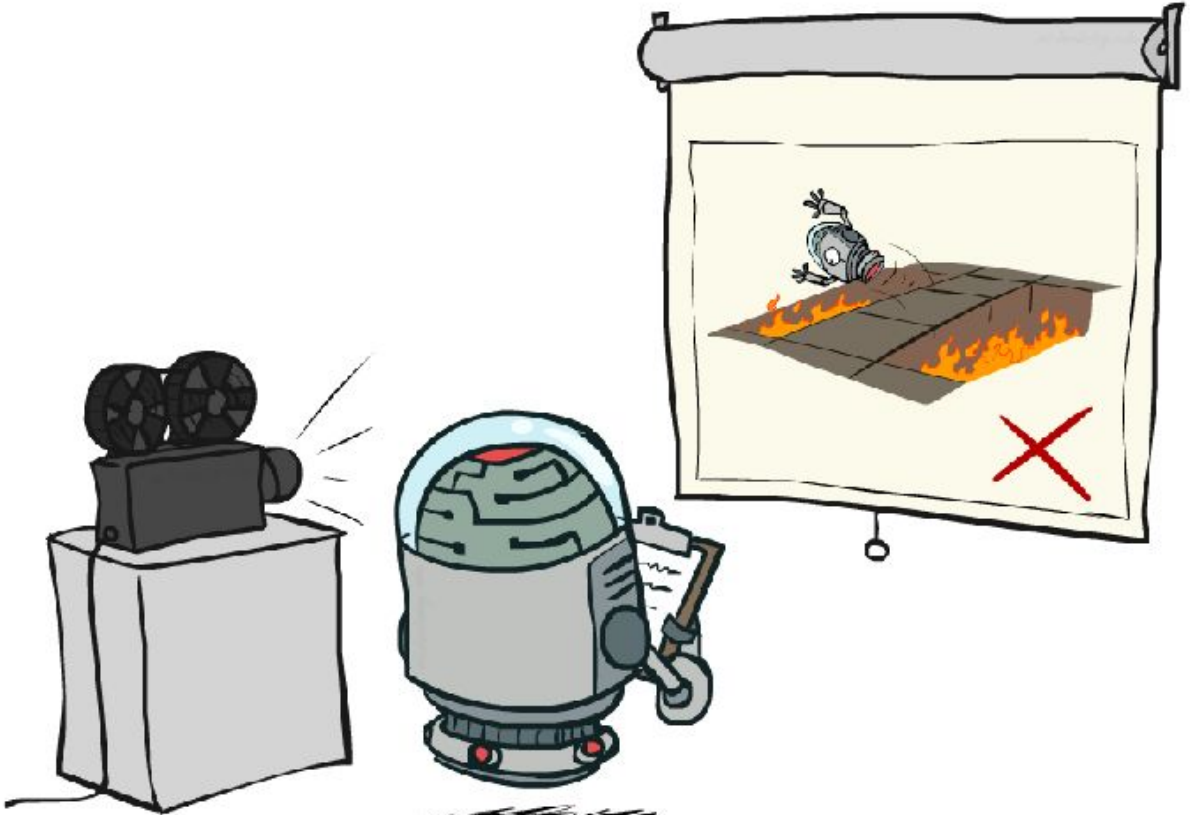
- Receber feedbacks na forma de Recompensas
- Utilidade do agente definida pela Função de Recompensa
- Deve aprender para agir de forma que maximize a recompensa esperada

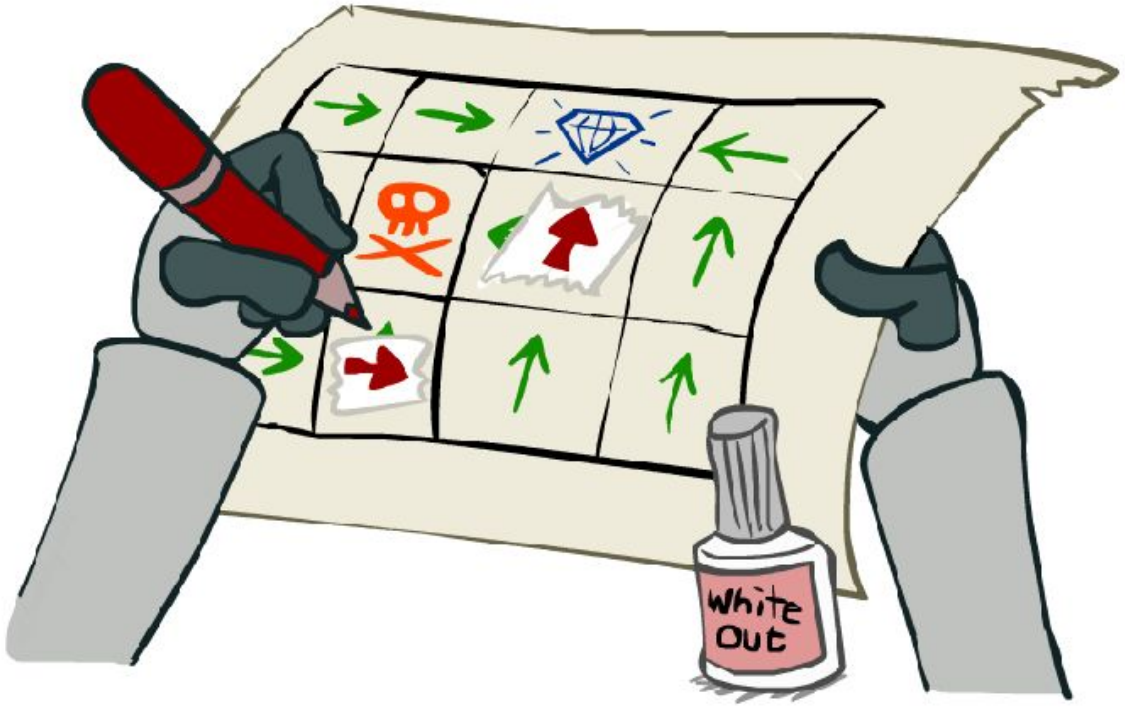

Agent
State: s
Reward: r
Actions: a
Environment

RL

- Modela-se o ambiente como um MDP
- Possui pelo menos um estado terminal
- Não se sabe a função de transição P e nem a função de recompensas R

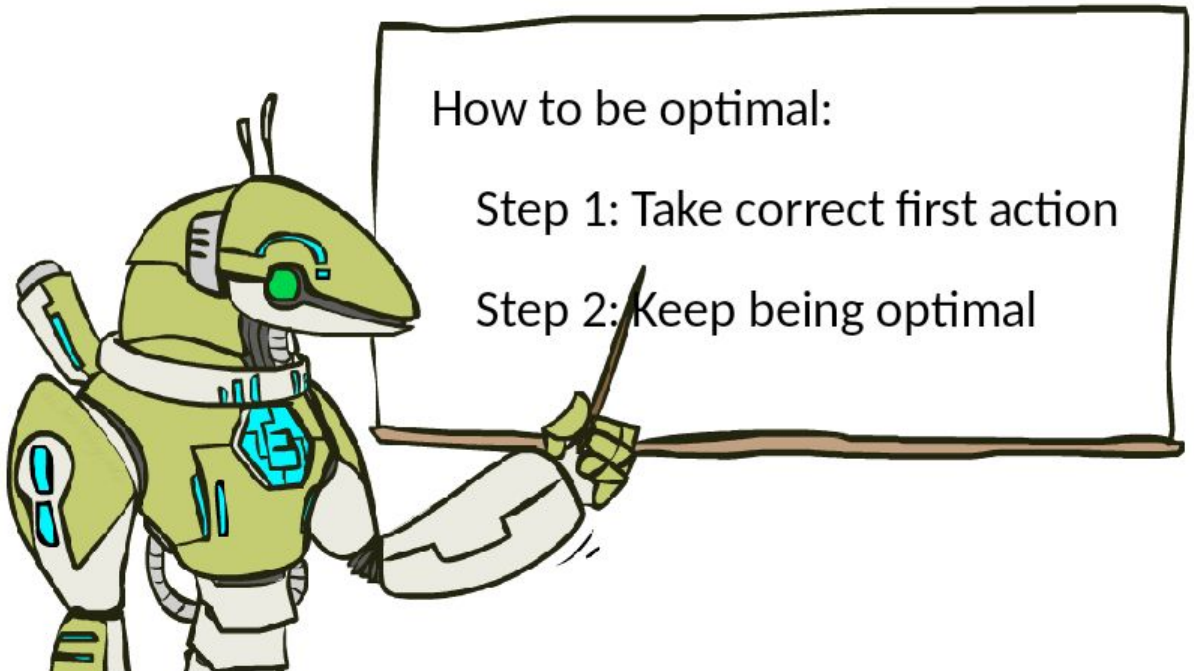





White
Out

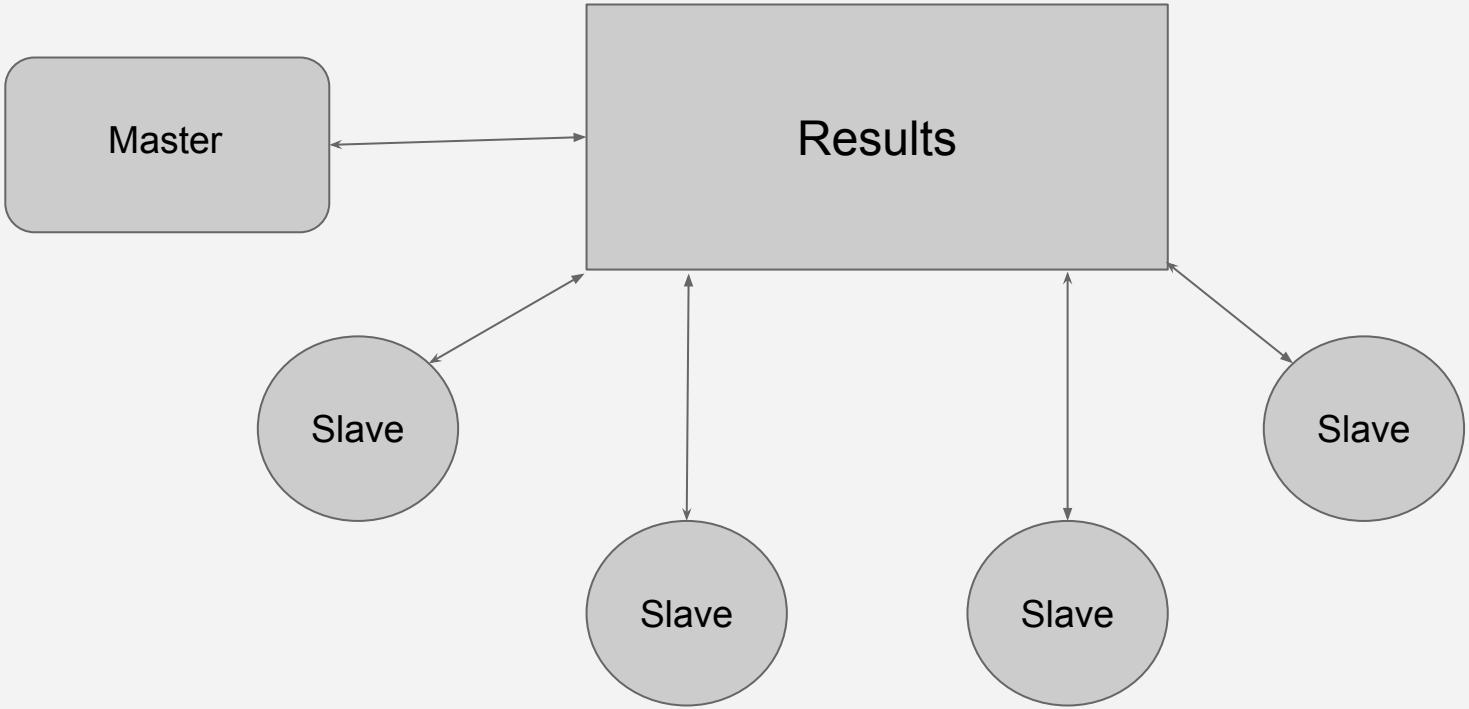
RL

- Algoritmo Q-Learning



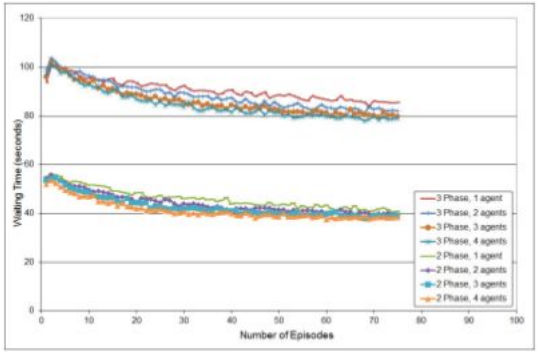
Parallel RL

- Parallel Learning
  - Múltiplos agentes juntam informações do mesmo problema
  - Agentes aprendem em diferentes instâncias do mesmo problema
- Multi Agent Reinforcement Learning (MARL)
  - Interações diretas entre agentes
  - Todos aprendem na mesma instância do problema

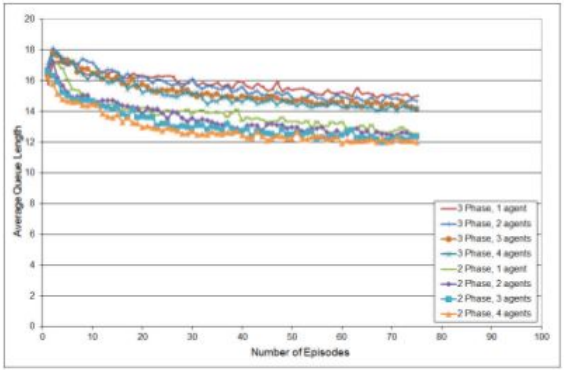

Master
Results
Slave
Slave
Slave
Slave

Parallel RL no controle de tráfego

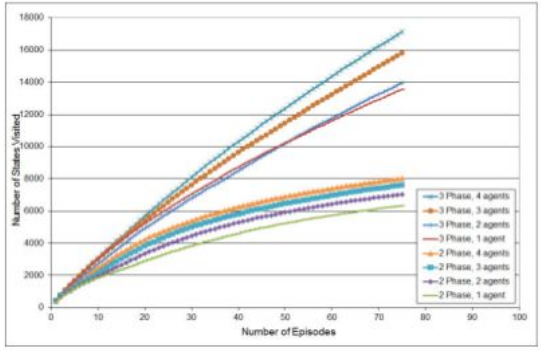
- Normalmente apresentam um único agente para cada interseção
  - Já foi visto que o uso de RL melhora o trânsito
- Uso de até 4 agentes simultâneos
  - Simulação “real”, com picos de quantidade de carros
  - Podem manter ou mudar o sinal para verde



(a) Average Waiting Times



(b) Average Queue Lengths



(c) Average Number of States Visited

Experiment	AWT (s)	% Reduction AWT	AQL	% Reduction AQL	ANSV	% Increase ANSV
2 Phase, 1 agent	41.45	-	12.74	-	6,170.02	-
2 Phase, 2 agents	39.84	3.88 %	12.37	2.90 %	6,874.23	11.41 %
2 Phase, 3 agents	38.66	6.73 %	12.23	4.00 %	7,425.38	20.35 %
2 Phase, 4 agents	38.29	7.62 %	12.12	4.87 %	7,868.37	27.53 %
3 Phase, 1 agent	85.52	-	15.06	-	13,001.19	-
3 Phase, 2 agents	82.45	3.59 %	14.77	1.93 %	13,326.58	2.50 %
3 Phase, 3 agents	80.40	5.99 %	14.42	4.25 %	15,035.03	15.64 %
3 Phase, 4 agents	78.86	7.79 %	14.21	5.64 %	16,322.52	25.55 %

Least-Squares Policy Iteration - LSPI

- Recebe como entrada um conjunto de amostras do ambiente e usa a função LSTDQ para calcular a função do valor de estado-ação durante a avaliação da política a ser adotada. Para a função *Q* no tempo 1, 2, ..., podemos achar a melhor ação para determinado estado.

Algorithm 1 LSPI

Given:
- D - Samples of the form (s, a, r, s')
- k - Number of basis functions
- ϕ - Basis functions
- γ - Discount factor
- ϵ - Stopping criterion
- w_0 - Initial policy

$w' \leftarrow w_0$
repeat
  $w \leftarrow w'$
  $w' \leftarrow \text{LSTD}Q(D, k, \phi, \gamma, w)$
until $(\| w - w' \| < \epsilon)$
return w

Algorithm 2 LSTDQ

Given:
- D - Samples of the form (s, a, r, s')
- k - Number of basis functions
- ϕ - Basis functions
- γ - Discount factor
- w - Current policy

$B \leftarrow \frac{1}{\delta} I$ // $(k \times k)$ matrix
$b \leftarrow 0$ // $(k \times 1)$ vector
for all $(s, a, r, s') \in D$ **do**
  $B \leftarrow B - \frac{B\phi(s,a)\left(\phi(s,a) - \gamma\phi(s', \pi(s'))\right)^T B}{1 + \left(\phi(s,a) - \gamma\phi(s', \pi(s'))\right)^T B\phi(s,a)}$
  $b \leftarrow b + \phi(s, a)r$
end for
$\tilde{w} \leftarrow Bb$
return $\tilde{w}$

Sequencial

- **scal** - realiza multiplicação vetor-escalar
- **axpy** - adição de dois vetores com um escalar multiplicando
- Funções que suportam CUDA

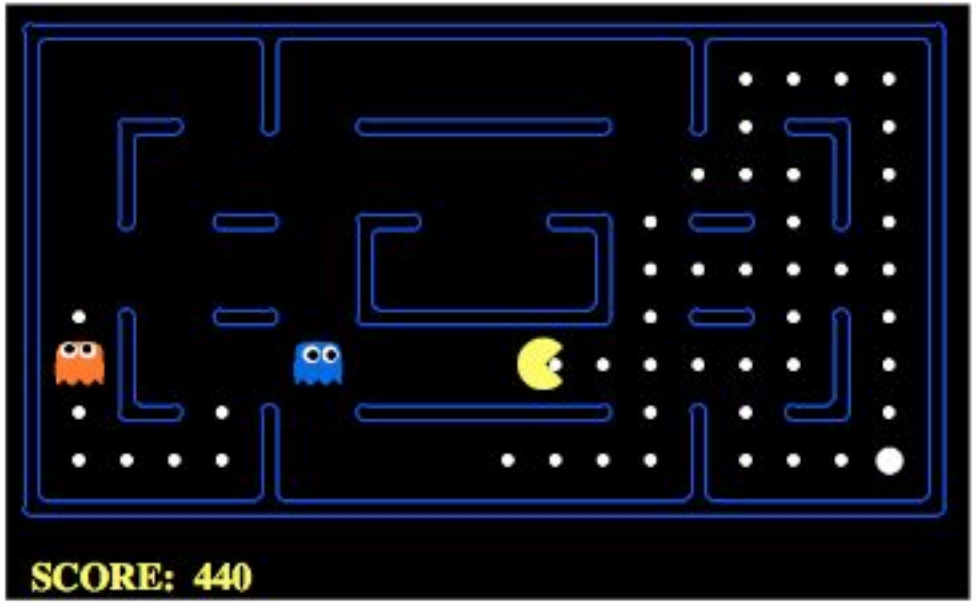
Algorithm 3 LSTDQ BLAS Implementation

Given:
- All input should be explicitly passed into GPU memory
- D - Samples of the form (s, a, r, s')
- k - Number of basis functions
- ϕ - Basis functions
- γ - Discount factor
- w - Current policy

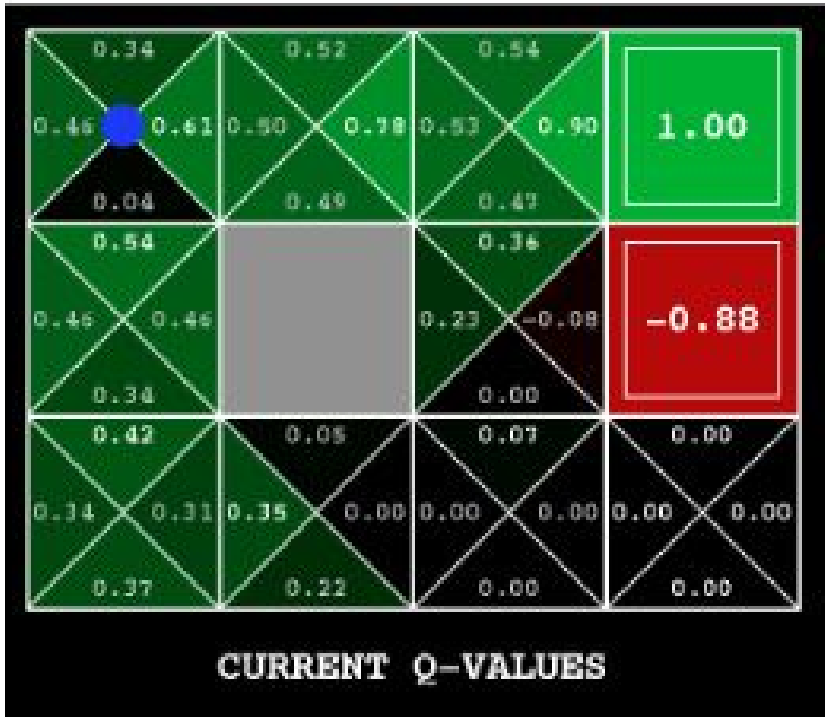
1: $B \leftarrow scal(I, \delta)$ // $(k \times k)$ matrix
2: $b \leftarrow 0$ // $(k \times 1)$ vector
3: **for all** $(s, a, r, s') \in D$ **do**
4: $x \leftarrow scal(\phi(s', \pi(s')), \gamma)$
5: $x \leftarrow axpy(\phi, x, -1)$
6: $x \leftarrow scal(x, -1)$ // Now $x = \phi(s, a) - \gamma\phi(s', \pi(s'))$
7: $y \leftarrow gemv(B, \phi(s, a))$
8: $z \leftarrow gemv(x, B)$
9: $num \leftarrow ger(y, z)$ // Calculate the numerator, see Equation (4.3)
10: $denom \leftarrow 1 + dot(\phi(s, a), z)$ // Calculate the denominator, see Equation (4.3)
11: $\Delta B \leftarrow scal(num, \frac{1}{denom})$
12: $B \leftarrow axpy(\Delta B, B, -1)$
13: $\Delta b \leftarrow scal(\phi(s, a), r)$
14: $b \leftarrow axpy(\Delta b, b)$
15: **end for**
16: $\tilde{w} \leftarrow gemv(B, b)$
17: **return** $\tilde{w}$

- $gemv$ - Performs vector-matrix multiplication
- ger - Performs vector-vector multiplication
- dot - Calculates the dot product of two vectors

Pacman



Gridworld


0.34
0.46
0.61
0.04
0.52
0.50
0.78
0.49
0.54
0.53
0.90
0.47
1.00
0.54
0.46
0.46
0.34
0.36
0.23
-0.08
0.00
-0.88
0.42
0.34
0.31
0.37
0.05
0.35
0.00
0.22
0.07
0.00
0.00
0.00
0.00
0.00
0.00
0.00
CURRENT Q-VALUES

Outros projetos

- Controle de robôs
- Rota de elevadores
- Telecomunicações
- Backgammon
- Damas
- Go (AlphaGo)
- Treinamento de agentes (OpenAI Gym)

Referências

- Notas de aulas:
  - https://www.ime.usp.br/~ddm/mac425/aulas/mdp.pdf
  - https://www.ime.usp.br/~ddm/mac425/aulas/rl.pdf
  - https://www.ime.usp.br/~ddm/mac425/aulas/rl2.pdf
  - http://ai.berkeley.edu/lecture_slides.html
- Artigos:
  - engr.case.edu/ray_soumya/papers/TylerGoeringer_thesis.pdf
  - http://ai2-s2-pdfs.s3.amazonaws.com/d171/424c1c87afdd3d169b4df781797820eeec62.pdf
  - http://www.sciencedirect.com/science/article/pii/S1877050915009722
- Outras fontes:
  - https://devblogs.nvidia.com/parallelforall/train-reinforcement-learning-agents-openai-gym/
  - https://en.wikipedia.org/wiki/Reinforcement_learning