Hierarchical RL

Valdinei Freire

(EACH - USP)

Hierarchical Reinforcement Learning

Flat RL

- tamanho do caminho até a recompensa influencia no custo de aprendizado
- propagação da informação
- exploração

Hierarchical RL

- atribuição de crédito de horizonte longo
- exploração estruturada
- transfer learning

Hierarchical Reinforcement Learning

Formalismos:

- Macro-actions
- Options
- Sub-tasks
- Abstract Actions
- Temporal Abstraction

Semi-MDP

Considere que no problema de decisão sequencial existem épocas de decisão: $t_0, t_1, \ldots$

MDP: épocas de decisão fixos $\{t_0 = 0, t_1 = 1, t_2 = 2, \ldots\}$

Semi-MDP: épocas de decisão $t_i \in \mathbb{R}$

Semi-MDP

Seja $d_i = t_{i+1} - t_i$ o tempo entre duas épocas de decisão, r_{i+1} a recompensa acumulada entre as épocas de decisão t_i e t_{i+1}.

Dado o estado $s_{t_i} = s$ no tempo t_i e dada ação $a_{t_i} = a$ escolhida no tempo t_i.

As variáveis aleatórias d_i, r_{i+1} e s_{t+1} dependem apenas de s e a e seguem uma distribuição conjunta dada pela função cumulada de probabilidade:

$$F(d, r, s'|s, a)$$

Função valor:

$$V(s) = \max_{a \in \mathcal{A}} \int_{d,r,s'} r + \gamma^d V(s') \mathrm{d}F(d, r, s'|s, a)$$

Options Framework

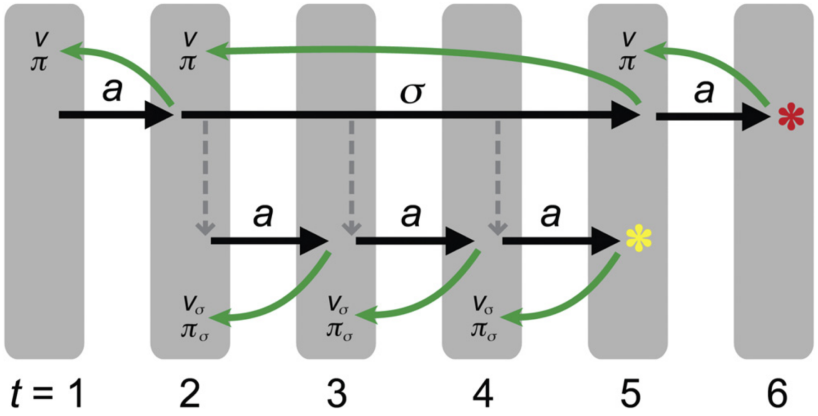
Uma *option* é definida por uma tupla $\langle I_\omega, \pi_\omega, \beta_\omega \rangle$:

- $I_\omega \subseteq \mathcal{S}$ é o conjunto de estados onde a *option* ω pode ser iniciada;
- $\pi_\omega : \mathcal{S} \in \mathcal{A}$ é a política seguida pelo agente quando a *option* ω é executada; e
- $\beta_\omega : \mathcal{S} \in [0, 1]$ é a função que indica a probabilidade de uma ação terminar em qualquer estado.

Options Framework

Considere um MDP $M = \langle \mathcal{S}, \mathcal{A}, T, R \rangle$ e um conjunto de *options* Ω, de tal forma que para todo estado $s \in \mathcal{S}$ existe $\omega \in \Omega$ tal que $s \in I_\omega$.

O MDP M e o conjunto de *options* Ω definem um Semi-MDP.



Options Framework

Dado um conjunto de *options*, pode-se considerar uma política inter-options:

$$\pi : \mathcal{S} \rightarrow \Omega$$

ou

$$\pi : \mathcal{S} \times \Omega \rightarrow [0, 1]$$

Pode-se então utilizar qualquer algoritmo para encontrar a política ótima hierárquica.

Recuperando otimalidade: pode-se considerar as ações de um MDP como *options* atômicas: $I_a = \mathcal{S}$, $\pi_a(s) = a$, e $\beta_a = 1$.

Feudal Reinforcement Learning

Múltiplas escalas temporais (*option* de *options*)

Divida e conquiste

Abstrato, mas com sub-metas concretas

Representações complementares

Feudal Reinforcement Learning

Estrutura de Árvore: super-managers, managers, sub-managers

Recursivamente:

- Managers tem poder absoluto sobre seus sub-managers: define qual função recompensa eles devem maximizar
- Managers devem satisfazer seus super-managers: maximiza uma função recompensa
- Managers possui um grau de abstração maior que os sub-managers e menor que o seu super-manager
- Managers cobrem um espaço de estado maior que os seus sub-managers e menor que o seu super-manager

Feudal Reinforcement Learning

Quando um manager escolhe uma ação, controle é delegado para o sub-manager e é apenas retornado quando o estado muda no nível de abstração do manager.

Reward Hiding: as recompensas escolhidas para os sub-managers podem ser diferentes das recompensas escolhidas pelos super-managers, permite que sub-managers aprendam a atingir uma meta mesmo que a meta do super-manager não foi atingida.

Information Hiding: managers só precisam conhecer o estado do sistema segundo sua abstração

Feudal Reinforcement Learning

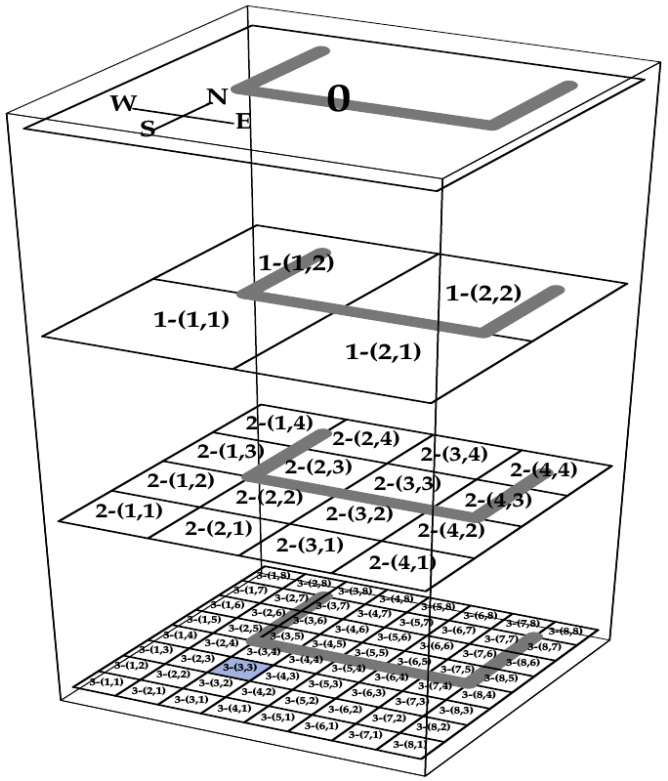


Figure 1: Figure 1: The Grid Task. This shows how the maze is divided up at different levels in the hierarchy. The 'U' shape is the barrier, and the shaded square is the goal. Each high level state is divided into four low level ones at every step.

Feudal Reinforcement Learning

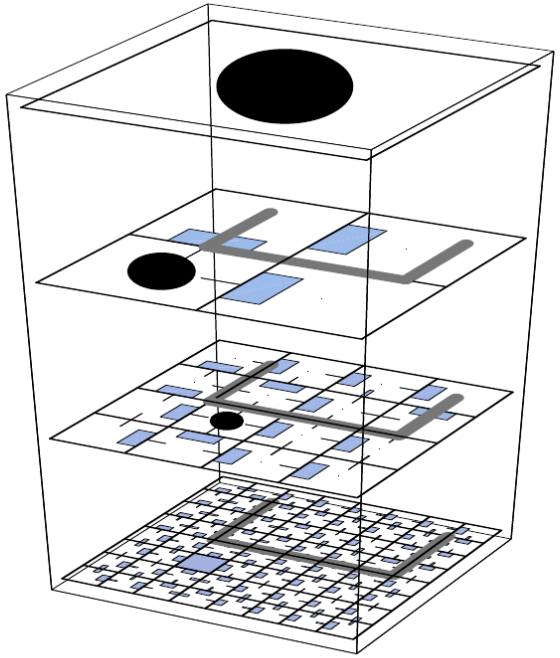


Figure 3: The Learned Actions. The area of the boxes and the radius of the central circle give the probabilities of taking action NSEW and * respectively.

MAX-Q

Similar ao Feudal RL.

Permite um manager ser utilizado em mais de um nível.

Permite um manager ser reutilizado.

Cada option é aprendida utilizando uma pseudo-função de recompensa $\tilde{R}(s, a)$.

MAX-Q

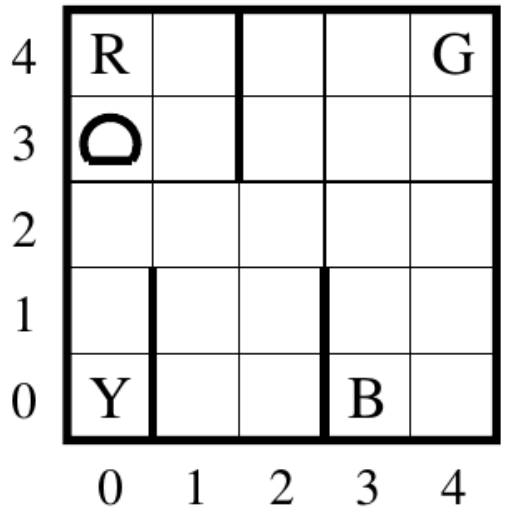


Figure 1: The Taxi Domain.

Problema do Taxista:

- Pega passageiro em um dos 4 lugares (RGBY)
- Entrega passageiro em um dos 4 lugares

MAX-Q

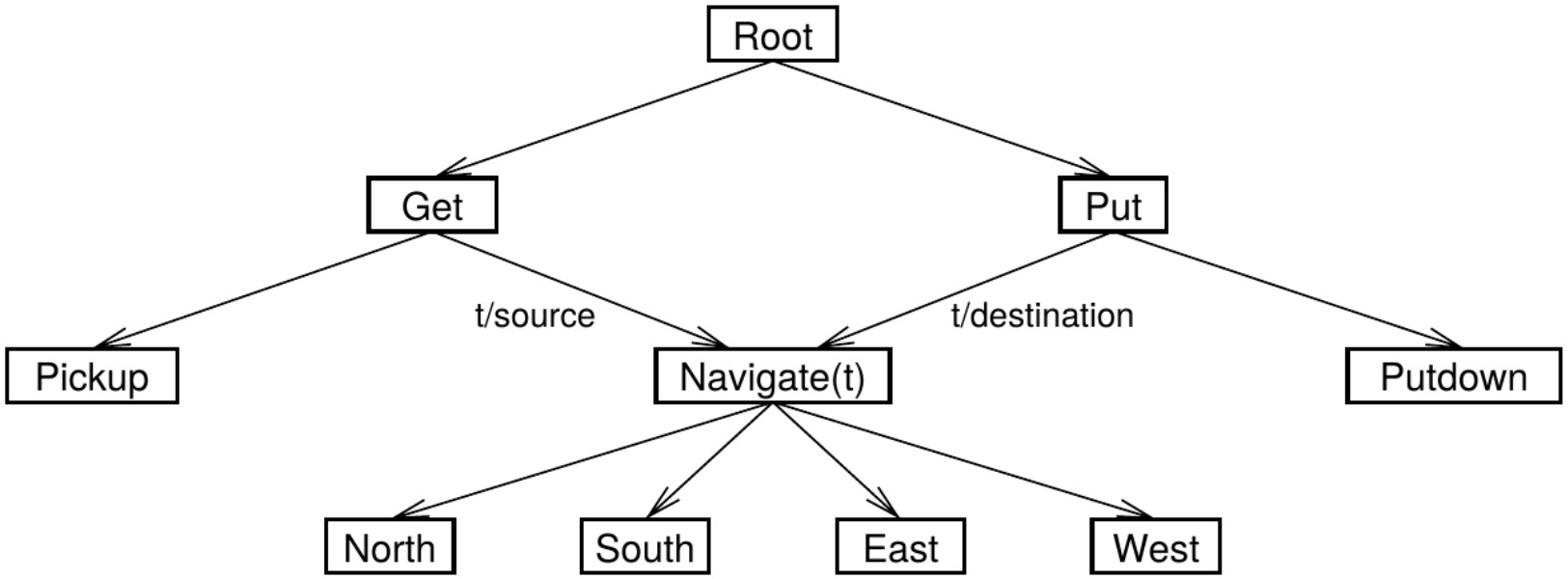
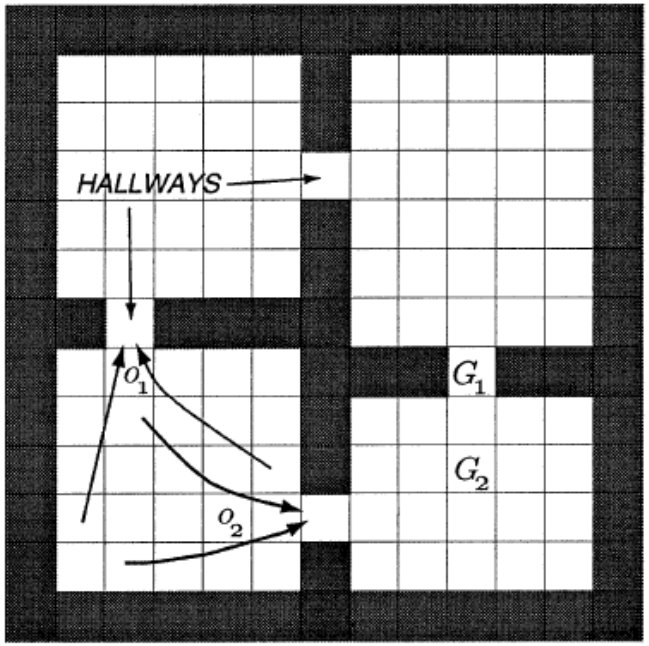


Figure 2: A task graph for the Taxi problem.

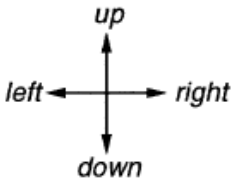
Option Discover

- Feudal RL considera uma sub-divisão específica para o problema e é problema dependente
- MAX-Q é menos restritivo, mas depende da pré-especificação de sub-tarefas
- Pergunta: como descobrir automaticamente *options* independe do domínio?
  - descoberta de sub-goals
  - busca no espaço de *options*

Bottleneck



4 stochastic primitive actions



Fail 33% of the time

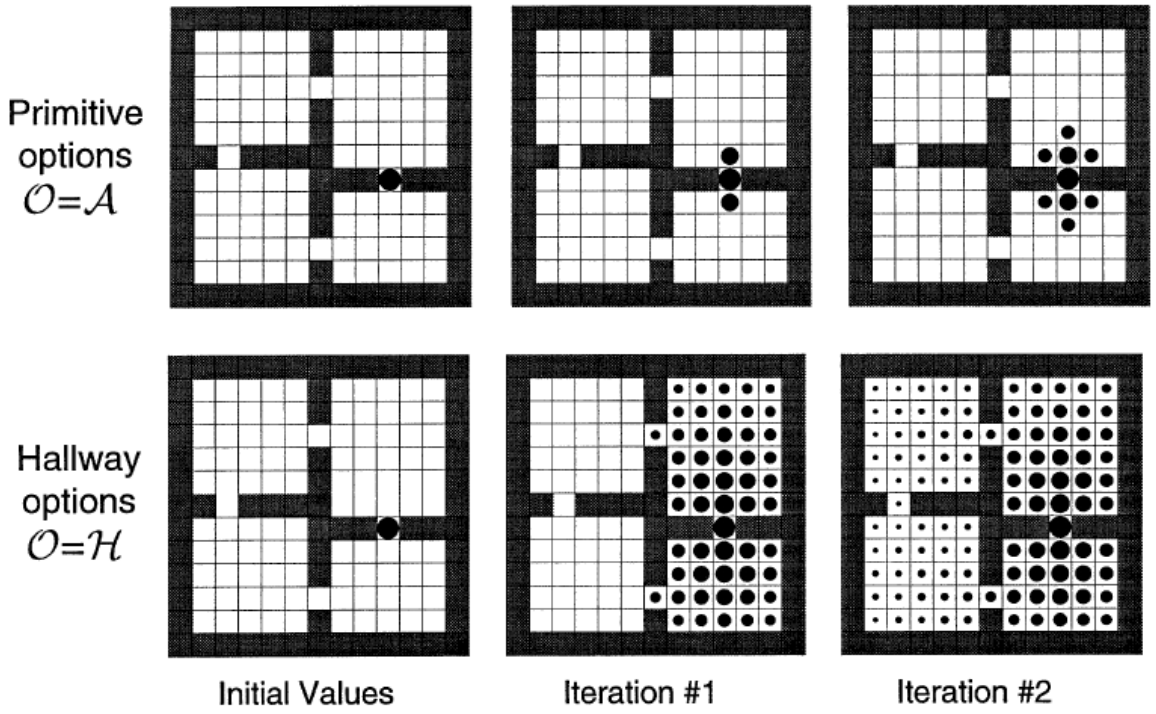
8 multi-step options

(to each room's 2 hallways)

Bottleneck

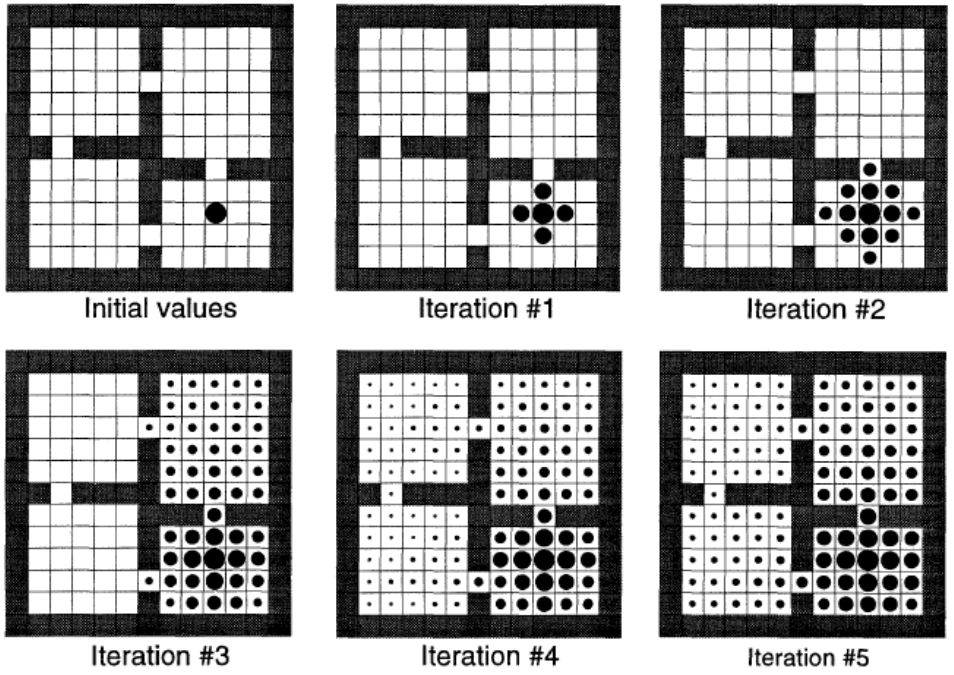
1. Selecione aleatoriamente N pares de estados (s_I, s_T)
2. Para cada par (s_I, s_T)
   (a) Resolva a tarefa de iniciar em s_I e chegar em s_T.
   (b) Execute a política ótima por K episódios e acumule para cada estado s a quantidade $n(s)$ que ele foi visitado.
3. Repita até encontrar a quantidade necessária de *options*
   (a) escolha o estado que foi mais visitado, isto é, $s^* = \arg\max_{s \in \mathcal{S}} n(s)$
   (b) compute $n(s, s^*)$, a quantidade de vezes que s ocorreu em uma mesma trajetória antes de s^*
   (c) compute $\bar{n}(s^*) = \frac{\sum_{s \in \mathcal{S}} n(s, s^*)}{|\mathcal{S}|}$
   (d) construa o conjunto de estados iniciais $I = \{s \in \mathcal{S} | n(s, s^*) > \bar{n}(s^*)\}$
   (e) complete o conjunto de estados iniciais I interpolando os estados em I
   (f) elimine estados que já foram contemplados pela meta atual
4. Para cada *option*, aprenda sua política interna

Bottleneck


Primitive options
$\mathcal{O}=\mathcal{A}$
Hallway options
$\mathcal{O}=\mathcal{H}$
Initial Values
Iteration #1
Iteration #2

Bottleneck

Primitive and hallway options $\mathcal{O}=\mathcal{A}\cup\mathcal{H}$



HIRO

HIRO: HIerarchical Reinforcement learning with Off-policy correction

- Alto-nível: indica posição absoluta indiretamente por uma direção
- Baixo-nível: aprende política genérica para todas posições
- Batch RL: reetiqueta experiências

HIRO

Alto-nível:

- A cada c passos:

$$g_t \sim \mu^{hi}(s)$$

- Nos passos restantes:

$$g_t = h(s_{t-1}, g_{t-1}, s_t) = s_t - g_{t-1} - s_{t-1}$$

- g_t indica mais do que direção, indica um acréscimo em s_t, que deve ser alcançado nos próximos c passos.

HIRO

Baixo-nível:

- Função Recompensa intrínseca e Parametrizada:

$$r_t = r(s_t, g_t, a_t, s_{t+1}) = - \parallel s_t + g_t - s_{t+1} \parallel_2$$

- Políticas de Baixo Nível Potencialmente Infinitas:

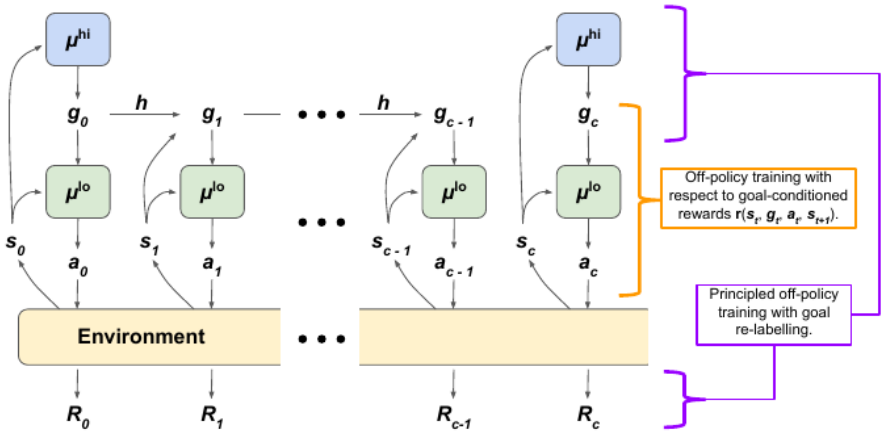
$$Q_\theta^{lo}(s_t, g_t, a_t)$$

HIRO

Algoritmo Off-policy

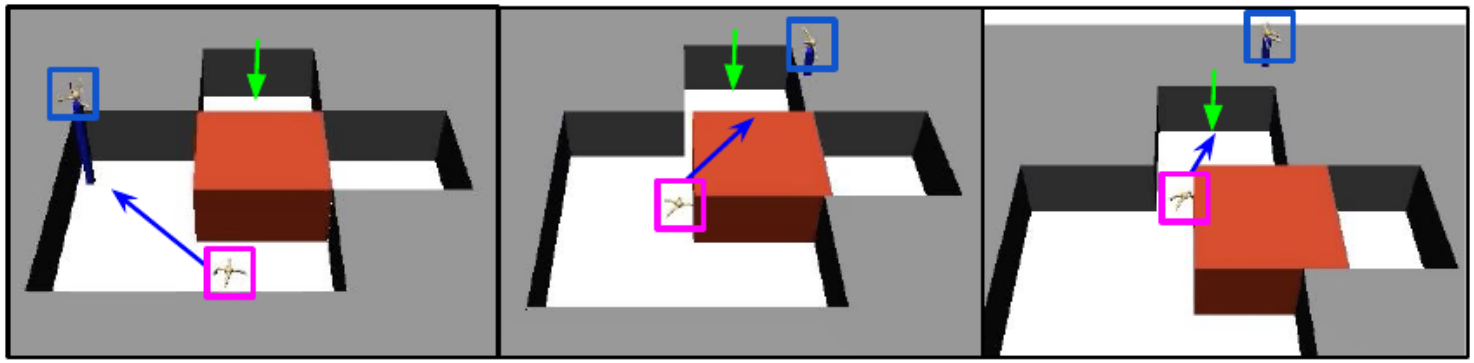
- coleta experiências de alto nível e baixo nível separadamente
- como a recompensa em baixo nível é intrínseca, pode-se utilizar qualquer algoritmo off-policy
- reetiqueta as experiência de alto nível
- considera o histórico na experiência e escolhe $\tilde{g}_t$ considerando a nova política de baixo nível baseado em likelihood

HIRO



1. Collect experience $s_t, g_t, a_t, R_t, \ldots$.
2. Train μ^{lo} with experience transitions $(s_t, g_t, a_t, r_t, s_{t+1}, g_{t+1})$ using g_t as additional state observation and reward given by goal-conditioned function $r_t = r(s_t, g_t, a_t, s_{t+1}) = -||s_t + g_t - s_{t+1}||_2$.
3. Train μ^{hi} on temporally-extended experience $(s_t, \tilde{g}_t, \sum R_{t:t+c-1}, s_{t+c})$, where $\tilde{g}_t$ is re-labelled high-level action to maximize probability of past low-level actions $a_{t:t+c-1}$.
4. Repeat.

HIRO



Feudal Networks for HRL

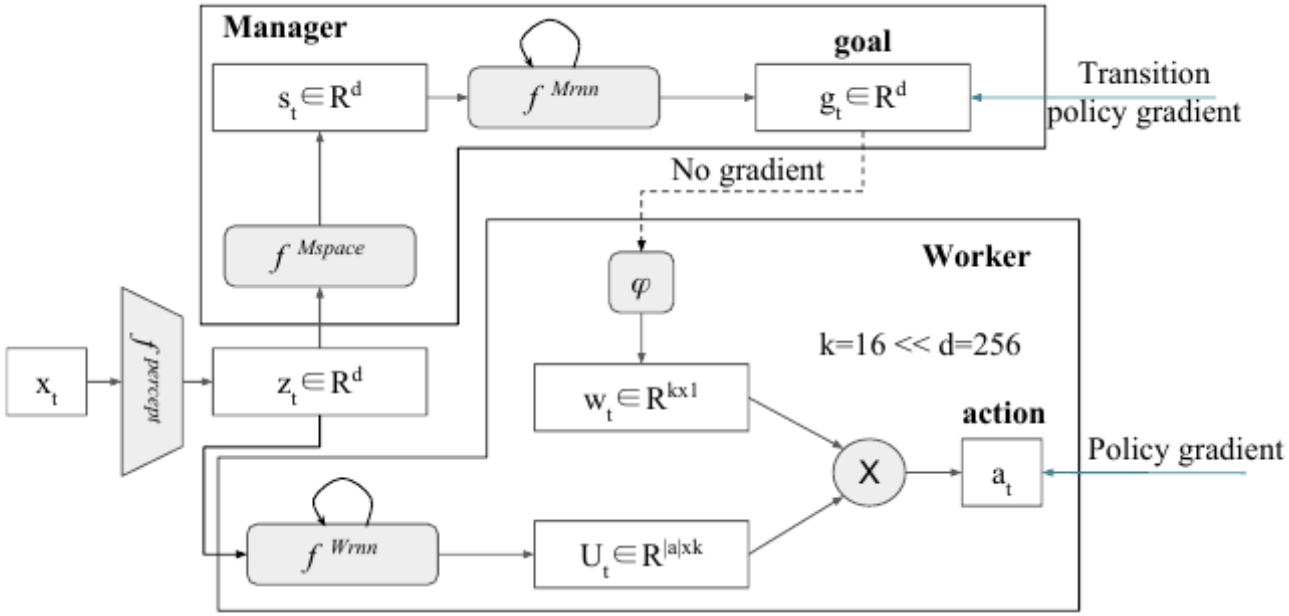


Figure 1. The schematic illustration of FuN (section 3)

Feudal Networks for HRL

Política de alto nível: Manager

$$g_t = \mu(s_t, \theta)$$

Assume-se distribuição de transição:

$$\mathsf{Pr}(s_{t+c}|s_t, g_t) \propto \exp(d_{cos}(s_{t+c} - s_t, g_t))$$

Política de transição:

$$\pi^{TP}(s_{t+c}|s_t; \theta) = \mathsf{Pr}(s_{t+c}|s_t, \mu(s_t, \theta))$$

Aprendendo θ:

$$\nabla g_t = (V_t - V(x_t))\nabla_\theta d_{cos}(s_{t+c} - s_t, g_t(\theta))$$

Feudal Networks for HRL

Política de baixo nível: Worker

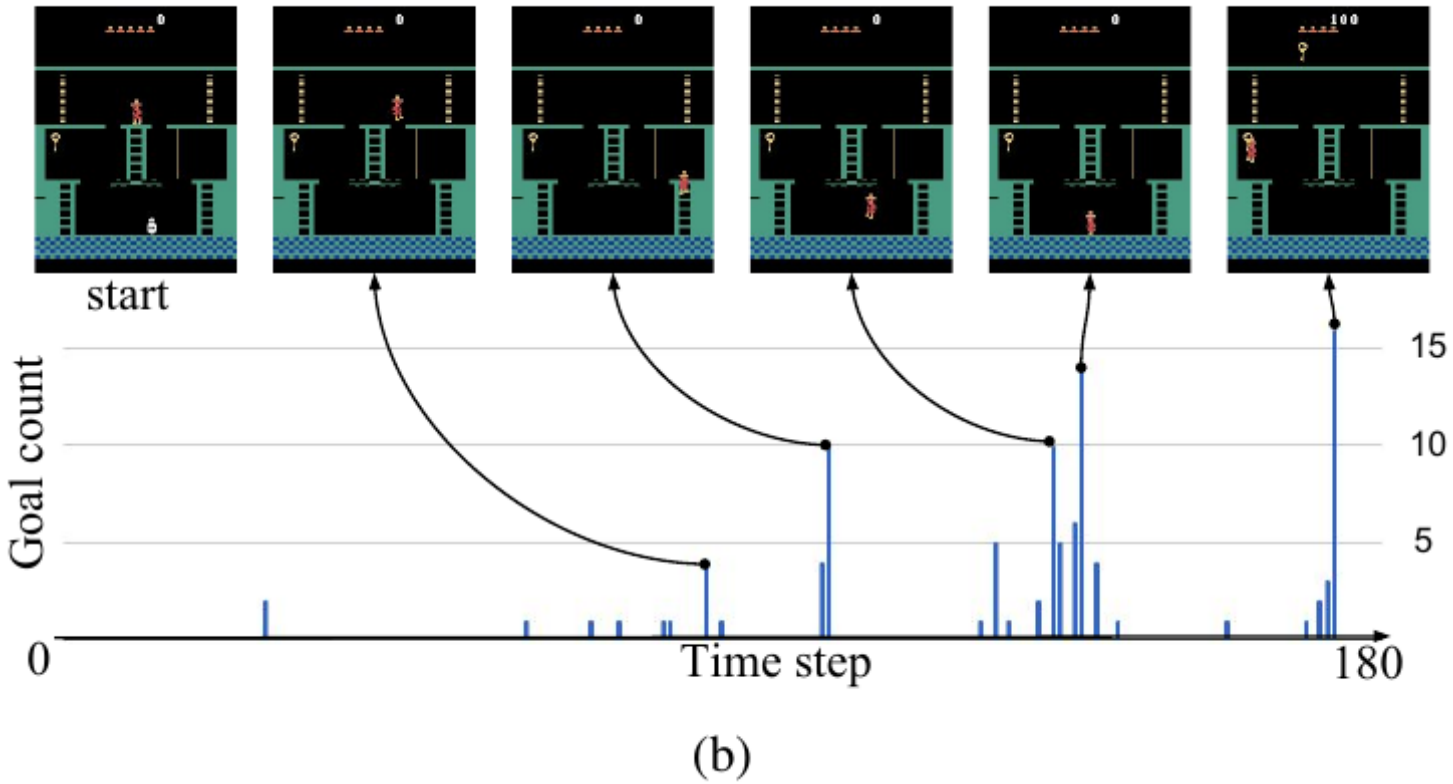
Worker produz um vetor para cada ação relacionado a cada possível meta.

Além da recompensa original (extrínseca), também considera uma recompensa intrínseca:

$$r_t^I = \frac{1}{c} \sum_{i=1}^{c} d_{cos}(s_t - s_{t-i}, g_{t-i})$$

O manager assume que o worker vai seguir uma determinada direção, enquanto o worker aprende a seguir essa direção.

Feudal Networks for HRL



(b)

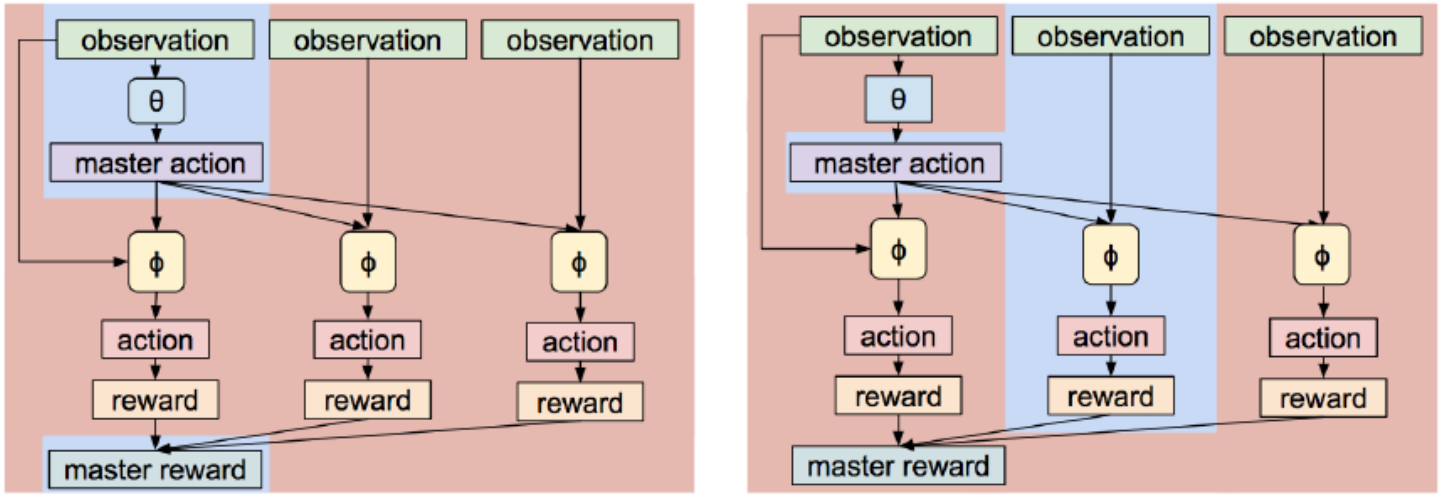
Life Long RL

Assumo que o agente vai resolver muitas tarefas

Aprendo um conjunto de *options* que pode me ajudar a resolver qualquer tarefa.

Considero alguma distribuição dos problemas potenciais.

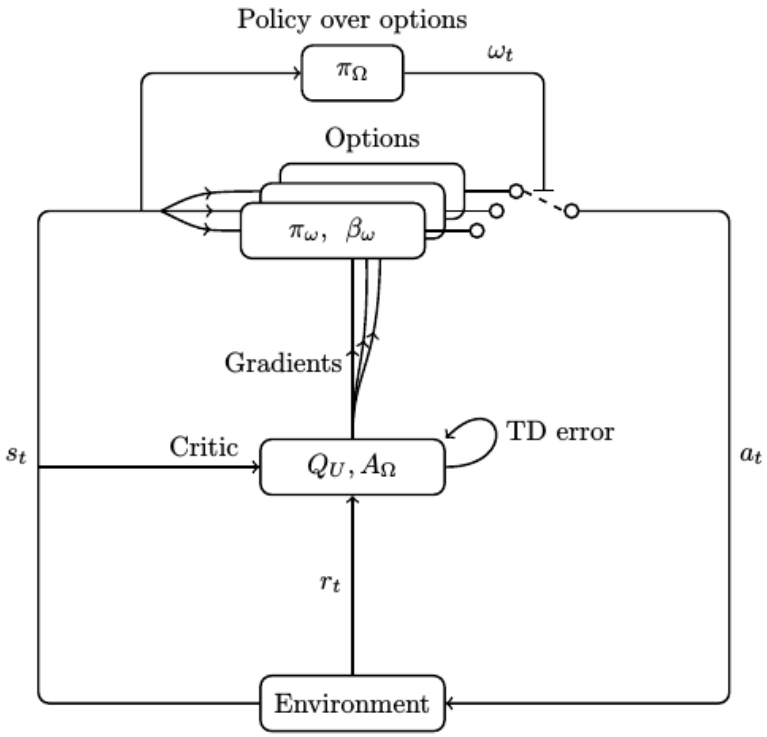
MLSH - Meta Learning Shared Libraries



MLSH - Meta Learning Shared Libraries

1. Inicializa ϕ
2. Repita
   (a) Inicialize θ
   (b) Sorteie uma tarefa $M \sim P_M$
   (c) durante um período de aquecimento faça
      i. colete D experiências utilizando $\pi_{\phi,\theta}$
      ii. atualize θ
   (d) durante um período de aprendizado conjunto
      i. colete D experiências utilizando $\pi_{\phi,\theta}$
      ii. atualize θ
      iii. atualize ϕ

Option-Critic Architecture



Option-Critic Architecture

Algorithm 1: Option-critic with tabular intra-option Q-learning

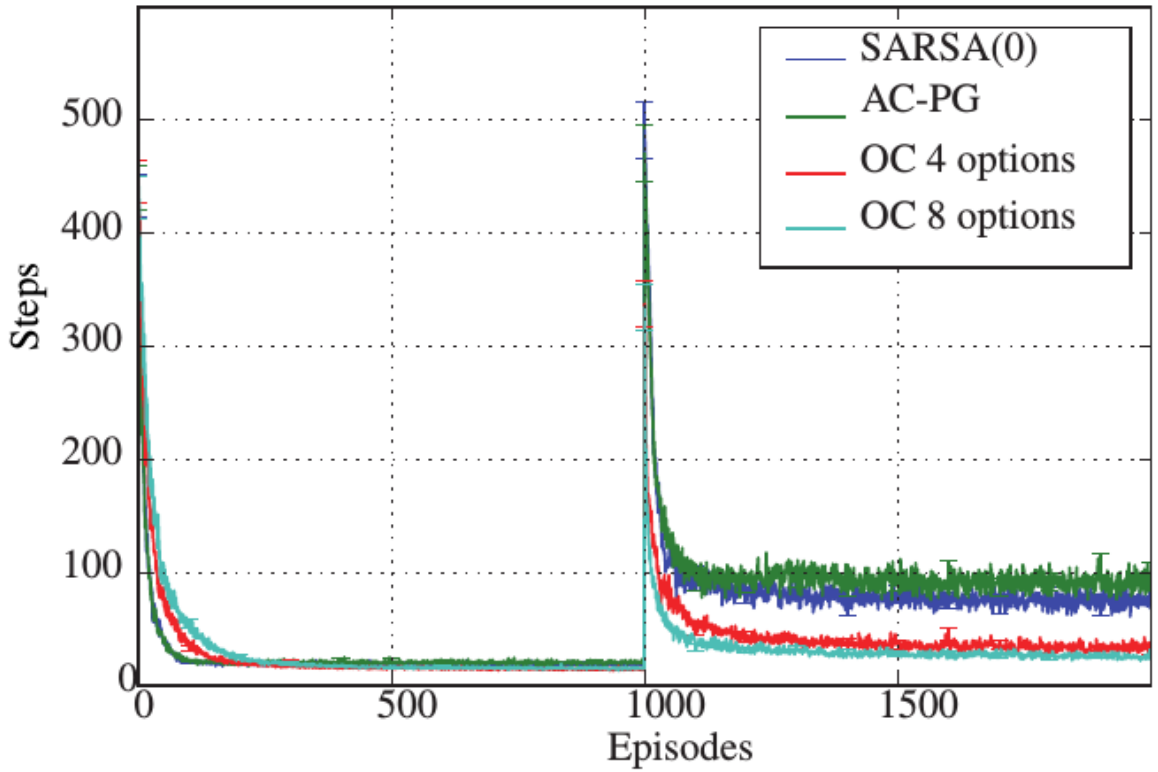
$s \leftarrow s_0$
Choose ω according to an ϵ-soft policy over options $\pi_\Omega(s)$
repeat
  Choose a according to $\pi_{\omega,\theta}(a \mid s)$
  Take action a in s, observe s', r

  1. Options evaluation:
  $\delta \leftarrow r - Q_U(s,\omega,a)$
  if s' *is non-terminal* **then**
    $\delta \leftarrow \delta + \gamma(1-\beta_{\omega,\vartheta}(s'))Q_\Omega(s',\omega) + \gamma\beta_{\omega,\vartheta}(s')\max_{\bar{\omega}} Q_\Omega(s',\bar{\omega})$
  end
  $Q_U(s,\omega,a) \leftarrow Q_U(s,\omega,a) + \alpha\delta$

  2. Options improvement:
  $\theta \leftarrow \theta + \alpha_\theta \frac{\partial \log \pi_{\omega,\theta}(a \mid s)}{\partial\theta} Q_U(s,\omega,a)$
  $\vartheta \leftarrow \vartheta - \alpha_\vartheta \frac{\partial \beta_{\omega,\vartheta}(s')}{\partial\vartheta}\left(Q_\Omega(s',\omega) - V_\Omega(s')\right)$

  if $\beta_{\omega,\vartheta}$ *terminates in* s' **then**
  choose new ω according to ϵ-soft$(\pi_\Omega(s'))$
  $s \leftarrow s'$
until s' *is terminal*

Option-Critic Architecture



Value-Based Option Architecture

Given an MDP M, a probability of terminating β, and a fixed quantity of options O; we define a new MDP $M' = \langle \mathcal{X}', \mathcal{U}', U', R', T' \rangle$ where:

- $\mathcal{X}' = \mathcal{S} \times \Omega$ where $\Omega = \{0, 1, \ldots, O\}$, i.e., consider $x = (s, \omega) \in \mathcal{X}'$, the agent may be in an option ($w \geqslant 1$) or not ($w = 0$);

- $\mathcal{U}' = \mathcal{A} \times (\Omega - \{0\})$, i.e., the agent can choose an atomic action or an option;

- if the agent is in an option, the agent can choose atomic actions; otherwise the agent can choose options, i.e., for any $x = (s, \omega) \in \mathcal{X}'$

$$U'(x) = \begin{cases} A(s) \times \{w\} & \text{, if } \omega \geqslant 1 \\ A(s) \times (\Omega - \{0\}) & \text{, otherwise} \end{cases};$$

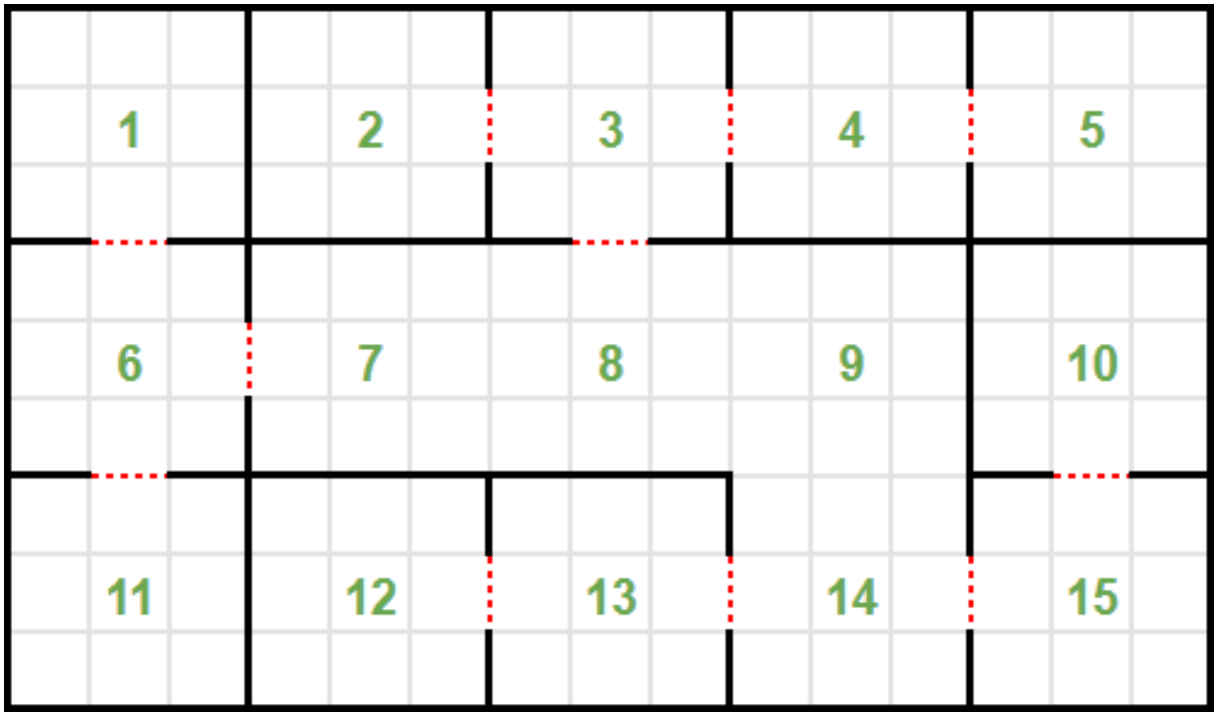
- given any state $x = (s, \omega) \in \mathcal{X}'$, next state $x' = (s', \omega') \in \mathcal{X}'$ and action $u = (a, \omega_a) \in U'(x)$, because whether $\omega_a = \omega$ or $\omega = 0$, we have that:

$$T'(x, u, x') = \begin{cases} \beta T(s, a, s') & \text{, if } \omega' = 0 \\ (1 - \beta) T(s, a, s') & \text{, if } \omega_a = \omega' \\ 0 & \text{, otherwise} \end{cases}$$

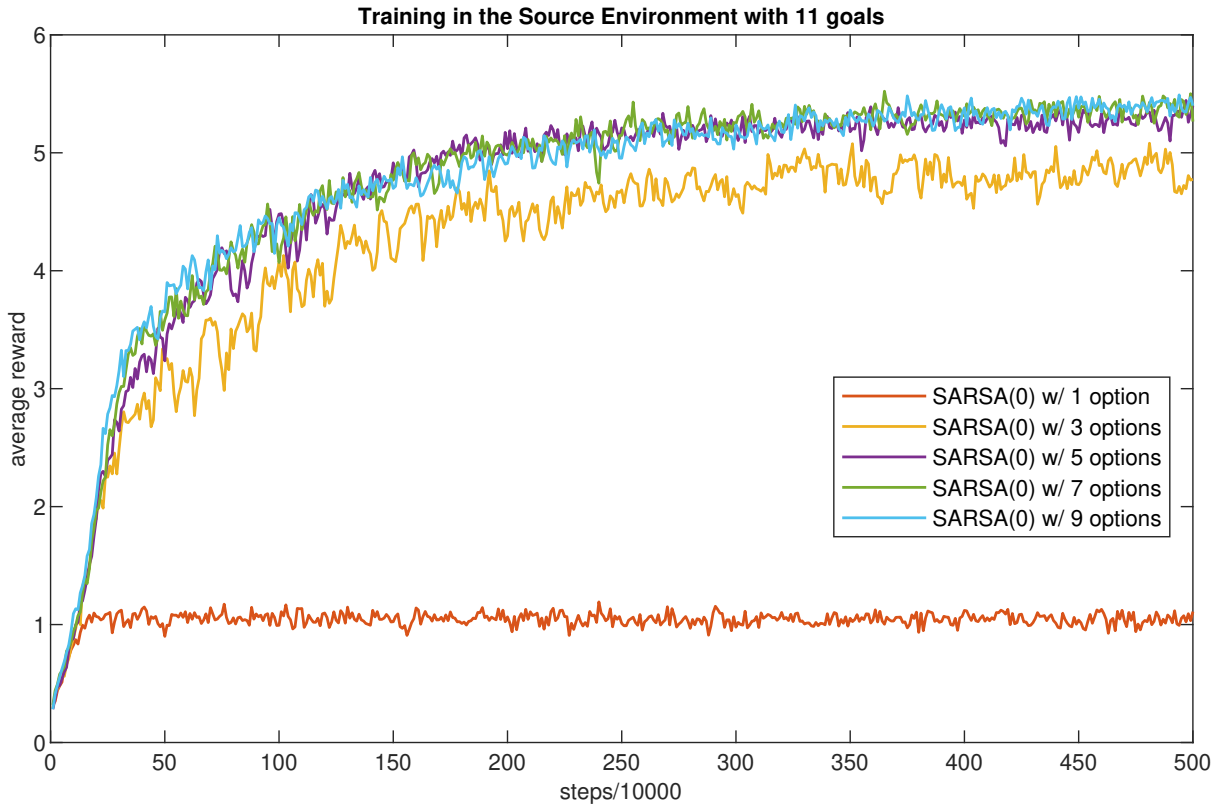
and

$$R(x, u, x') = R(s, a, s').$$

Value-Based Option Architecture



Value-Based Option Architecture



Option Architecture

Como aprender options úteis:

- abstração local: para garantir options que podem ser utilizadas in diferentes ambientes/tarefas
- abstração global: para garantir qualidade da política aprendida

Referências

1995 Feudal Reinforcement Learning, Dayan and Hinton

2000 Hierarchical Reinforcement Learning with the MAXQ Value Function Decomposition, Dietterich

2017 FeUdal Networks for Hierarchical Reinforcement Learning, Vezhnevets

2017 The Option-Critic Architecture, Bacon et al.

2018 Data-Efficient Hierarchical Reinforcement Learning, Nachum et al.

2018 Meta Learning Shared Hierarchies, Frans et al.