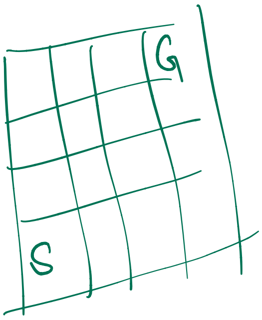
Lecture-11

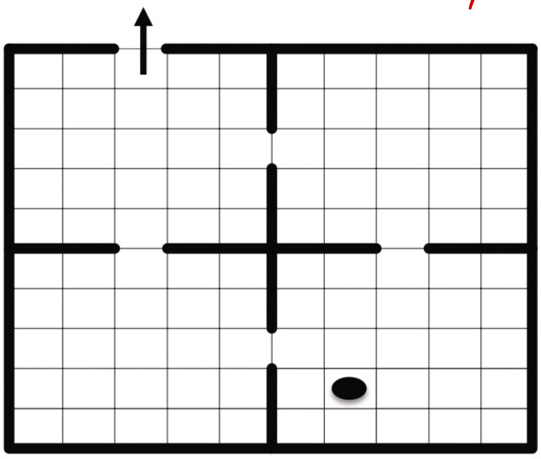
Hierarchical RL

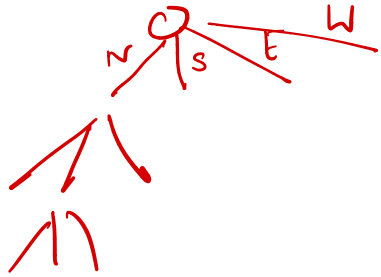
flat RL

Primitive action → N, S, E, W

macro actions → exit the room from east door
West
North door







Hierarchical RL

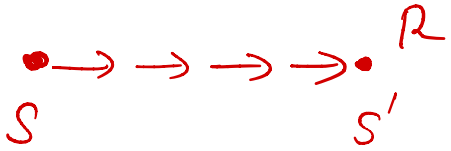
aims to find good re-usable temporally extended actions that may also provide opportunities for state abstractions.

Advantages:

1. Scalability
2. reusability / transfer learning
3. parallelize learning
4. long-term credit assignment
5. structured exploration.

Abstract actions:- ⟶

- temporally extended actions.



Ex: Room-leaving action.

- 'Macros' in programming

---

higher level RL agent ⟺ Program.

options/macro-action ⟺ Sub-routines.

Semi-MDP:-

SMDP → MDP that includes abstract actions.

$N \geq 1$ — random variable that denotes the # of steps that an abstract action 'a' takes to complete.

Transition fn: $T: S \times A \times S \times N \rightarrow [0, 1]$

$$T(s, a, s', N) = \Pr\{S_{t+N} = s' \mid S_t = s, a_t = a\}$$

Reward fn: $R: S \times A \times S \times N \rightarrow \mathbb{R}$

$$R(s, a, s', N) = \mathbb{E}\left[\sum_{n=0}^{N-1} \gamma^n r_{t+n} \mid S_t = s, a_t = a, S_{t+N} = s' \right]$$

State value fn

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{k=0}^{\infty} \gamma^k r_{t+k} \,\middle|\, S_t = s\right]$$

$$= \mathbb{E}_{\pi}\Big[\left(r_t + \gamma r_{t+1} + \ldots + \gamma^{N-1} r_{t+N-1}\right) + \left(\gamma^N r_{t+N+1} \ldots\right) \Big| S_{t+N}\Big]$$

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

Optimal value fn

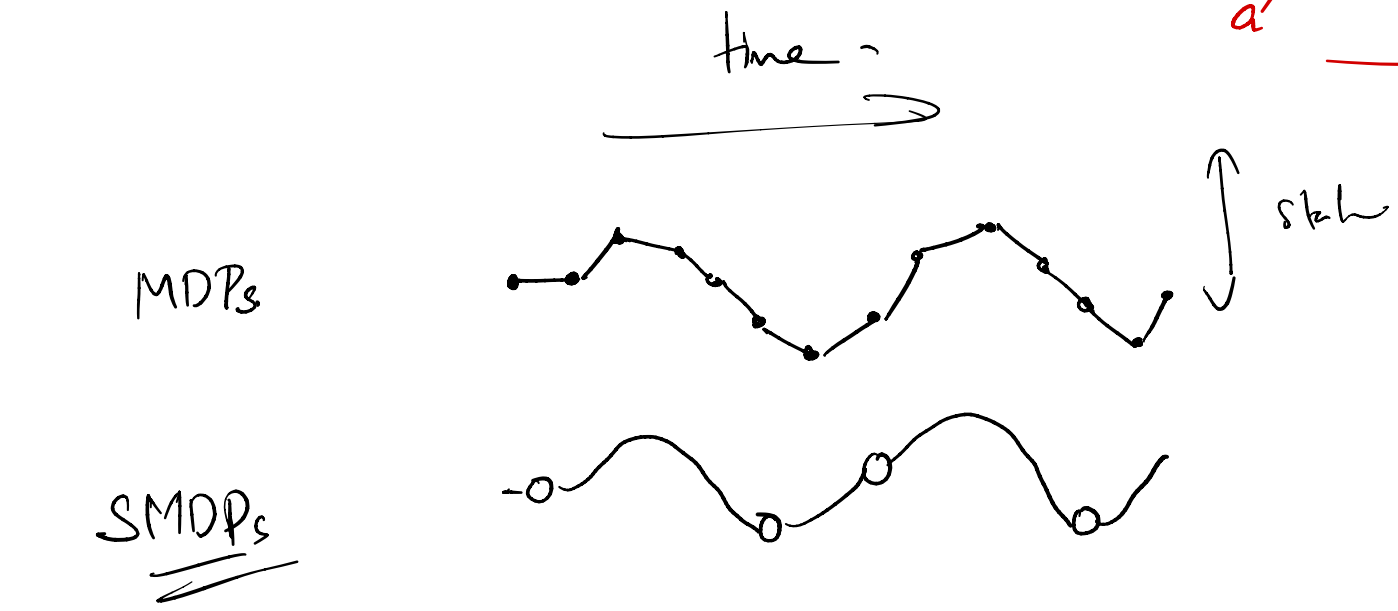
$$V^*(s) = \max_a \sum_{S', N} T(S, a, S', N)\left[R(S, a, S', N) + \gamma^N V^*(S')\right]$$

State action value fn

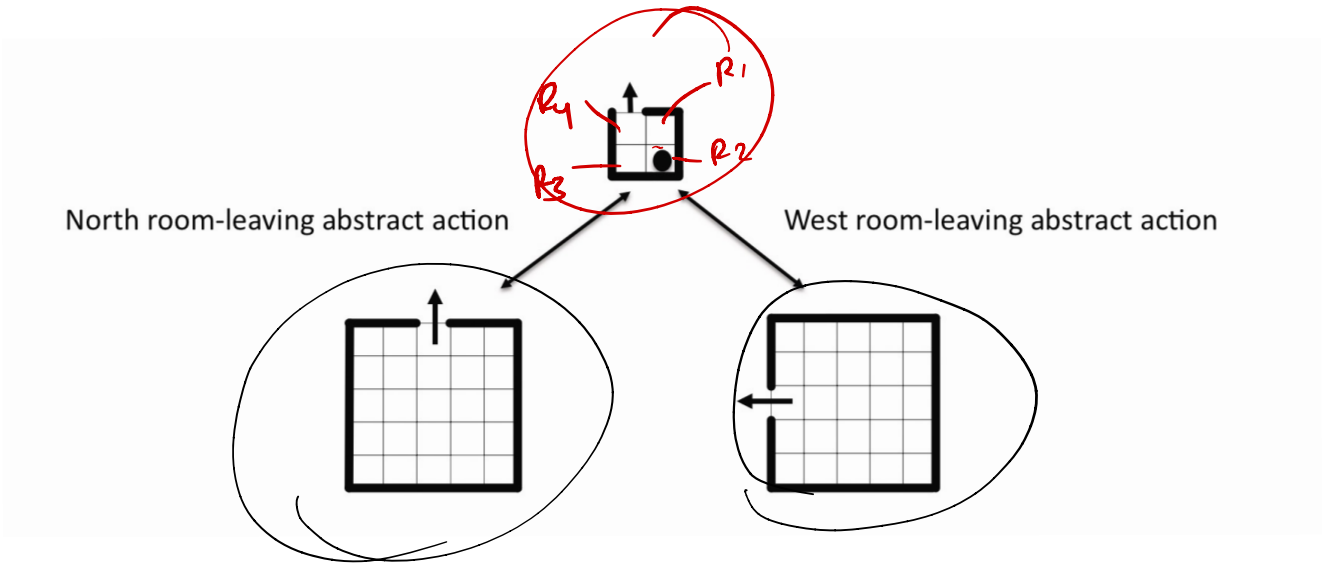
$$Q^{\pi}(S, a) = \sum_{S', N} T(S, a, S', N)\left[R(S, a, S', N) + \gamma^N Q^{\pi}(S', \pi(S'))\right]$$

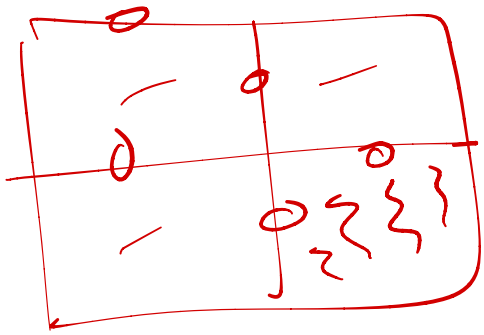
Optimal Q-fn

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



— Structure

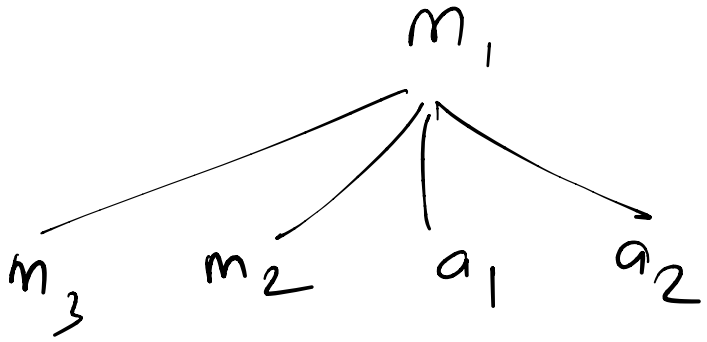

North room-leaving abstract action
West room-leaving abstract action



Optimality: → flat optimality ✓

→ hierarchical optimality ✓

→ Recursive optimality

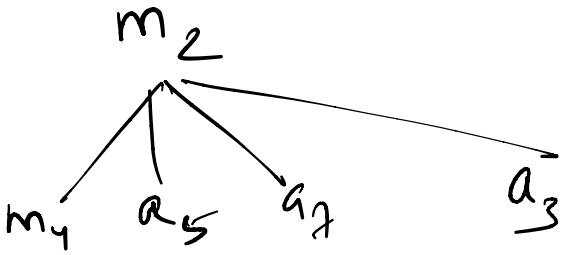


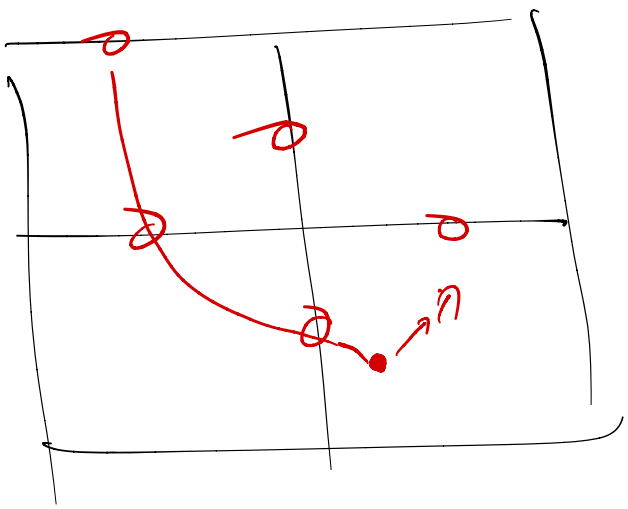
m_1 : $a_1\ a_2$ →

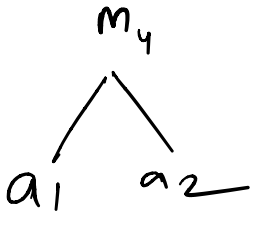
m_1 : $m_2\ a_1\ a_2$

↓

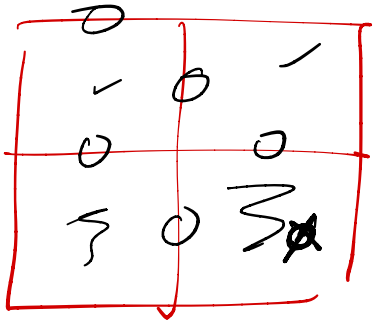
m_1 : $a_5\ a_1\ a_2$ →







① Options (Sutton et al. 1999) :



MDP: $\langle S, A, T, R \rangle$

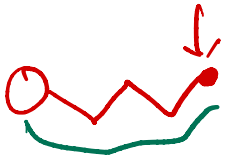
Option: $\langle I, \pi, \beta \rangle$

$I \subseteq S$ initiation set.

$\pi : S \times A \to [0,1]$ Policy.

$\beta : S^+ \to [0,1]$ is a termination Condn.

Markov options 

Semi-Markov options. 

Primitive actions are also options.

$a : \langle I = S, \pi(s,a) = 1.0, \beta(s) = 1 \rangle$ for all $s \in S$.

SMDP Q-learning :-

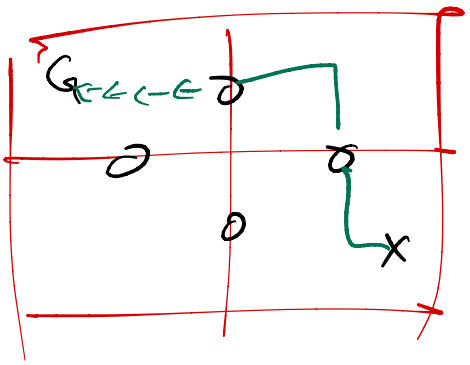
Cum. reward by end of the option

$$Q(s,o) \leftarrow Q(s,o) + \alpha \left[r + \gamma^{k} \max_{a \in O} Q(s',a) - Q(s,o) \right]$$

o - option

$k \rightarrow$ # of timesteps between s, s'.

r - Cumulative discounted reward over this time period



$O \Rightarrow$ { N, S, E, W,
exit thru N
exit thru S
exit thru E
exit thru W

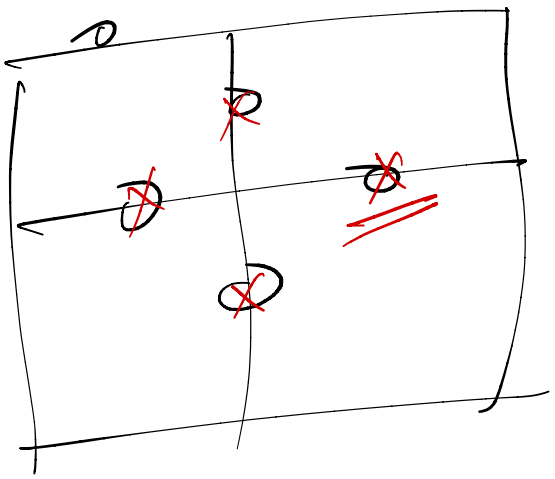
Intra-option Q learning :-

$$\tilde{Q}_O^*(s,o) = (1-\beta(s))\, Q_O^*(s,o) + \beta(s) \max_{o' \in O} Q_O^*(s,o')$$

$$Q_O^*(s,o) = \sum_{a \in A_s} \pi(s,a)\, E\left[r + \gamma \tilde{Q}_O^*(s',o) \mid s,a \right]$$

one-step TD update :

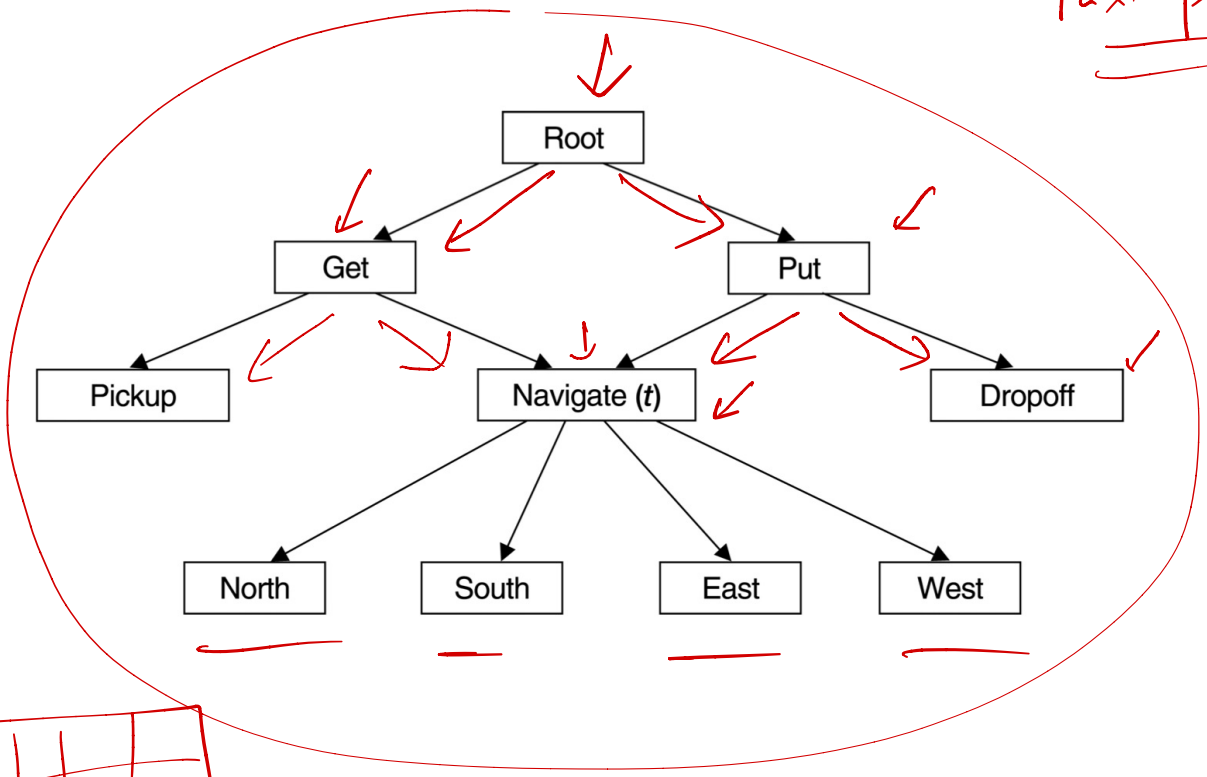
$$Q(s_t,o) \leftarrow Q(s_t,o) + \alpha \left[r_{t+1} + \gamma \tilde{Q}_O^*(s_{t+1},o) - Q(s_t,o) \right]$$

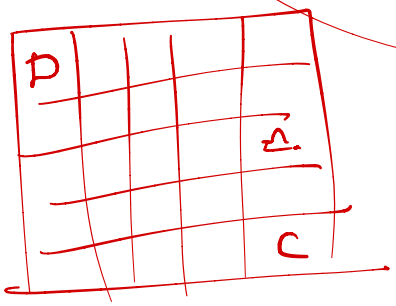


"bottleneck state".

MAXQ (Dietterich, 2000)

Taxi problem



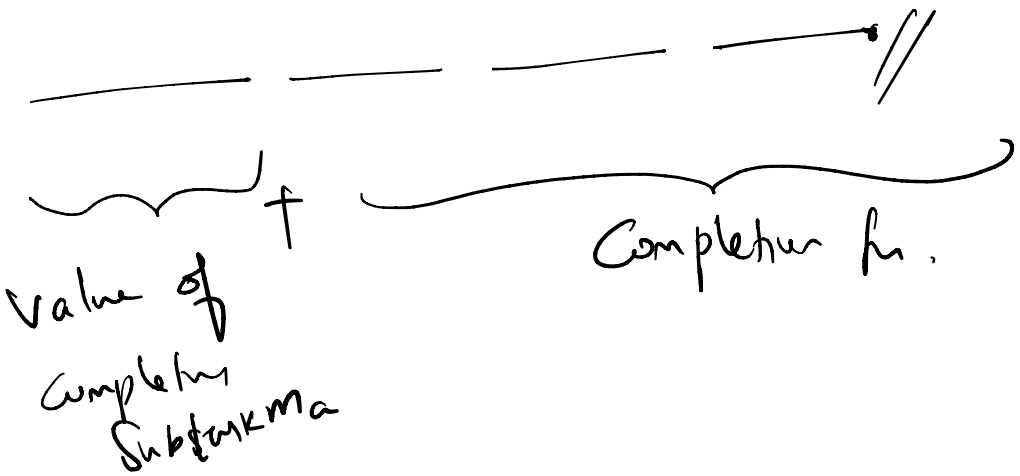


Task - decomposition.

Value fn decomposition.

m-subtask

$$Q^{\pi}(m, s, a) = \sum_{s', N} T(m, s, a, s', N) \left[R(m, s, a, s', N) + \gamma^{N} Q^{\pi}(m, s', \pi(s')) \right]$$

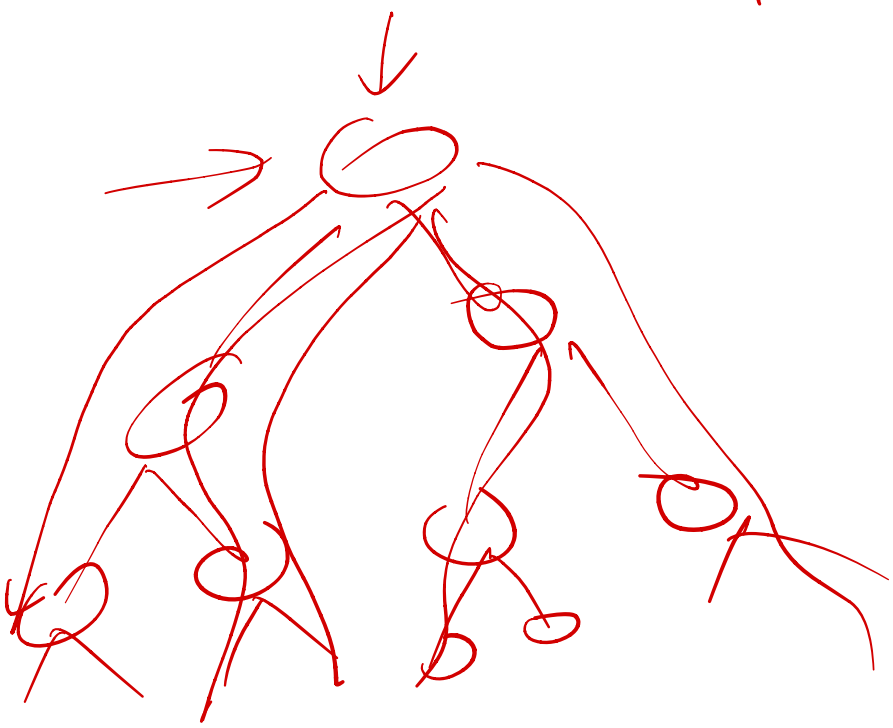


$$C^{\pi}(m, s, a) = \sum_{s', N} T(m, s, a, s', N) \gamma^{N} Q^{\pi}(m, s', \pi(s'))$$

$$Q^{\pi}(m, s, a) = V^{\pi}(m_a, s) + C^{\pi}(m, s, a)$$

$$V^{\pi}(m_a, s) = \begin{cases} \sum_{s'} T(s, a, s') R(s, a, s') & \text{if } a \text{ is pm.} \\ Q^{\pi}(m_a, s, \pi(s)) & \text{if } a \text{ is abt.} \end{cases}$$

$$Q^{\pi}(m_0, s, \pi(s)) = V^{\pi}(m_k, s) + C^{\pi}(m_{k-1}, s, a_{k-1})$$
$$\ldots + C^{\pi}(m_1, s, a_1) + C^{\pi}(m_0, s, a_0)$$



Max Q.

$$V^{*}(m_a, s) = \max_{a'} Q^{*}(m_a, s, a')$$

Require: V, C, A, learning rate α
if m is a primitive action **then**
  execute m, receive reward r, and observe the result state s'
  $V(m,s) \leftarrow (1-\alpha)V(m,s) + \alpha r$
  return 1
else
  $count = 0$
  while m has not terminated **do**
    choose an exploration action a
    $N = MAXQ(a,s)$
    observe result state s'
    $\langle V(j^{greedy},s'),a_{j^{greedy}} \rangle \leftarrow Evaluate(m,s')$
    $C(m,s,a) \leftarrow (1-\alpha)C(m,s,a) + \alpha\gamma^N V(j^{greedy},s')$
    $count \leftarrow count + N$
    $s = s'$
    return *count*

Algorithm 19. $MAXQ(m,s)$ [Dietterich (2000)]