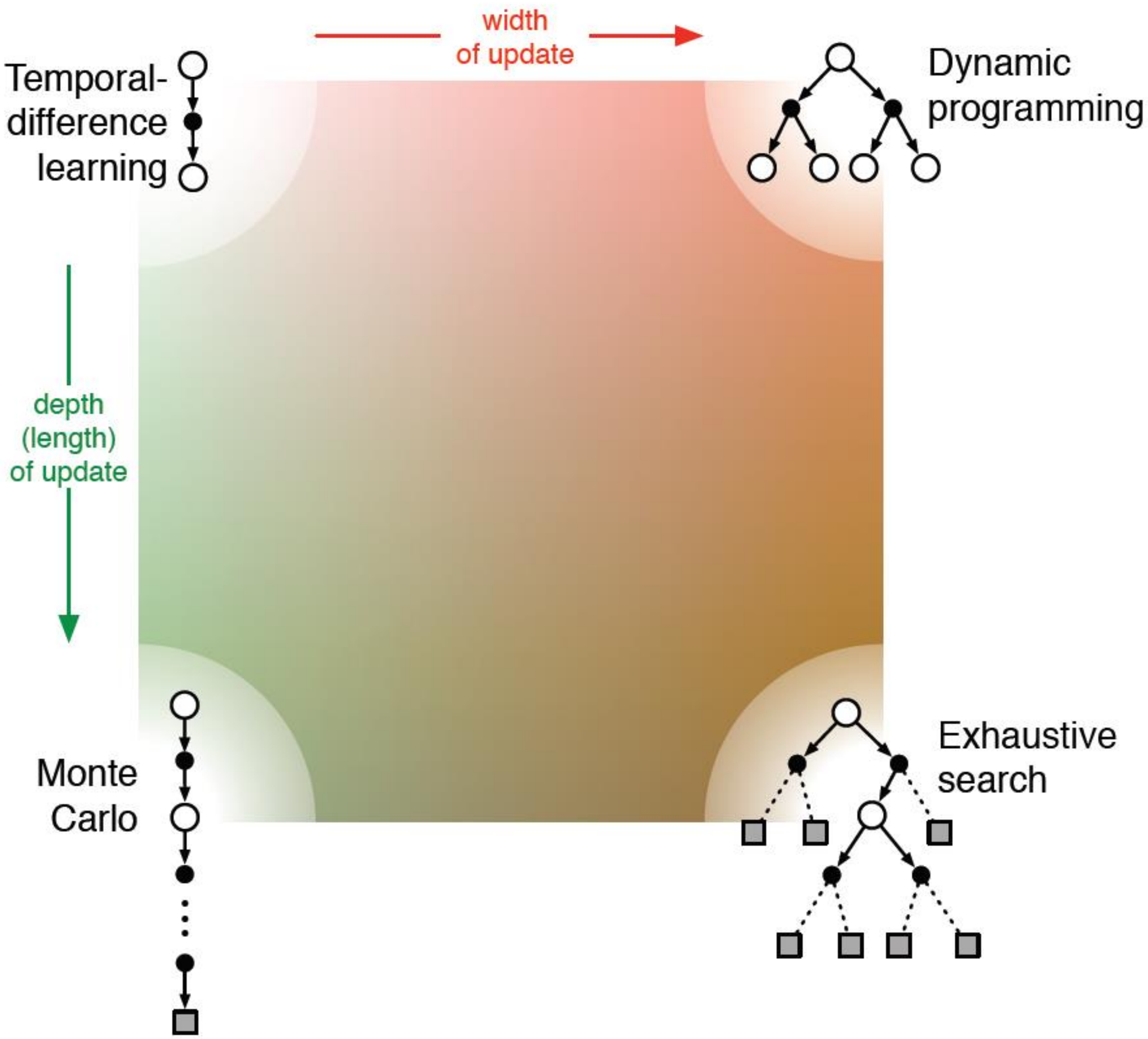
Temporal-Difference Learning

Prof. Kuan-Ting Lai

2020/5/4

Temporal Difference (TD) Learning

- Learn from the experience of few time steps
- TD is model-free
- TD methods learn from a guess from a guess (bootstrap)
- TD combines the sampling of MC with the bootstrapping of DP
- Most novel idea in reinforcement learning



Sutton, Richard S.; Barto, Andrew G.. Reinforcement Learning (Adaptive Computation and Machine Learning series) (p. 189)

Monte-Carlo vs. Temporal-Difference

- MC waits until end of the episode and uses Return G as target

$$V(S_t) \leftarrow V(S_t) + \alpha[G_t - V(S_t)]$$

- TD only needs few time steps and uses observed reward R_{t+1}

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)]$$

TD Error and MC Error

TD Error : $\delta_t = R_{t+1} + \gamma V(S_{t+1}) - V(S_t)$

MC Error :

$$G_t - V(S_t) = R_{t+1} + \gamma G_{t+1} - V(S_t) + \gamma V(S_{t+1}) - \gamma V(S_{t+1})$$

δ_t

$$= \delta_t + \gamma [G_{t+1} - V(S_{t+1})]$$

$$= \delta_t + \gamma \delta_{t+1} + \gamma^2 \delta_{t+2} + \cdots + \gamma^{T-t}\left(\overset{0}{\overset{\parallel}{G_T}} - \overset{0}{\overset{\parallel}{V(S_T)}}\right)$$

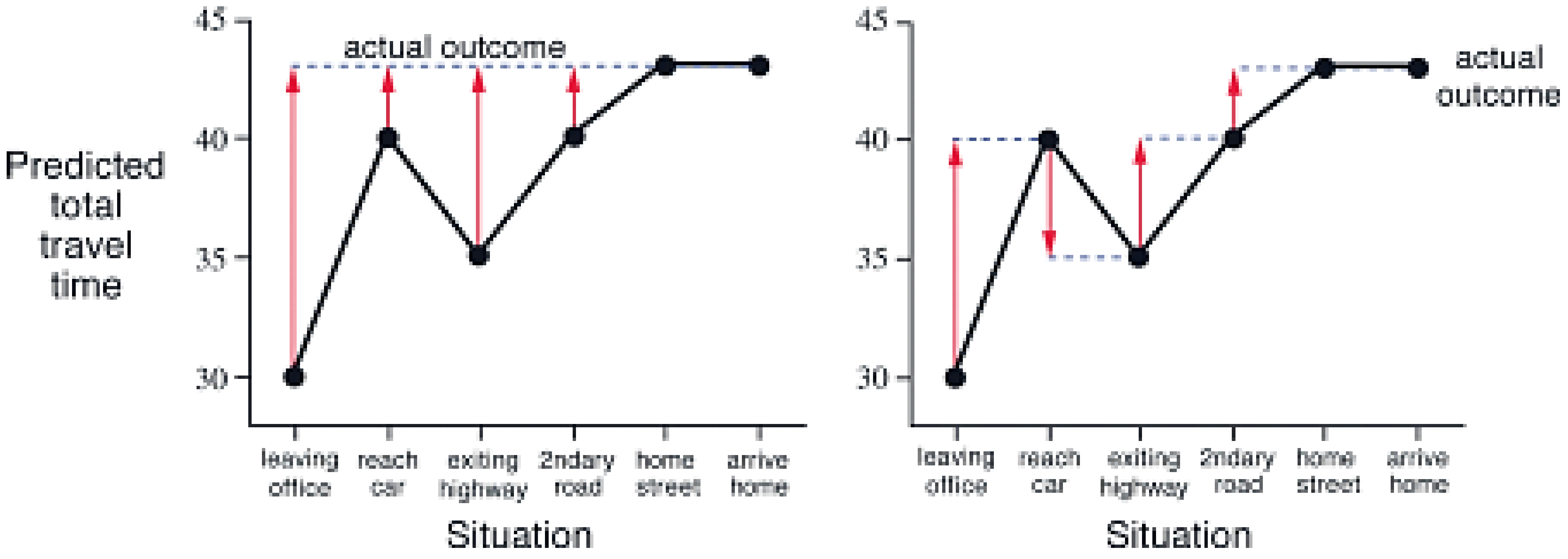
$$= \sum_{k=t}^{T-1} \gamma^{k-t} \delta_k$$

Example: Driving Home

- Estimate the time of arriving home

	Rewards	Values	
State	*Elapsed Time (minutes)*	*Predicted Time to Go*	*Predicted Total Time*
leaving office, friday at 6	0	30	30
reach car, raining	5	35	40
exiting highway	20	15	35
2ndary road, behind truck	30	10	40
entering home street	40	3	43
arrive home	43	0	43

Example: Driving Home (TD vs. MC)

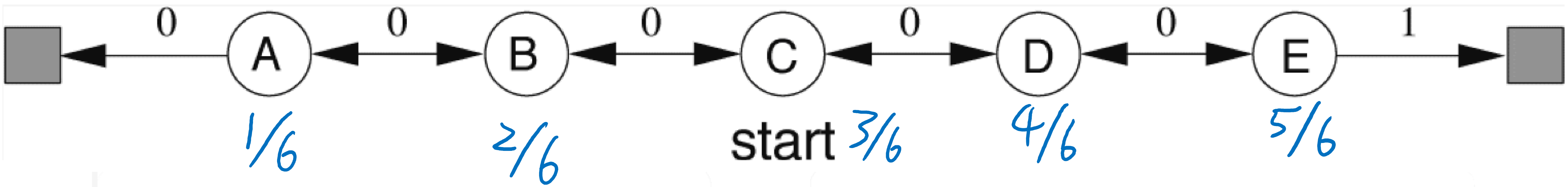


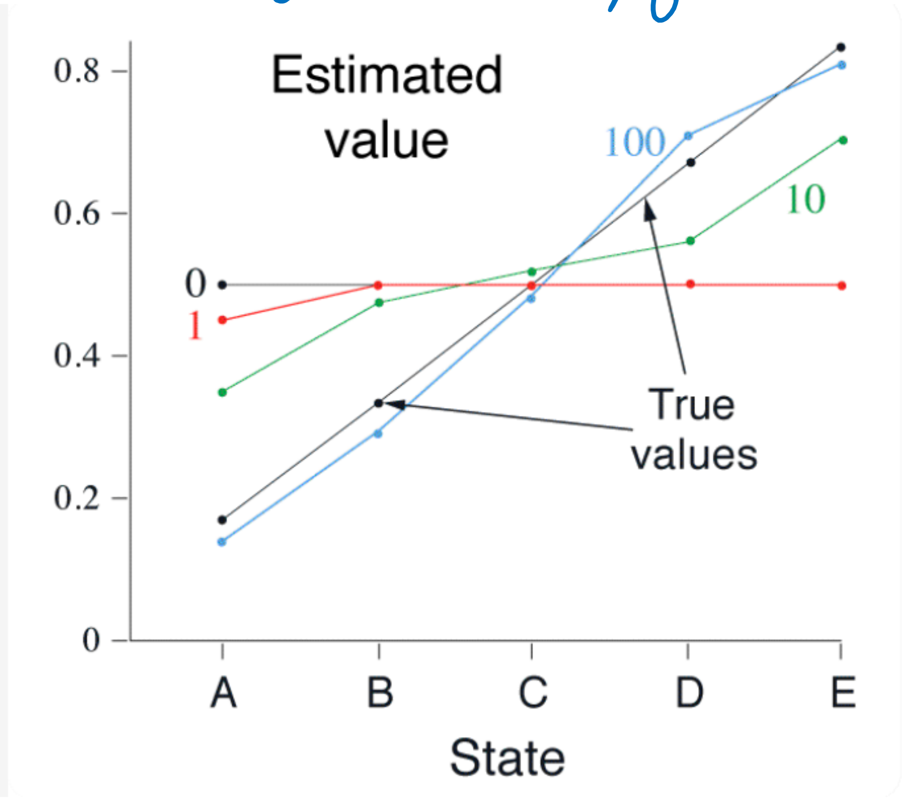
Advantages of TD

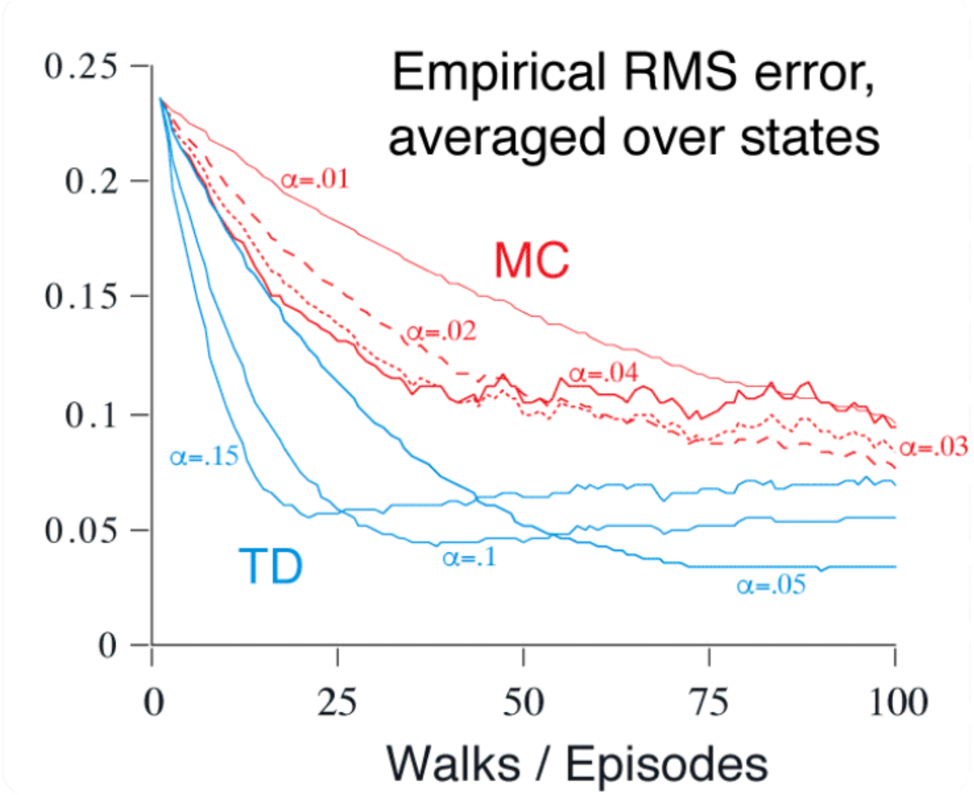
- TD vs. Dynamic Programming
  - Do not require knowledge of all states
- TD vs. Monte Carlo
  - Some applications have very long episodes

Example: Random Walk

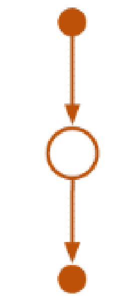
- Markov Reward Process



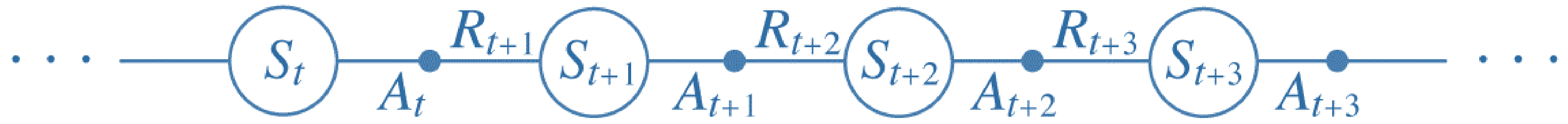




On-policy TD: SARSA

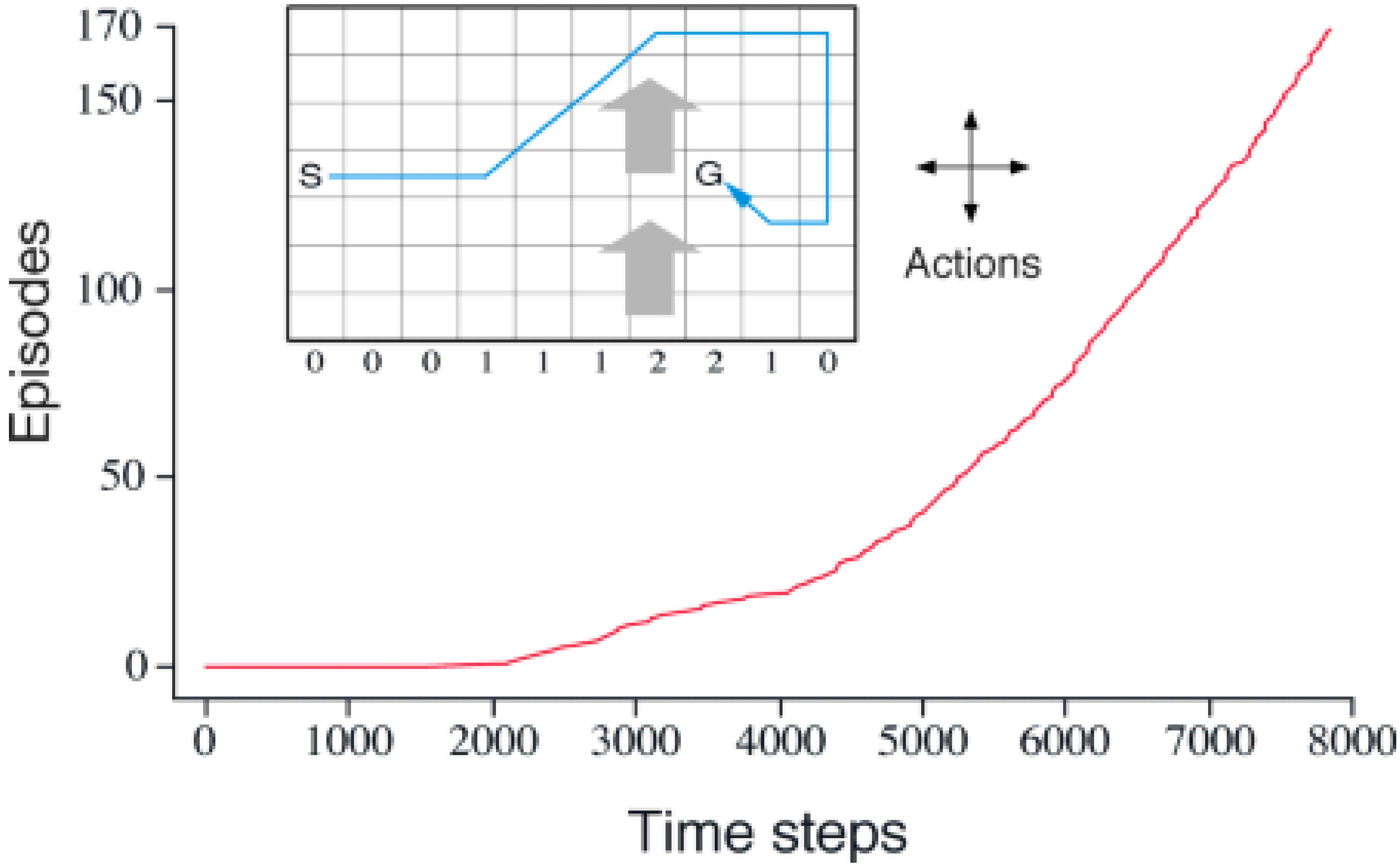


- Use state-action function Q
- One-step TD(0): $S_t, A_t, R_t, S_{t+1}, A_{t+1}$



$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \Big]$$

Windy Grid World



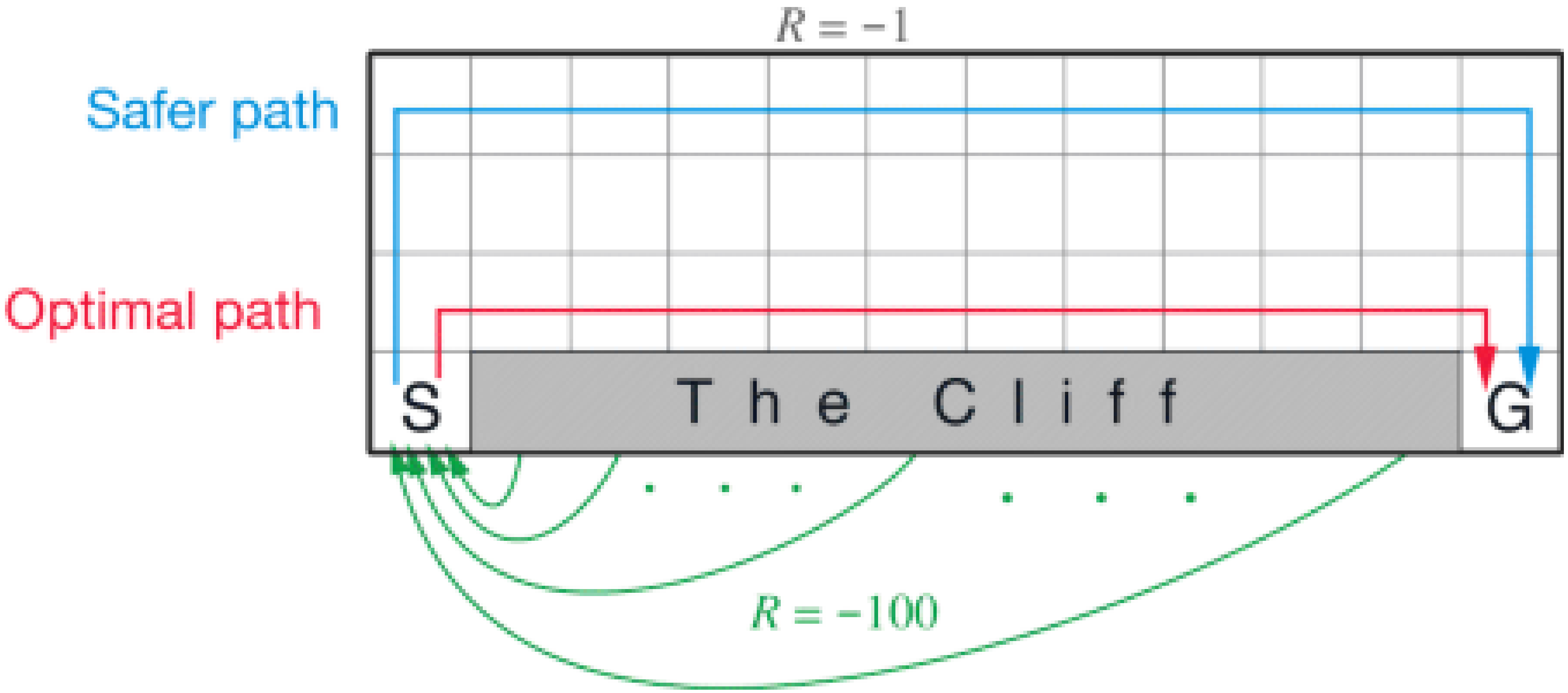
Off-policy TD: Q-Learning

- Target policy: Greedy policy

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \Big[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \Big]$$

Learning to Walk Cliff

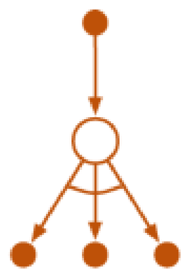
- Agent sometimes falls to the cliff due to ε-greedy
- SARSA vs. Q-learning



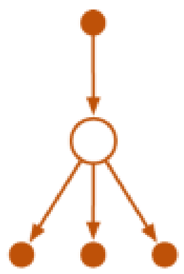
Expected SARSA

- Calculate expected value of next state

$$
\begin{aligned}
Q(S_t, A_t) &\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \mathbb{E}_\pi[Q(S_{t+1}, A_{t+1}) \mid S_{t+1}] - Q(S_t, A_t)\Big] \\
&\leftarrow Q(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma \sum_a \pi(a|S_{t+1}) Q(S_{t+1}, a) - Q(S_t, A_t)\Big],
\end{aligned}
$$

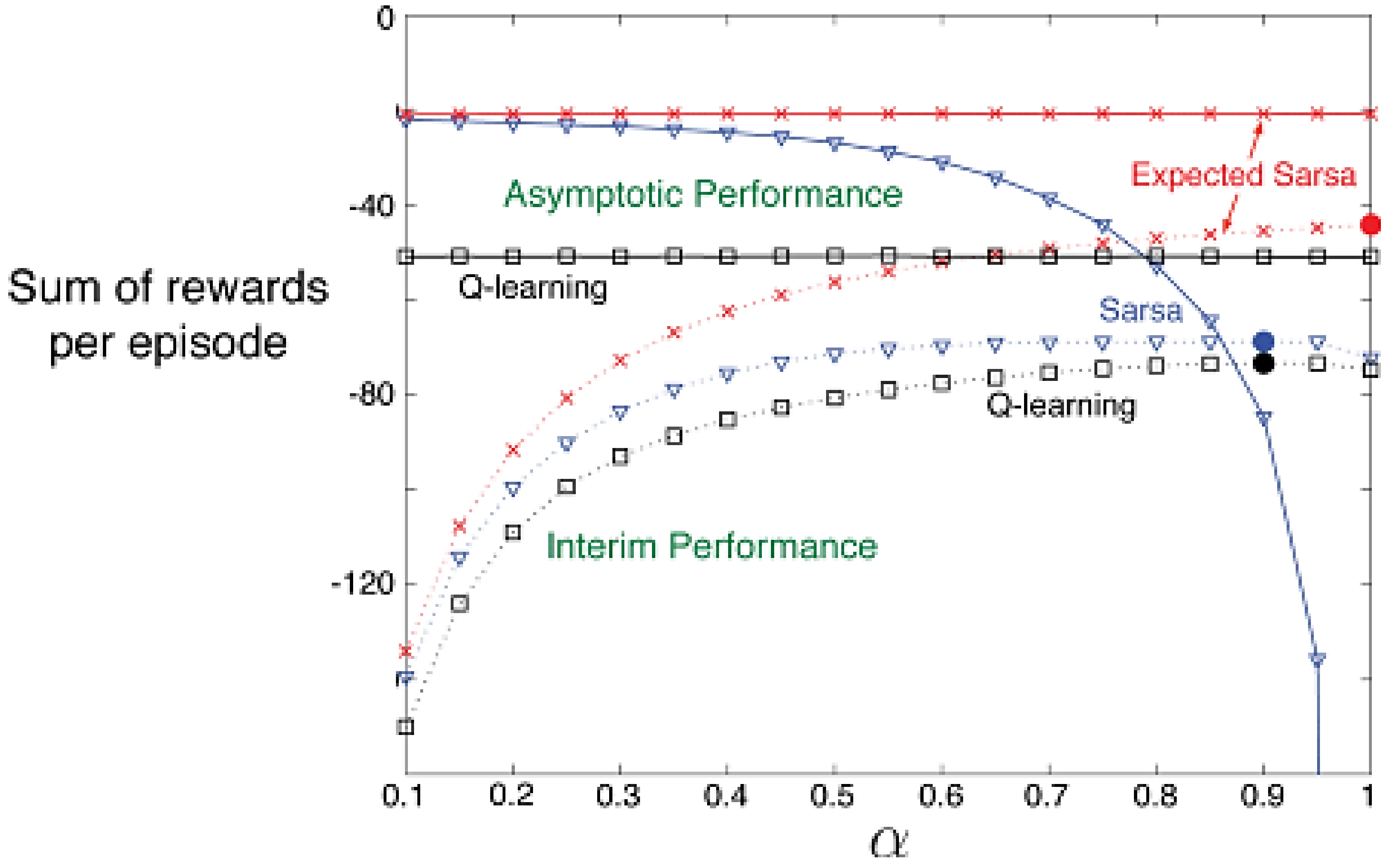


Q-learning



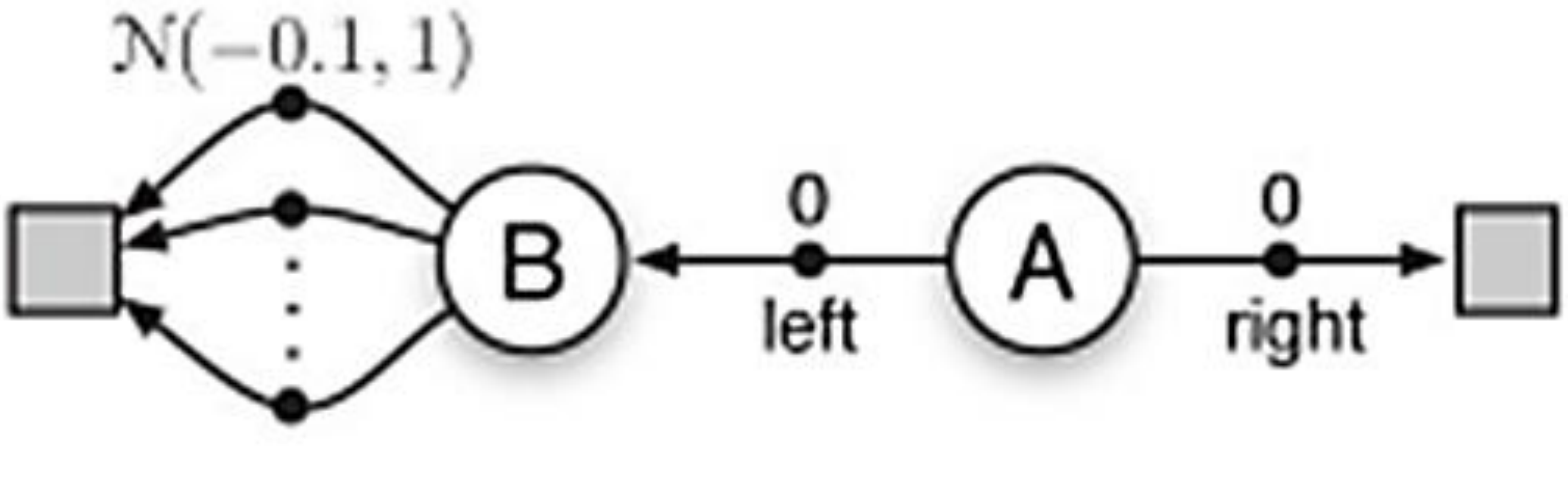
Expected Sarsa

Comparison on Cliff Walking



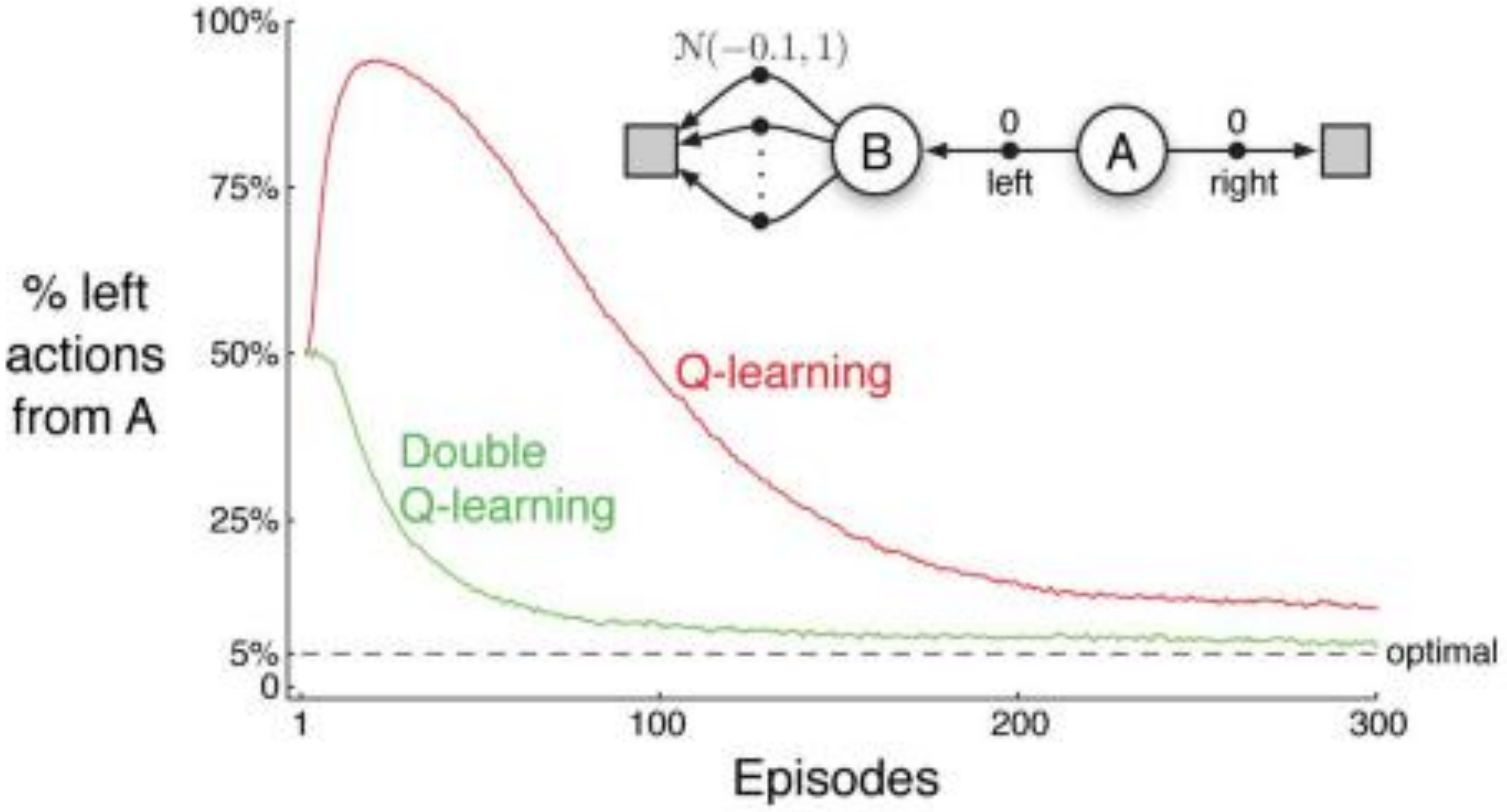
Maximization Bias

- State A has 0 reward
- State B reward is a normal distribution (mean=-0.1, variance=1)
- TD may favor B



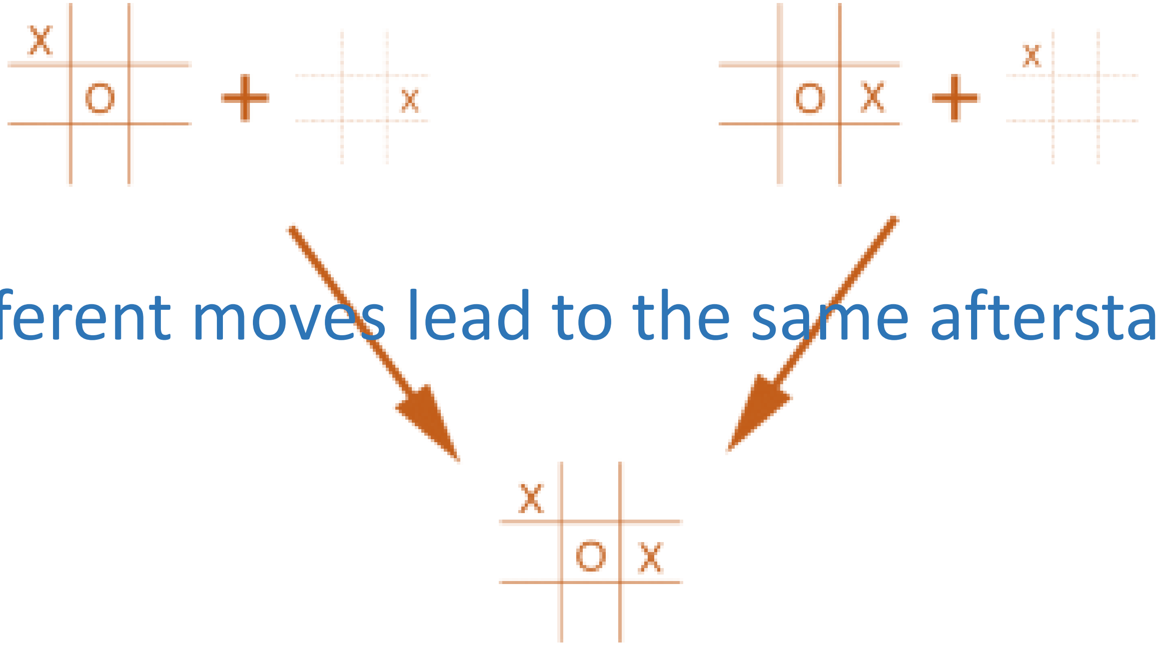
Double Q-Learning

$$Q_1(S_t, A_t) \leftarrow Q_1(S_t, A_t) + \alpha\Big[R_{t+1} + \gamma Q_2\big(S_{t+1}, \arg\max_a Q_1(S_{t+1}, a)\big) - Q_1(S_t, A_t)\Big]$$



Special Value Function: Afterstate

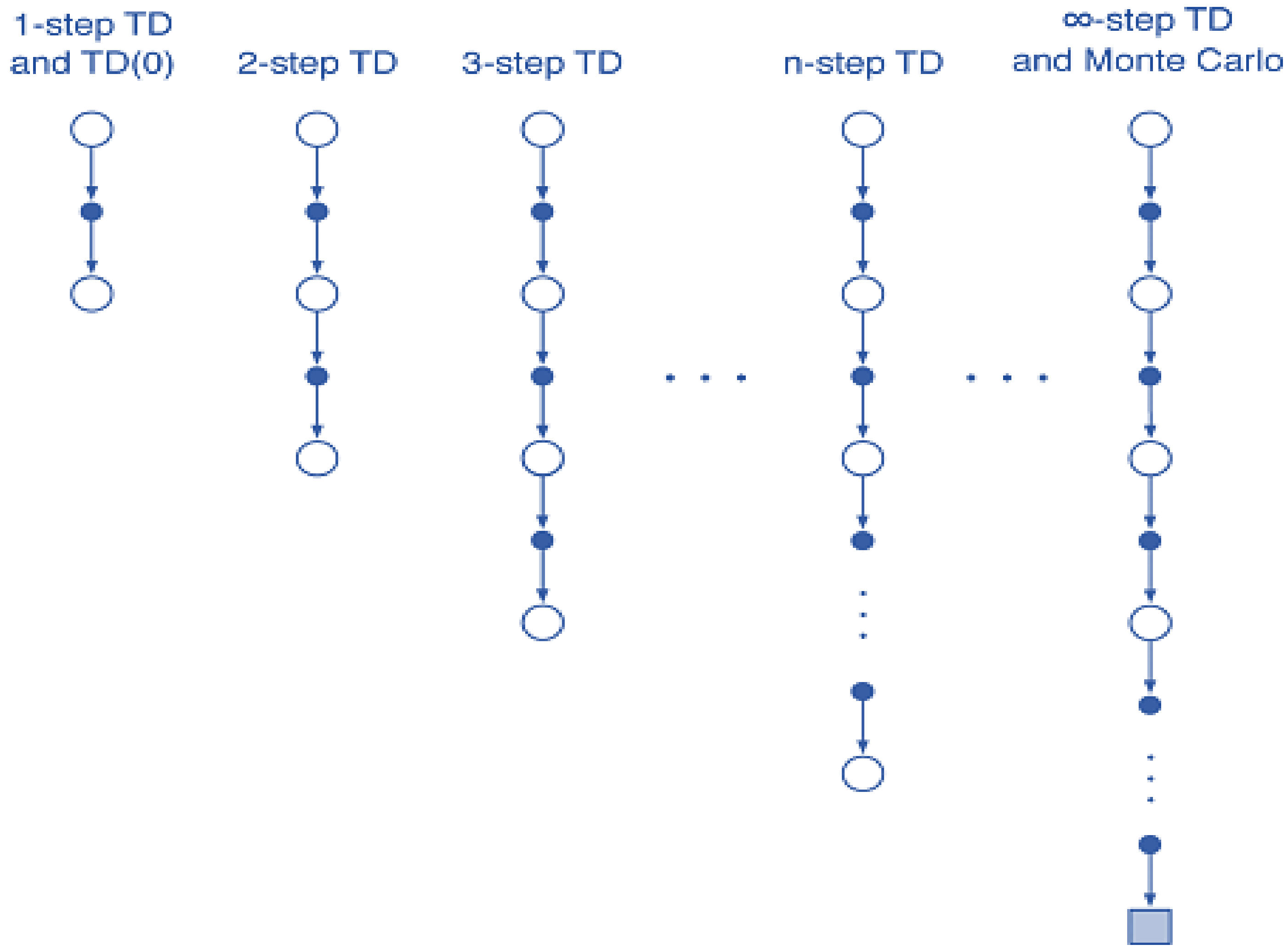
- State-value function used in tic-tac-toe evaluates board positions after the agent has made its move



Two different moves lead to the same afterstate result

n-step TD Prediction

- n-step bootstrapping
- Combine 1-step TD with Monte Carlo



Formulating n-step TD

$$G_{t:t+1} \doteq R_{t+1} + \gamma V_t(S_{t+1})$$

$$G_{t:t+2} \doteq R_{t+1} + \gamma R_{t+2} + \gamma^2 V_{t+1}(S_{t+2})$$

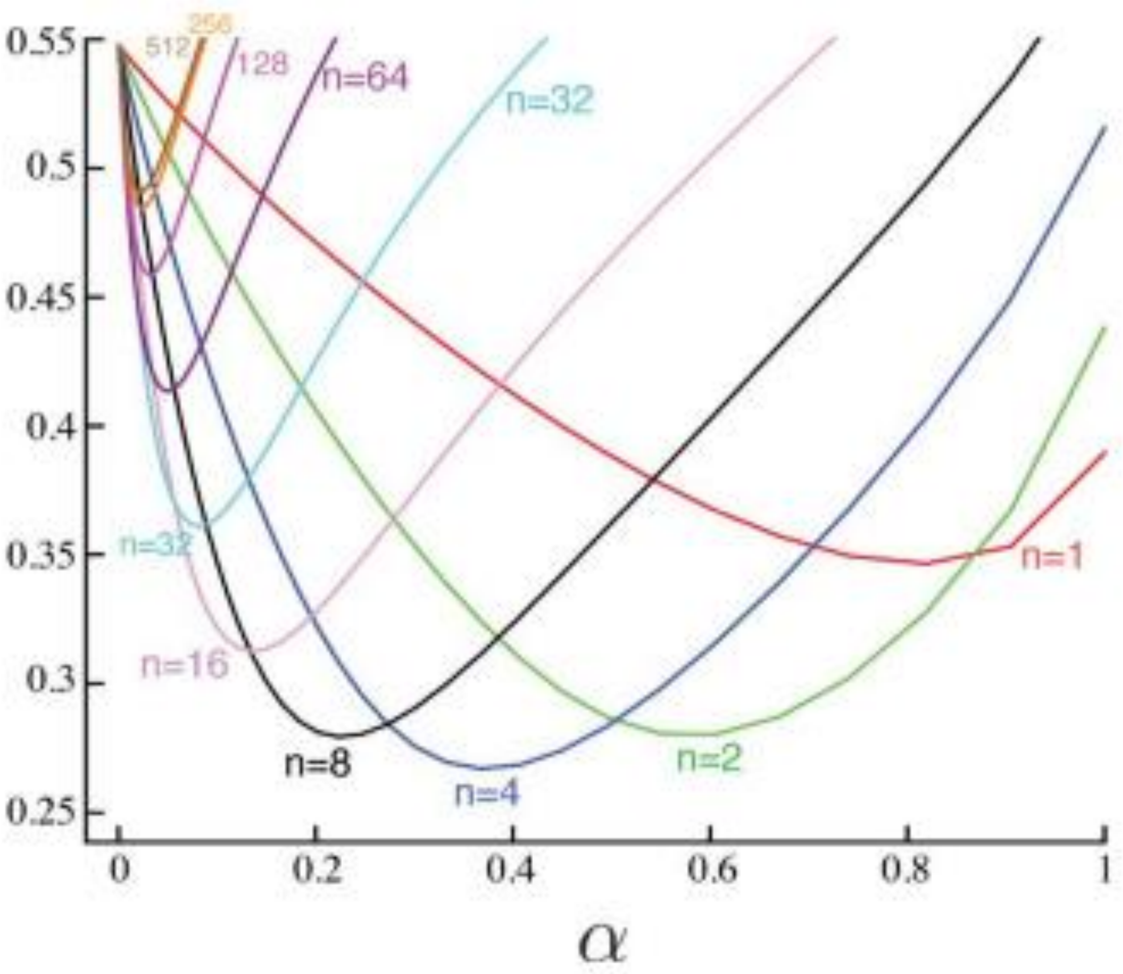
$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n V_{t+n-1}(S_{t+n})$$

$$\Downarrow$$

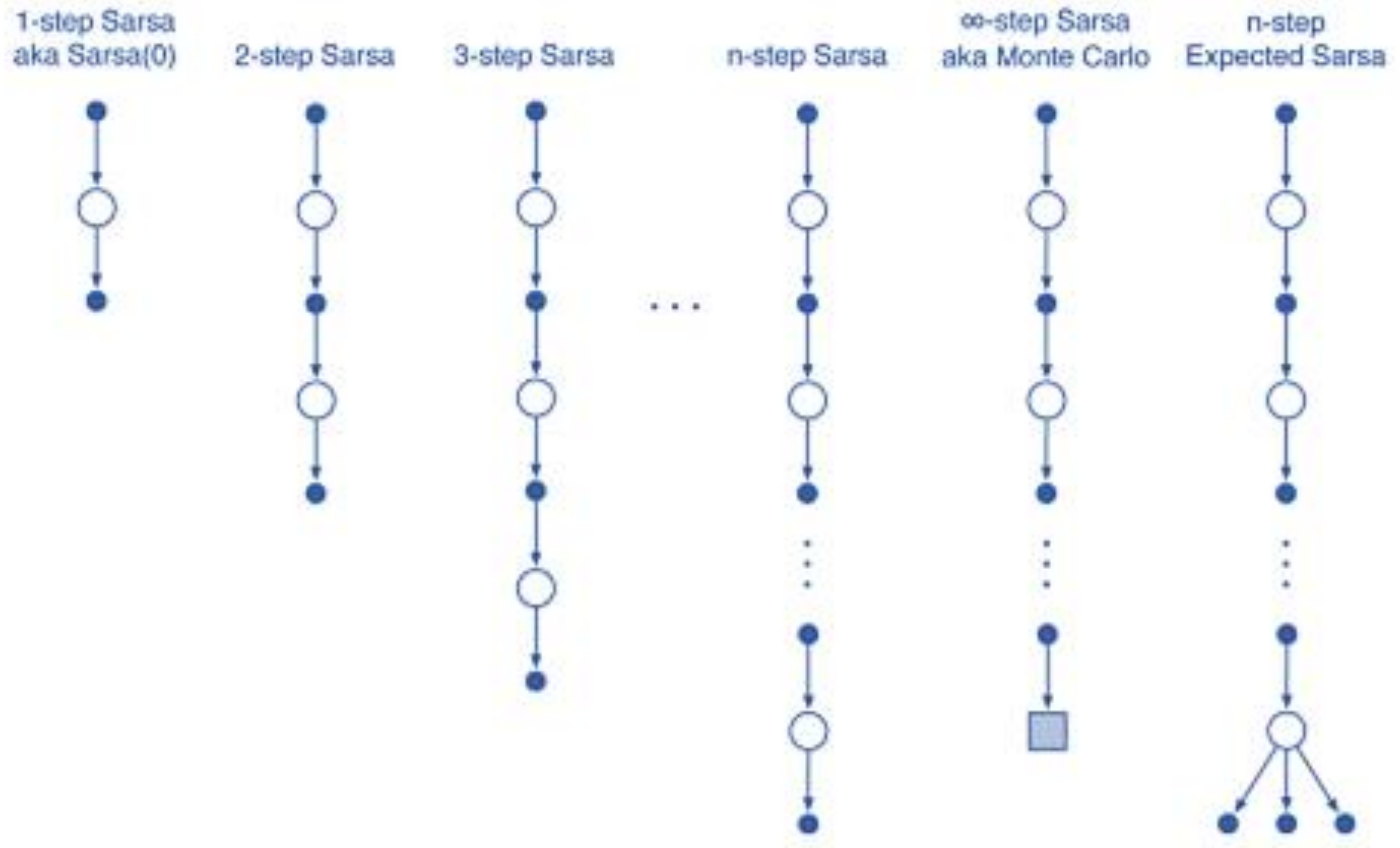
$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha\left[G_{t:t+n} - V_{t+n-1}(S_t)\right]$$

N-step TD on 19-state random walk

Average RMS error over 19 states and first 10 episodes



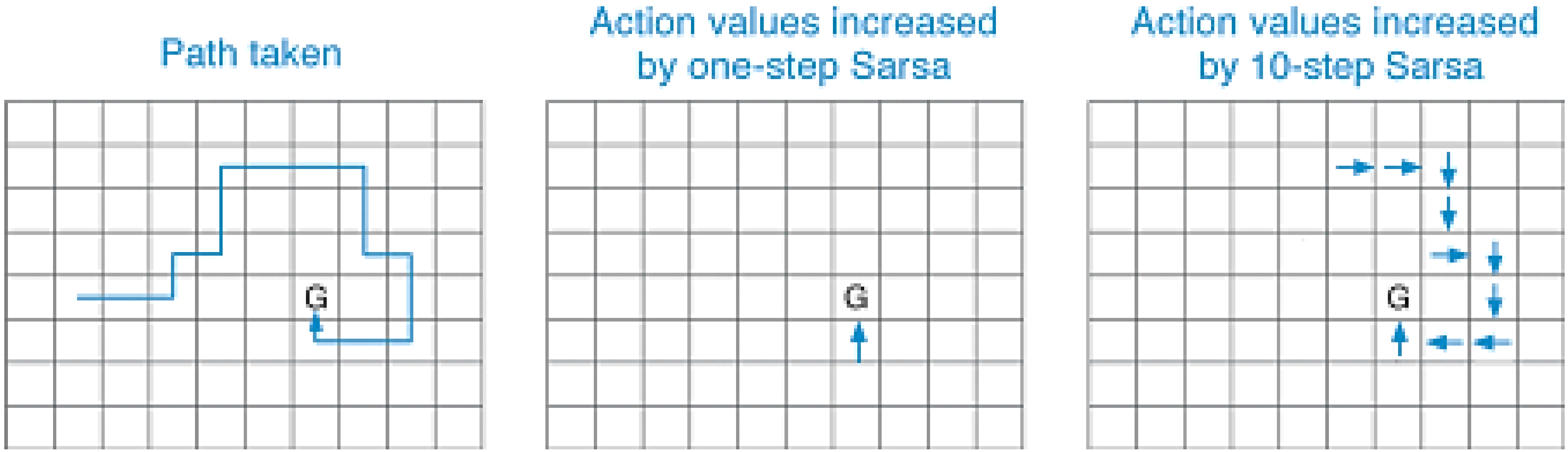
N-step SARSA


1-step Sarsa
aka Sarsa(0)
2-step Sarsa
3-step Sarsa
n-step Sarsa
∞-step Sarsa
aka Monte Carlo
n-step
Expected Sarsa

N-step SARSA

$$G_{t:t+n} \doteq R_{t+1} + \gamma R_{t+2} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n Q_{t+n-1}(S_{t+n}, A_{t+n}), \quad n \geq 1, 0 \leq t < T-n$$

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha \left[G_{t:t+n} - Q_{t+n-1}(S_t, A_t) \right]$$



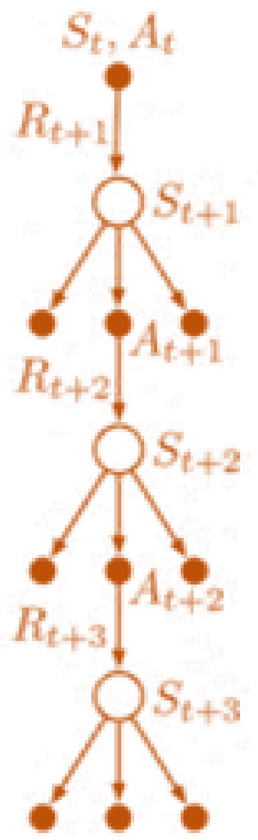
N-step Tree Backup

- Off-policy learning without importance sampling

$$G_{t:t+1} \doteq R_{t+1} + \gamma \sum \pi(a|S_{t+1}) Q_t(S_{t+1}, a)$$

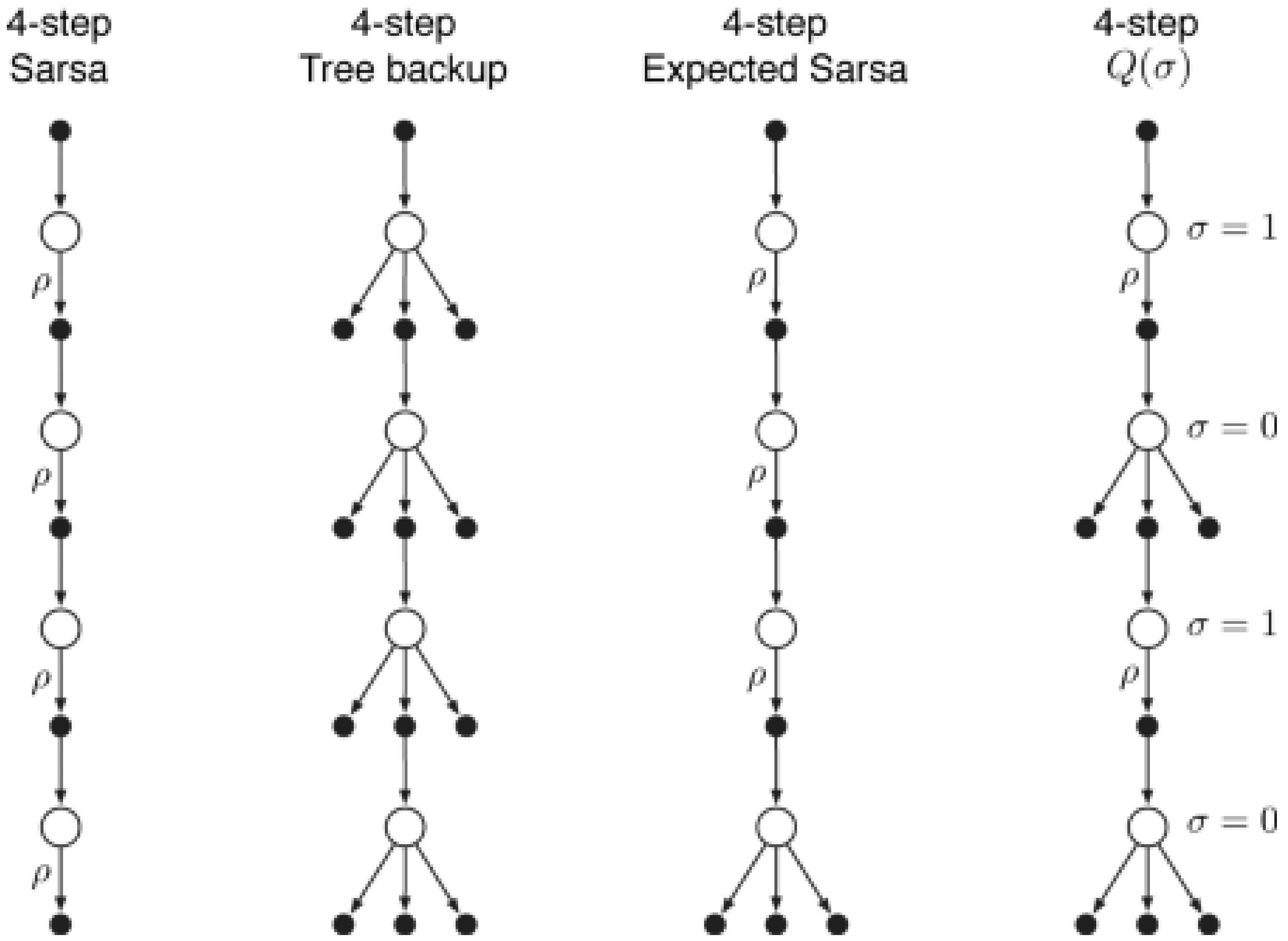
$$\begin{aligned} G_{t:t+2} \doteq R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+1}(S_{t+1}, a) \\ + \gamma \pi(A_{t+1}|S_{t+1}) \Big(R_{t+2} + \gamma \sum_a \pi(a|S_{t+2}) Q_{t+1}(S_{t+2}, a) \Big) \\ = R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+1}(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:t+2}, \end{aligned}$$

$$G_{t:t+n} \doteq R_{t+1} + \gamma \sum_{a \neq A_{t+1}} \pi(a|S_{t+1}) Q_{t+n-1}(S_{t+1}, a) + \gamma \pi(A_{t+1}|S_{t+1}) G_{t+1:t+n}$$



the 3-step tree-backup update

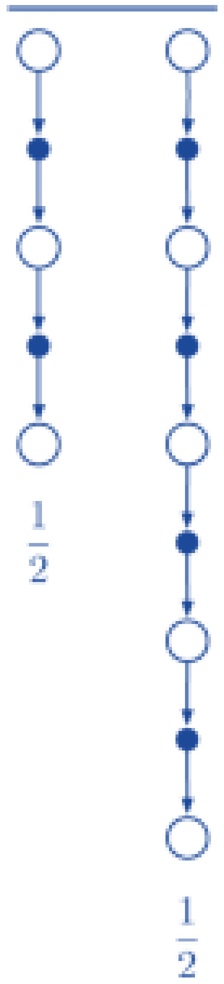
A Unifying Algorithm n-step Q(σ)



The λ Return

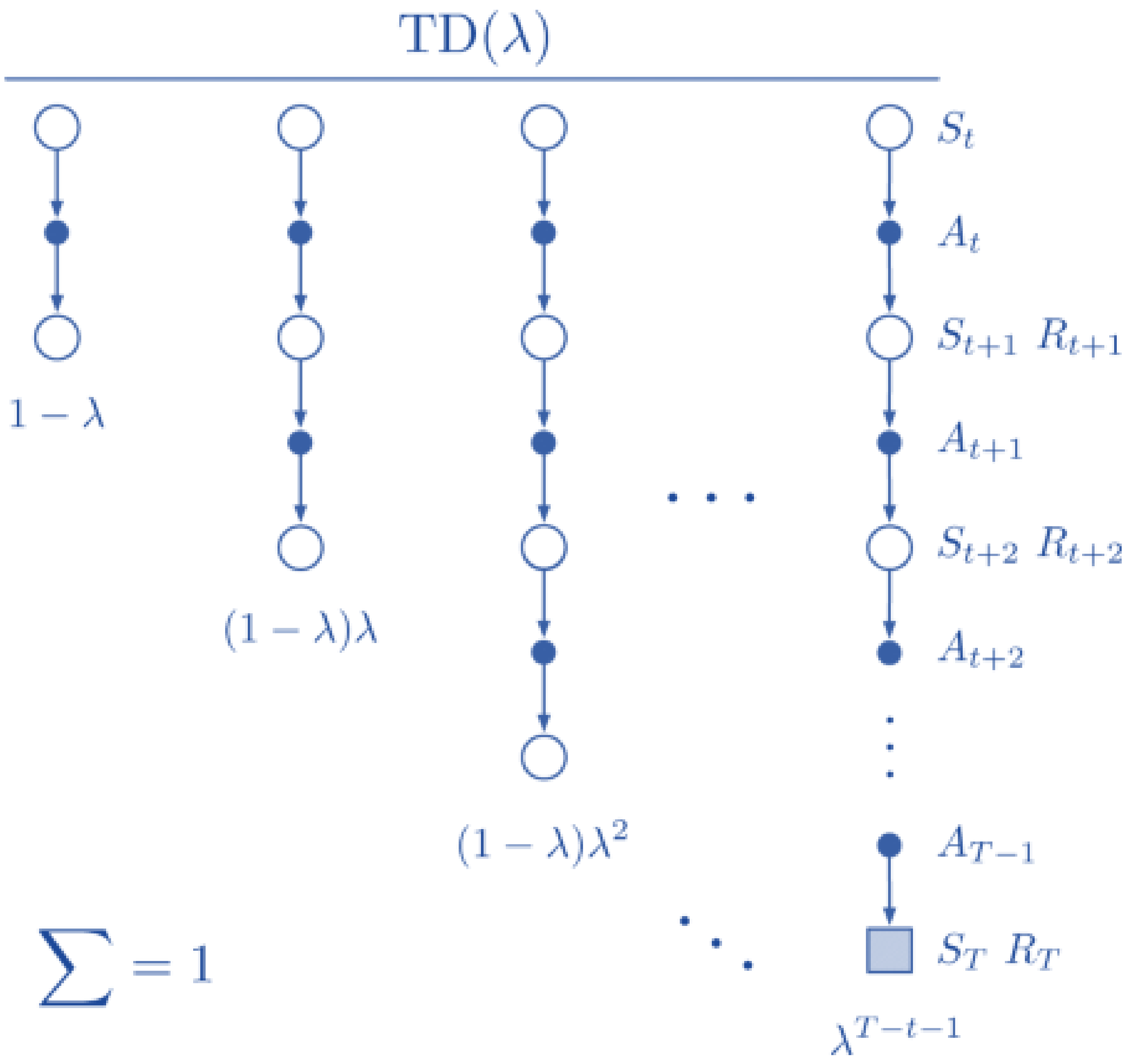
- Weighted sum of different n-step returns
- Compound update
- Sum of weights need to be 1
- Example:

$$\frac{1}{2}G_{t:t+2} + \frac{1}{2}G_{t:t+4}$$



TD(λ)

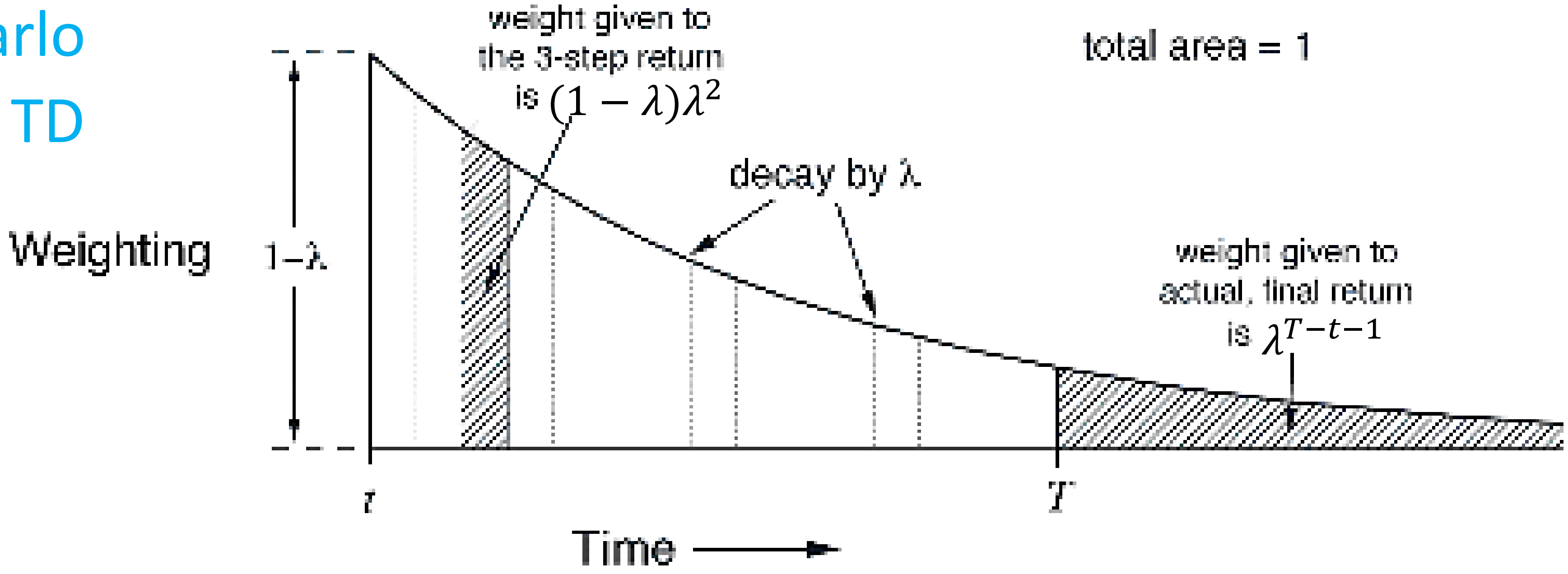
$$G_t^{\lambda} \leftarrow (1-\lambda) \sum_{n=1}^{\infty} \lambda^{n-1} G_{t:t+n}$$



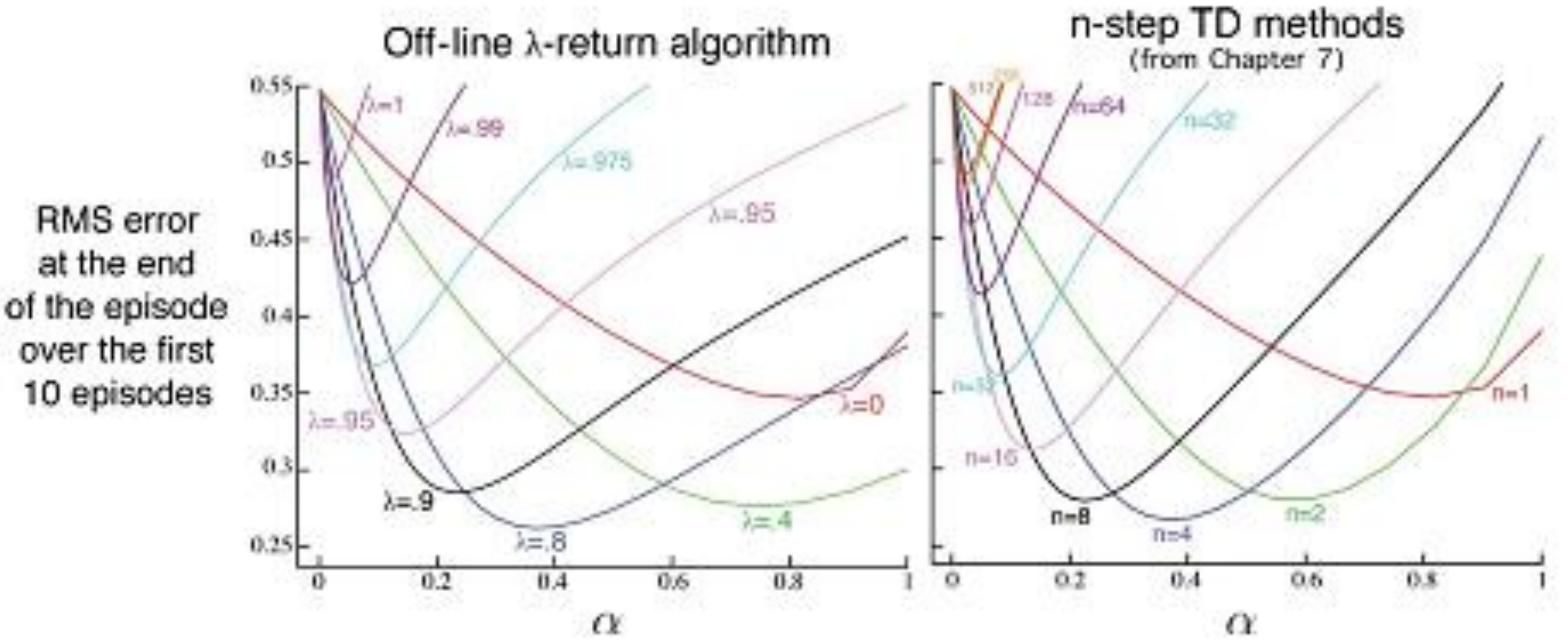
Weighting of each n-step Returns in λ Return

$$G_t^{\lambda} = (1-\lambda)\sum_{n=1}^{T-t-1} \lambda^{n-1} G_{t:t+n} + \lambda^{T-t-1} G_t$$

- λ=1: Monte Carlo
- λ=0: One-step TD

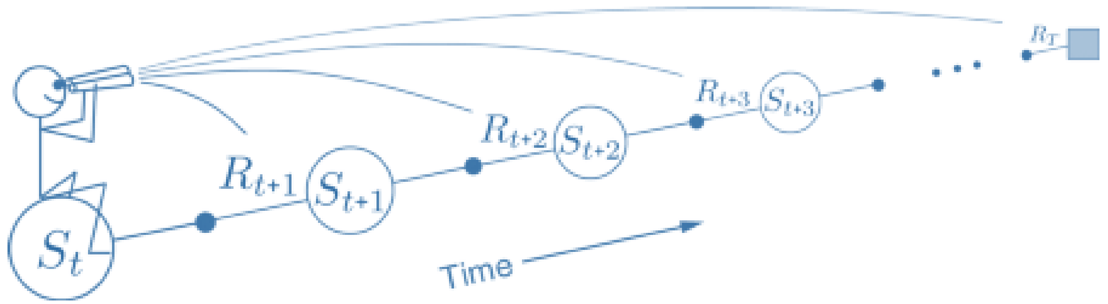


λ-return vs. n-step TD

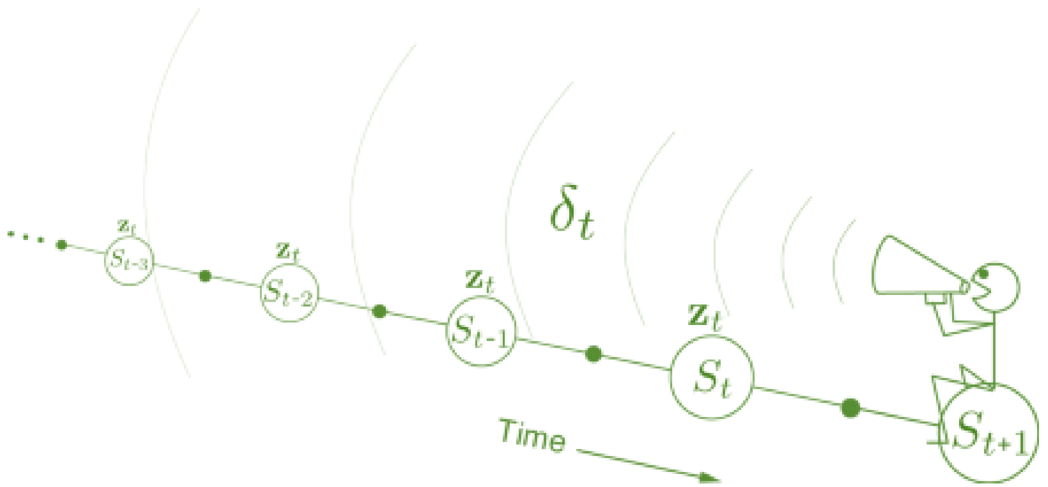


Forward View vs. Backward View

- N-step TD (and DP) are based on forward view



- TD(λ) is oriented backward in time



Reference

1. David Silver, Lecture 4: Model-Free Prediction
2. Chapter 6, 7 and 12, Richard S. Sutton and Andrew G. Barto, “Reinforcement Learning: An Introduction,” 2nd edition, Nov. 2018