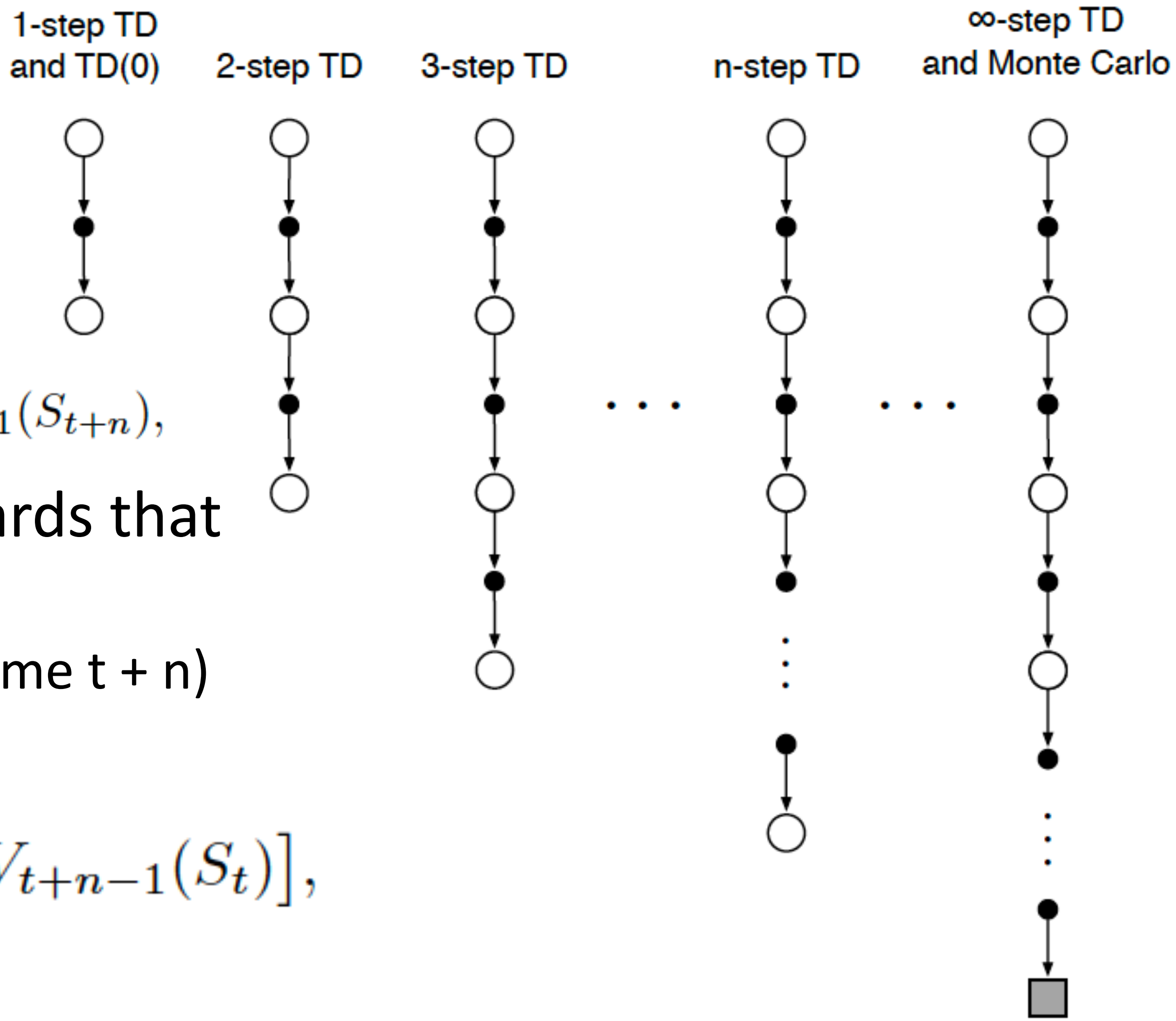
# n-step Bootstrapping

- n-step methods span a spectrum with MC methods at one end and one-step TD methods at the other
- Frees from the tyranny of the single time step in TD
  - same time step determines when actions are selected and time interval of bootstrapping

# n-step TD Prediction




- n-step return

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

- n-step returns involves future rewards that are not immediately available
  - update for time t (actually made at time t + n)
- The value function update is:

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

- Error reduction property ensures convergence

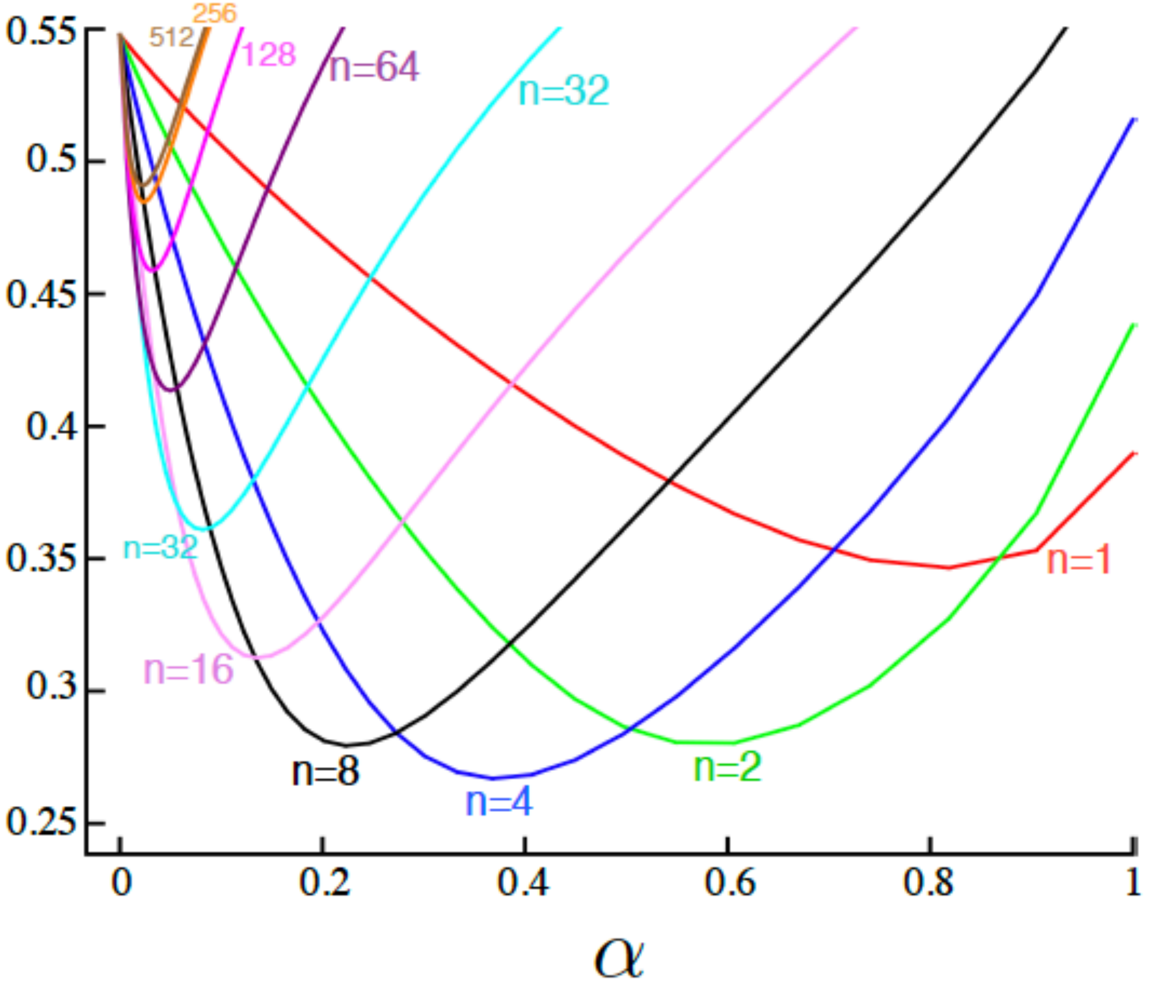
**$n$-step TD for estimating $V \approx v_\pi$**

Input: a policy $\pi$
Algorithm parameters: step size $\alpha \in (0, 1]$, a positive integer $n$
Initialize $V(s)$ arbitrarily, for all $s \in \mathcal{S}$
All store and access operations (for $S_t$ and $R_t$) can take their index mod $n + 1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$ :
  |  If $t < T$, then:
  |    Take an action according to $\pi(\cdot|S_t)$
  |    Observe and store the next reward as $R_{t+1}$ and the next state as $S_{t+1}$
  |    If $S_{t+1}$ is terminal, then $T \leftarrow t + 1$
  |  $\tau \leftarrow t - n + 1$   ($\tau$ is the time whose state's estimate is being updated)
  |  If $\tau \geq 0$:
  |    $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  |    If $\tau + n < T$, then: $G \leftarrow G + \gamma^n V(S_{\tau+n})$   $(G_{\tau:\tau+n})$
  |    $V(S_\tau) \leftarrow V(S_\tau) + \alpha \left[G - V(S_\tau)\right]$
  Until $\tau = T - 1$

# Random Walk with 19 states

Average RMS error over 19 states and first 10 episodes



# n-step Sarsa

- n-step returns in terms of estimated action values

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

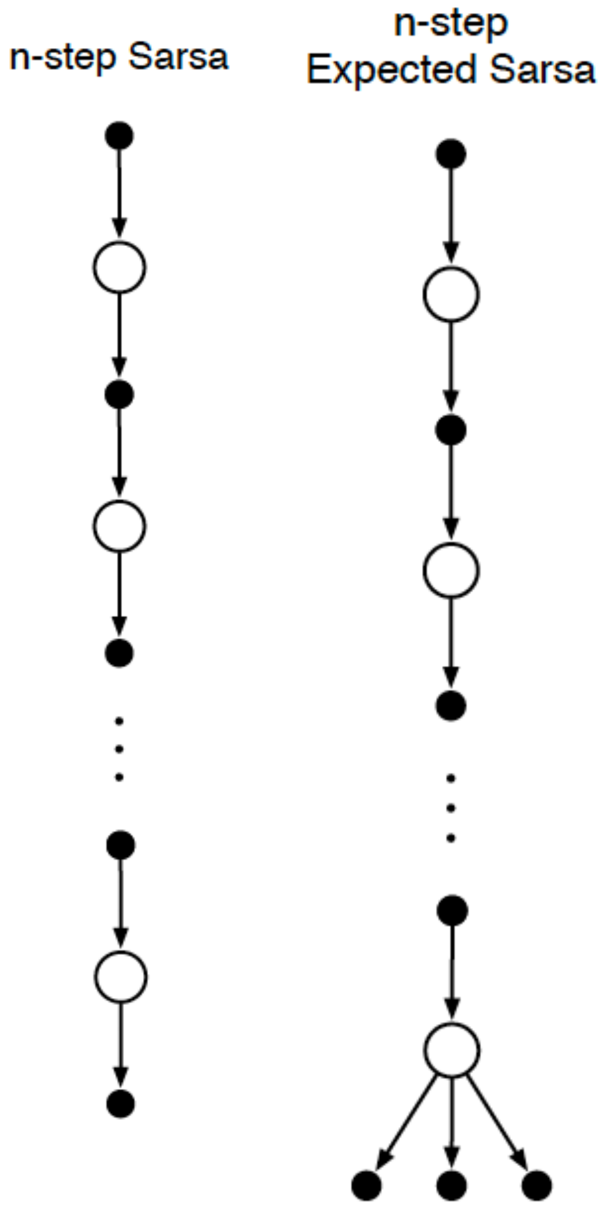
- n-step Sarsa update

$$Q_{t+n}(S_t, A_t) \doteq Q_{t+n-1}(S_t, A_t) + \alpha \left[ G_{t:t+n} - Q_{t+n-1}(S_t, A_t) \right], \quad 0 \leq t < T,$$

- Expected Sarsa

$$G_{t:t+n} \doteq R_{t+1} + \cdots + \gamma^{n-1} R_{t+n} + \gamma^n \bar{V}_{t+n-1}(S_{t+n}), \quad t+n < T,$$

$$\bar{V}_t(s) \doteq \sum_a \pi(a|s) Q_t(s,a), \quad \text{for all } s \in \mathcal{S}.$$

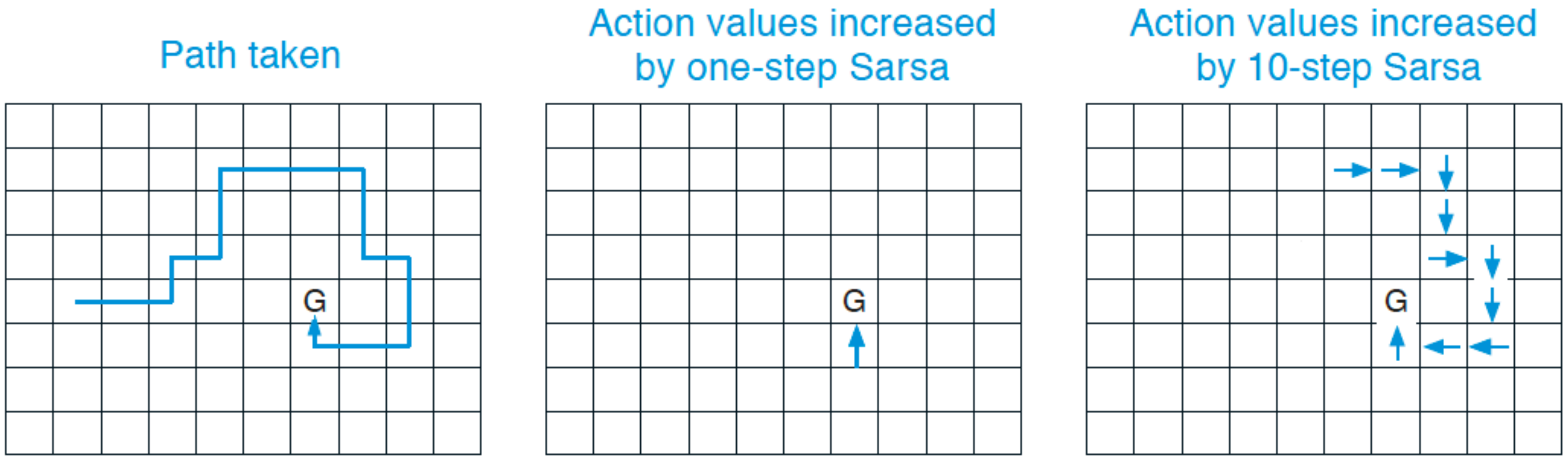


**$n$-step Sarsa for estimating $Q \approx q_*$ or $q_\pi$**

Initialize $Q(s, a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $\pi$ to be $\varepsilon$-greedy with respect to $Q$, or to a fixed given policy
Algorithm parameters: step size $\alpha \in (0, 1]$, small $\varepsilon > 0$, a positive integer $n$
All store and access operations (for $S_t$, $A_t$, and $R_t$) can take their index mod $n + 1$

Loop for each episode:
    Initialize and store $S_0 \neq$ terminal
    Select and store an action $A_0 \sim \pi(\cdot|S_0)$
    $T \leftarrow \infty$
    Loop for $t = 0, 1, 2, \ldots$ :
    |   If $t < T$, then:
    |       Take action $A_t$
    |       Observe and store the next reward as $R_{t+1}$ and the next state as $S_{t+1}$
    |       If $S_{t+1}$ is terminal, then:
    |           $T \leftarrow t + 1$
    |       else:
    |           Select and store an action $A_{t+1} \sim \pi(\cdot|S_{t+1})$
    |   $\tau \leftarrow t - n + 1$    ($\tau$ is the time whose estimate is being updated)
    |   If $\tau \geq 0$:
    |       $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
    |       If $\tau + n < T$, then $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$    $(G_{\tau:\tau+n})$
    |       $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha \left[G - Q(S_\tau, A_\tau)\right]$
    |       If $\pi$ is being learned, then ensure that $\pi(\cdot|S_\tau)$ is $\varepsilon$-greedy wrt $Q$
    Until $\tau = T - 1$

# Gridworld example: one-step vs n-step Sarsa



# n-step Off-policy Learning

- Off-policy n-step TD value function update

$$V_{t+n}(S_t) \doteq V_{t+n-1}(S_t) + \alpha\, \rho_{t:t+n-1} \left[ G_{t:t+n} - V_{t+n-1}(S_t) \right], \qquad 0 \le t < T,$$

  - Importance sampling ratio (relative probability under the learnt and behaviour policies of taking the n actions)

$$\rho_{t:h} \doteq \prod_{k=t}^{\min(h,T-1)} \frac{\pi(A_k|S_k)}{b(A_k|S_k)}.$$

- Off-policy n-step Sarsa

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

  - The IS ratio here starts and ends one step later than for n-step TD. This is because here we are updating a state–action pair having taking the action so IS is applied only for subsequent actions.

- As always with IS, high variance!

**Off-policy $n$-step Sarsa for estimating $Q \approx q_*$ or $q_\pi$**

Input: an arbitrary behavior policy $b$ such that $b(a|s) > 0$, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $Q(s,a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $\pi$ to be greedy with respect to $Q$, or as a fixed given policy
Algorithm parameters: step size $\alpha \in (0,1]$, a positive integer $n$
All store and access operations (for $S_t$, $A_t$, and $R_t$) can take their index mod $n+1$

Loop for each episode:
  Initialize and store $S_0 \neq$ terminal
  Select and store an action $A_0 \sim b(\cdot|S_0)$
  $T \leftarrow \infty$
  Loop for $t = 0, 1, 2, \ldots$ :
  | If $t < T$, then:
  |   Take action $A_t$
  |   Observe and store the next reward as $R_{t+1}$ and the next state as $S_{t+1}$
  |   If $S_{t+1}$ is terminal, then:
  |     $T \leftarrow t+1$
  |   else:
  |     Select and store an action $A_{t+1} \sim b(\cdot|S_{t+1})$
  | $\tau \leftarrow t - n + 1$   ($\tau$ is the time whose estimate is being updated)
  | If $\tau \geq 0$:
  |   $\rho \leftarrow \prod_{i=\tau+1}^{\min(\tau+n-1,T-1)} \frac{\pi(A_i|S_i)}{b(A_i|S_i)}$   $(\rho_{\tau+1:t+n-1})$
  |   $G \leftarrow \sum_{i=\tau+1}^{\min(\tau+n,T)} \gamma^{i-\tau-1} R_i$
  |   If $\tau + n < T$, then: $G \leftarrow G + \gamma^n Q(S_{\tau+n}, A_{\tau+n})$   $(G_{\tau:\tau+n})$
  |   $Q(S_\tau, A_\tau) \leftarrow Q(S_\tau, A_\tau) + \alpha\rho\,[G - Q(S_\tau, A_\tau)]$
  |   If $\pi$ is being learned, then ensure that $\pi(\cdot|S_\tau)$ is greedy wrt $Q$
  Until $\tau = T - 1$

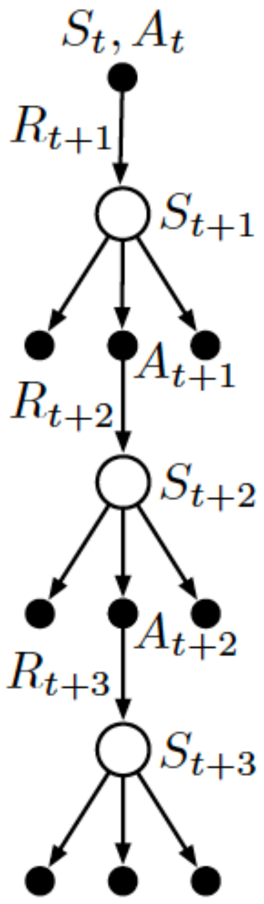
# Off-policy Learning Without Importance Sampling: the n-step Tree Backup Algorithm

- The top node is updated from the estimated action values of the leaf nodes of the tree
  - Interior action nodes are excluded (as based on behaviour policy)
  - Each leaf node contributes to the target with a weight proportional to its probability of occurring under the target policy
- N-step return

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

- Target is usual update from n-step Sarsa

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



the 3-step tree-backup update

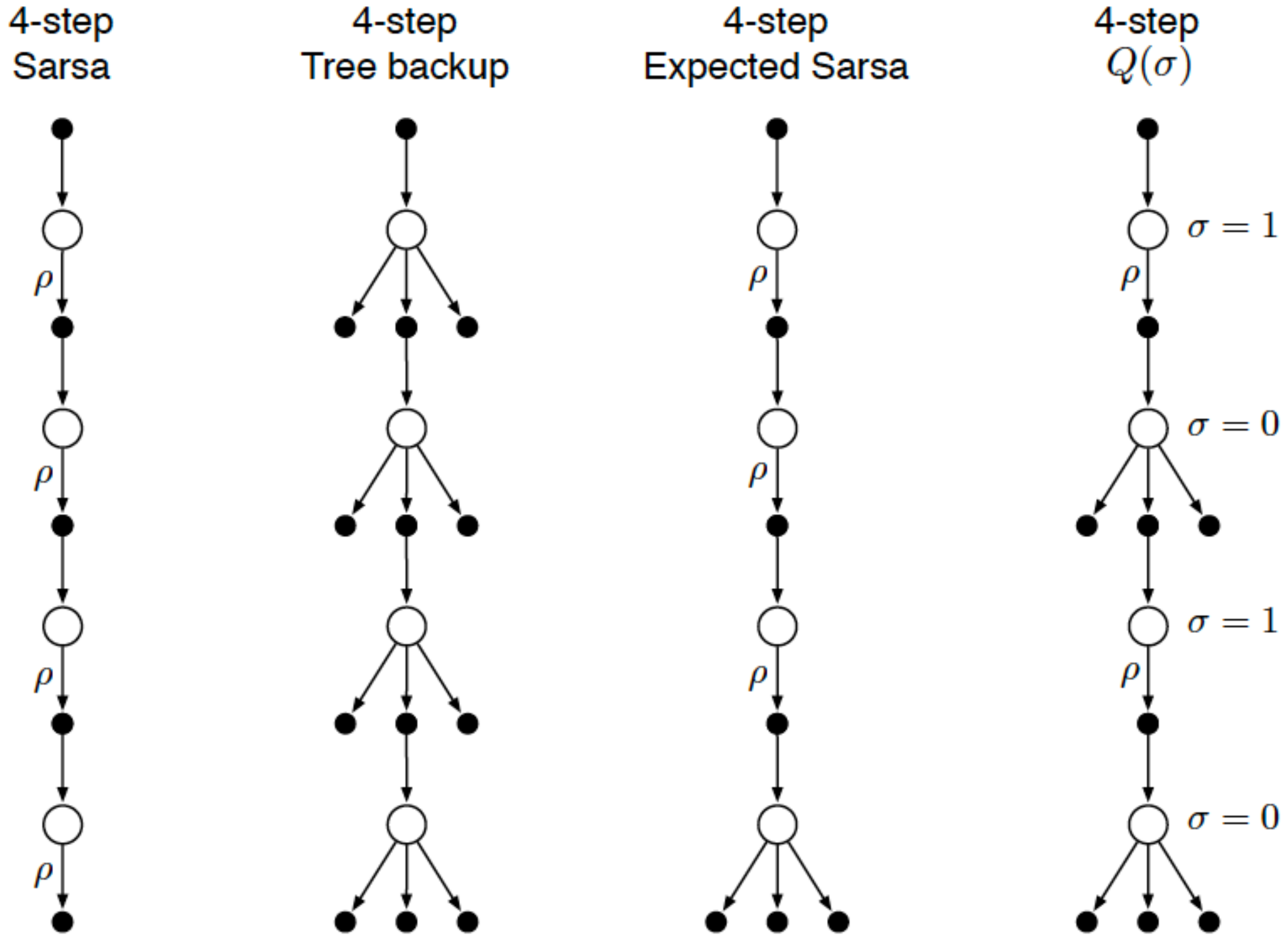
**$n$-step Tree Backup for estimating $Q \approx q_*$ or $q_\pi$**

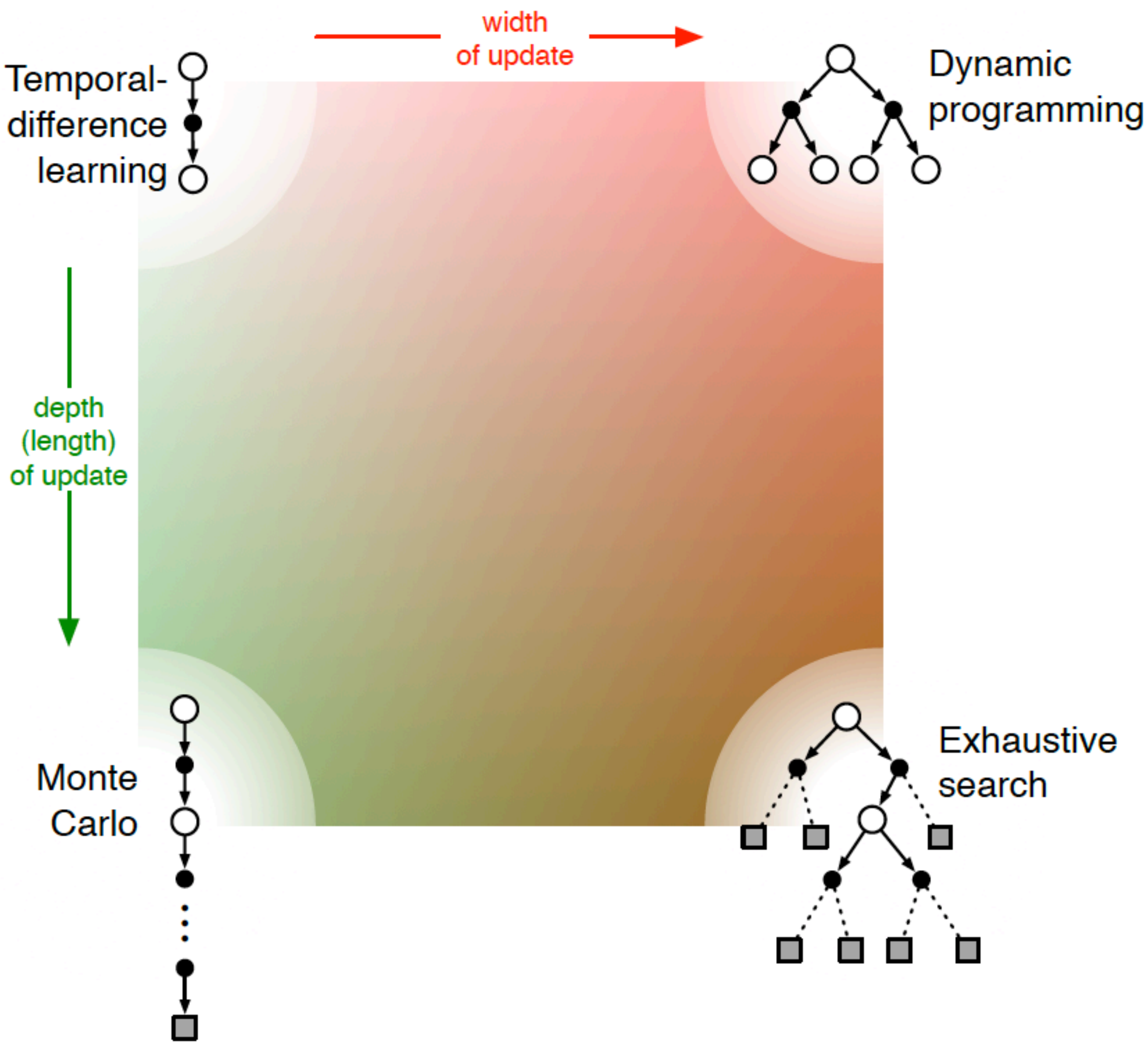
Initialize $Q(s, a)$ arbitrarily, for all $s \in \mathcal{S}, a \in \mathcal{A}$
Initialize $\pi$ to be greedy with respect to $Q$, or as a fixed given policy
Algorithm parameters: step size $\alpha \in (0, 1]$, a positive integer $n$
All store and access operations can take their index mod $n + 1$

```
Loop for each episode:
    Initialize and store S_0 ≠ terminal
    Choose an action A_0 arbitrarily as a function of S_0; Store A_0
    T ← ∞
    Loop for t = 0, 1, 2, ... :
    |   If t < T:
    |       Take action A_t; observe and store the next reward and state as R_{t+1}, S_{t+1}
    |       If S_{t+1} is terminal:
    |           T ← t + 1
    |       else:
    |           Choose an action A_{t+1} arbitrarily as a function of S_{t+1}; Store A_{t+1}
    |   τ ← t + 1 − n    (τ is the time whose estimate is being updated)
    |   If τ ≥ 0:
    |       If t + 1 ≥ T:
    |           G ← R_T
    |       else
    |           G ← R_{t+1} + γ Σ_a π(a|S_{t+1}) Q(S_{t+1}, a)
    |       Loop for k = min(t, T − 1) down through τ + 1:
    |           G ← R_k + γ Σ_{a≠A_k} π(a|S_k) Q(S_k, a) + γ π(A_k|S_k) G
    |       Q(S_τ, A_τ) ← Q(S_τ, A_τ) + α [G − Q(S_τ, A_τ)]
    |       If π is being learned, then ensure that π(·|S_τ) is greedy wrt Q
    Until τ = T − 1
```

# A Unifying Algorithm: n-step Q(σ)

- Degree of sampling on step t: $\sigma_t \in [0,1]$
- See book for equations and algorithms





width
of update
Temporal-
difference
learning
Dynamic
programming
depth
(length)
of update
Monte
Carlo
Exhaustive
search