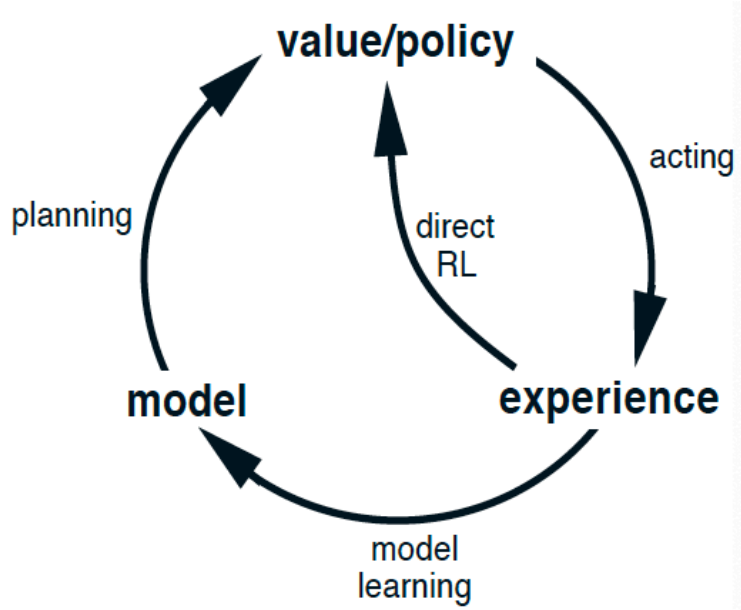
Planning and Learning with Tabular Methods

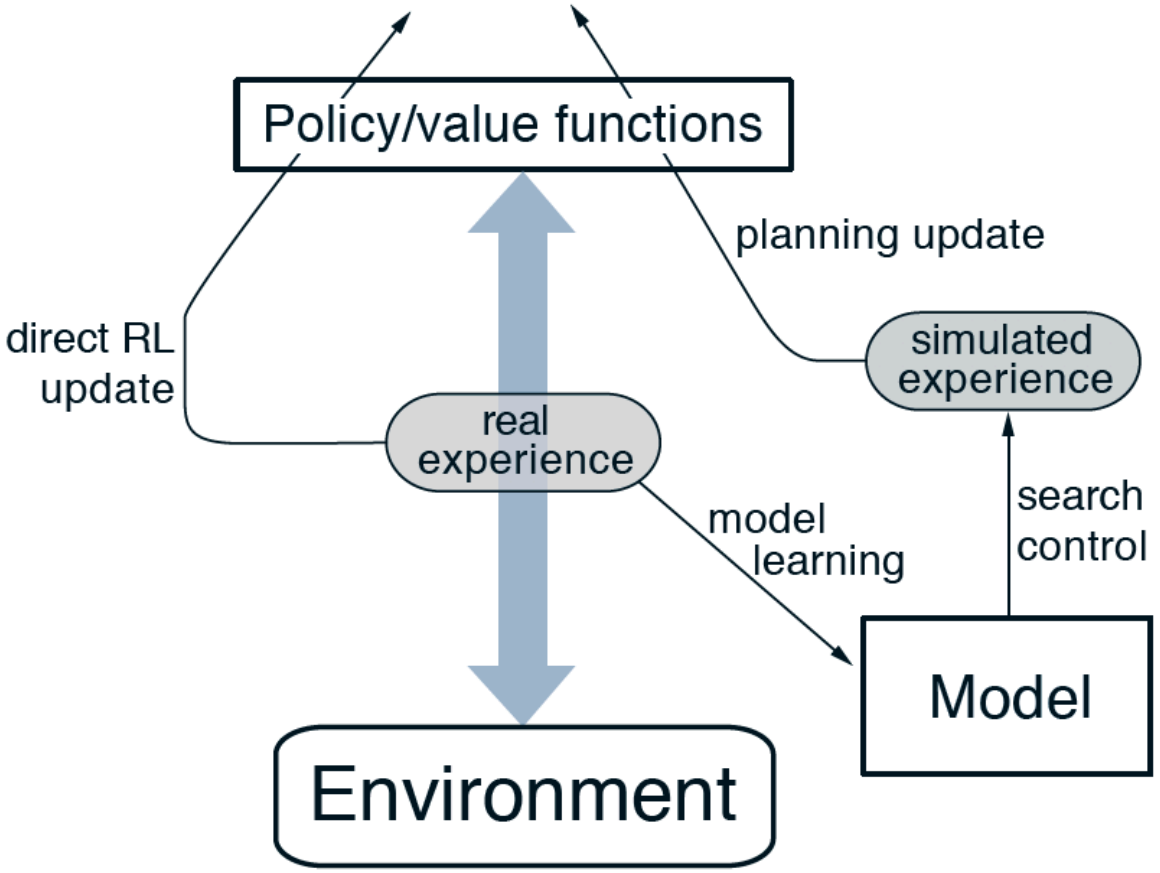
Models and Planning

- Models predict how the environment responds to actions (i.e. next state, reward):
  - Distribution model: used in DP to estimate MDP dynamics $p(s', r|s, a)$
  - Sample models
- Planning process takes model as input and using simulated experience updates the value function estimate (using backup operations) improving the policy

Dyna: Integrated Planning, Acting, and Learning

- Real experience is used to improve the model (model learning or indirect RL) or directly improve the value function (direct RL)





Tabular Dyna-Q

Initialize $Q(s,a)$ and $Model(s,a)$ for all $s \in \mathcal{S}$ and $a \in \mathcal{A}(s)$

Loop forever:

(a) $S \leftarrow$ current (nonterminal) state

(b) $A \leftarrow \varepsilon$-greedy(S,Q)

(c) Take action A; observe resultant reward, R, and state, S'

(d) $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

(e) $Model(S,A) \leftarrow R, S'$ (assuming deterministic environment)

(f) Loop repeat n times:

$S \leftarrow$ random previously observed state

$A \leftarrow$ random action previously taken in S

$R, S' \leftarrow Model(S,A)$

$Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$

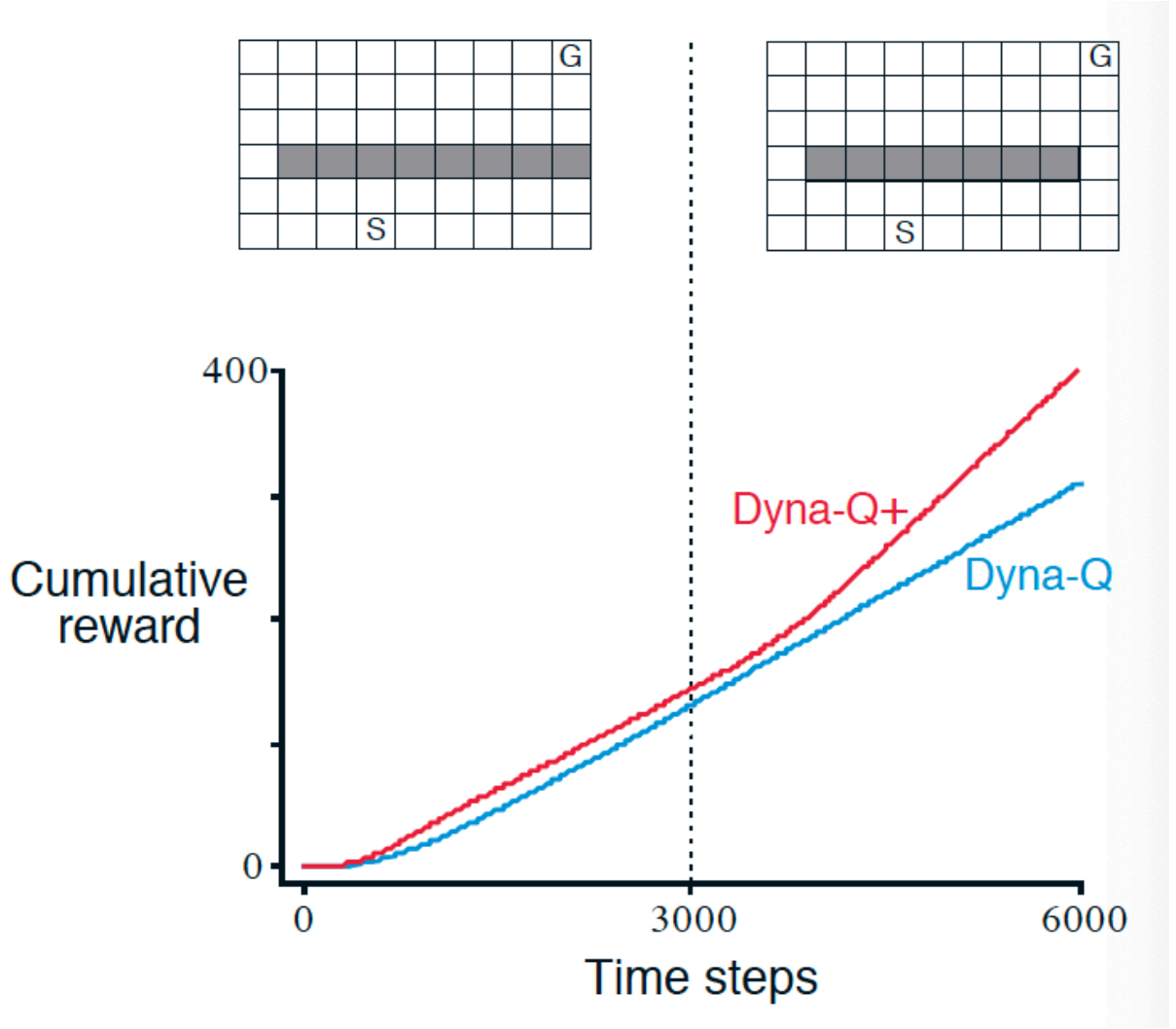
When the Model Is Wrong

- Incorrect model due to sampling from a stochastic environment, function approximation and/or non-stationarity
- With an incorrect model planning will not produce the optimal policy although optimistic estimation can get corrected after interaction in the environment
- Greater difficulties if environment changes to become better (pessimistic estimation)

Dyna-Q+ & shortcut maze example

- Dyna-Q+ encourages exploration by adjusting the reward: $r + \kappa\sqrt{\tau}$

  where τ is the number of steps since state-action pair was selected



Prioritized Sweeping

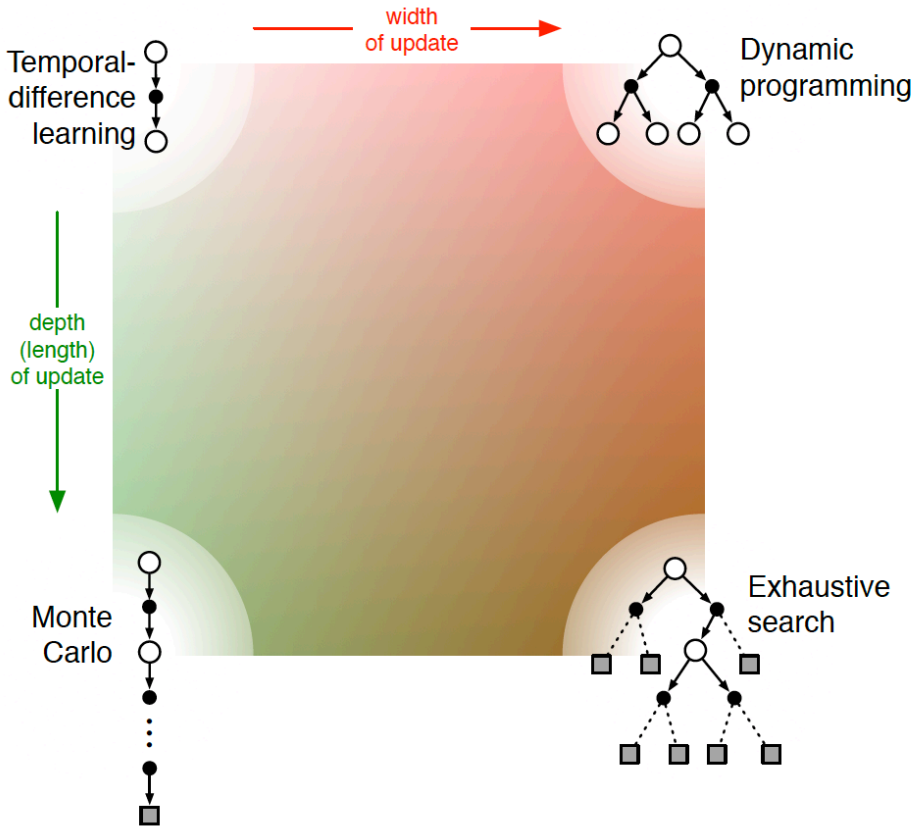
- Improved planning efficiency by selecting state-action pairs to update based on a priority as opposed to random sampling
- For deterministic environments a queue is maintained for every state-action pair sorted by the magnitude of the would be update (TD-error)

Prioritized sweeping for a deterministic environment

Initialize $Q(s,a)$, $Model(s,a)$, for all s,a, and $PQueue$ to empty
Loop forever:
- (a) $S \leftarrow$ current (nonterminal) state
- (b) $A \leftarrow policy(S,Q)$
- (c) Take action A; observe resultant reward, R, and state, S'
- (d) $Model(S,A) \leftarrow R, S'$
- (e) $P \leftarrow |R + \gamma \max_a Q(S',a) - Q(S,A)|$.
- (f) if $P > \theta$, then insert S,A into $PQueue$ with priority P
- (g) Loop repeat n times, while $PQueue$ is not empty:
  - $S,A \leftarrow first(PQueue)$
  - $R,S' \leftarrow Model(S,A)$
  - $Q(S,A) \leftarrow Q(S,A) + \alpha\big[R + \gamma \max_a Q(S',a) - Q(S,A)\big]$
  - Loop for all $\bar{S},\bar{A}$ predicted to lead to S:
    - $\bar{R} \leftarrow$ predicted reward for $\bar{S},\bar{A},S$
    - $P \leftarrow |\bar{R} + \gamma \max_a Q(S,a) - Q(\bar{S},\bar{A})|$.
    - if $P > \theta$ then insert $\bar{S},\bar{A}$ into $PQueue$ with priority P

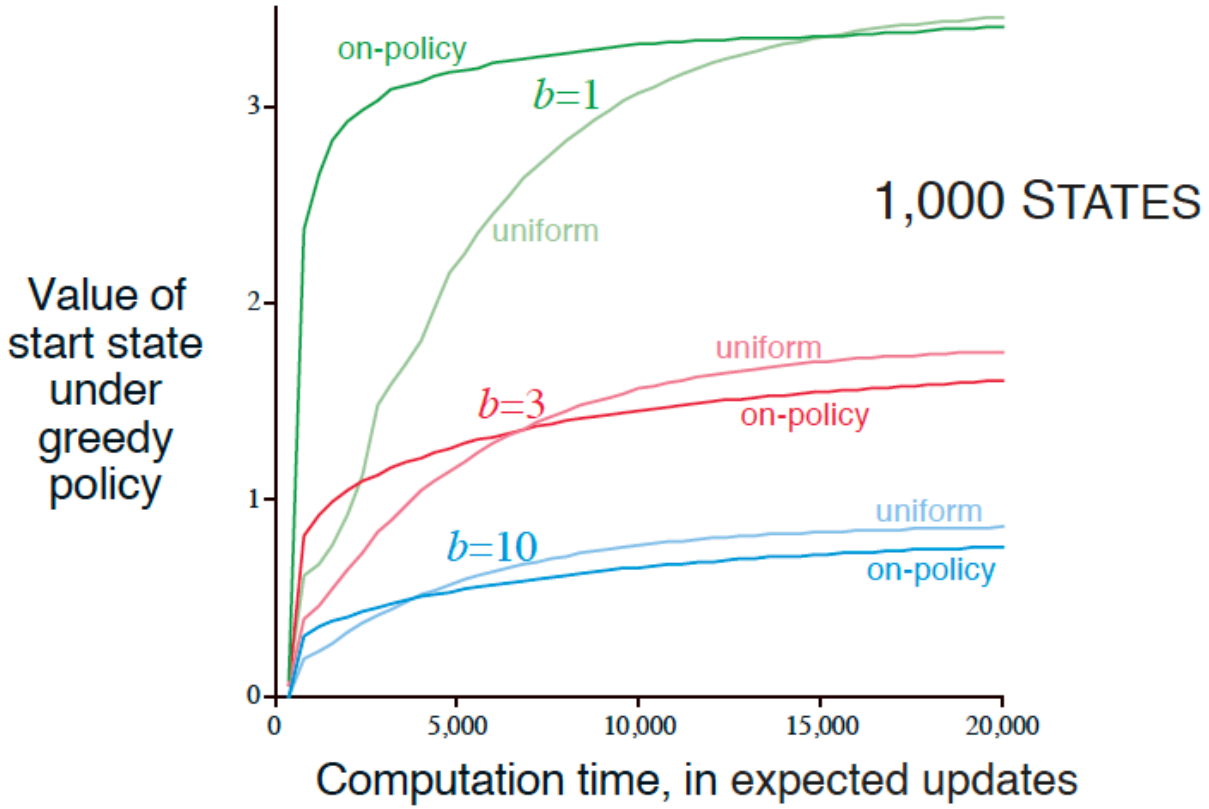
Expected vs. Sample Updates

- Expected updates (e.g. DP backups like policy and value iteration) that require a distribution model:
  - Pros: exact computation
  - Cons: computationally expensive
- Sample updates (e.g TD backups like TD0, Sarsa and Q-learning) that only require a sample model:
  - Pros: computationally cheaper so can perform more updates in same time and estimates are more accurate sooner when bootstrapping
  - Cons: sampling error



Trajectory Sampling

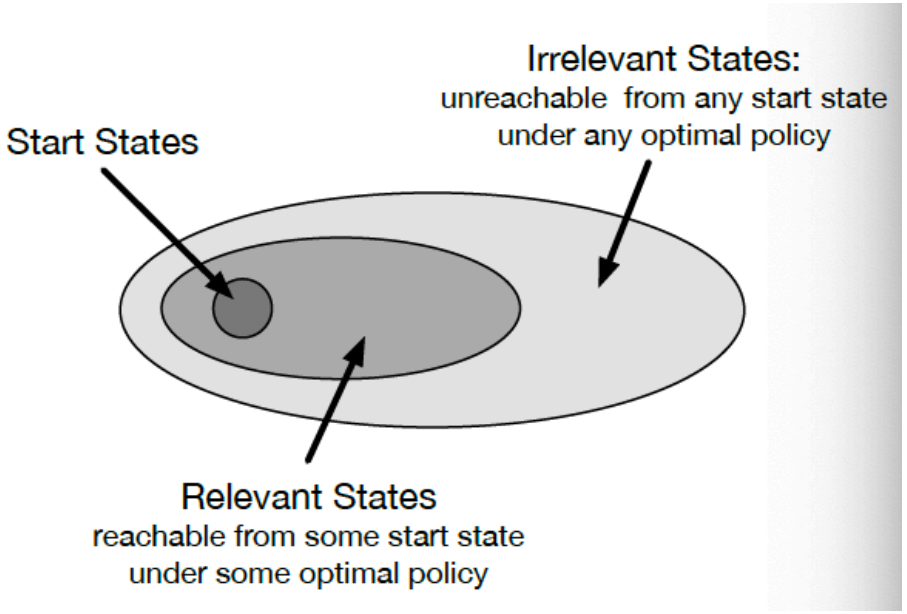
- Distribute updates according to the on-policy distribution (i.e. interact with the model following the current policy) speeds up planning
- However, could result in the same parts of the model being updated over and over



b = branching factor (number of possible next steps)

Real-time Dynamic Programming

- RTDP, is an on-policy trajectory-sampling version of the value-iteration algorithm of DP (asynchronous)
- Converges to the optimal partial policy (optimal for relevant states) without visiting every state infinitely often for stochastic optimal path problems (minimise the cost of a path from a start state to goal state)

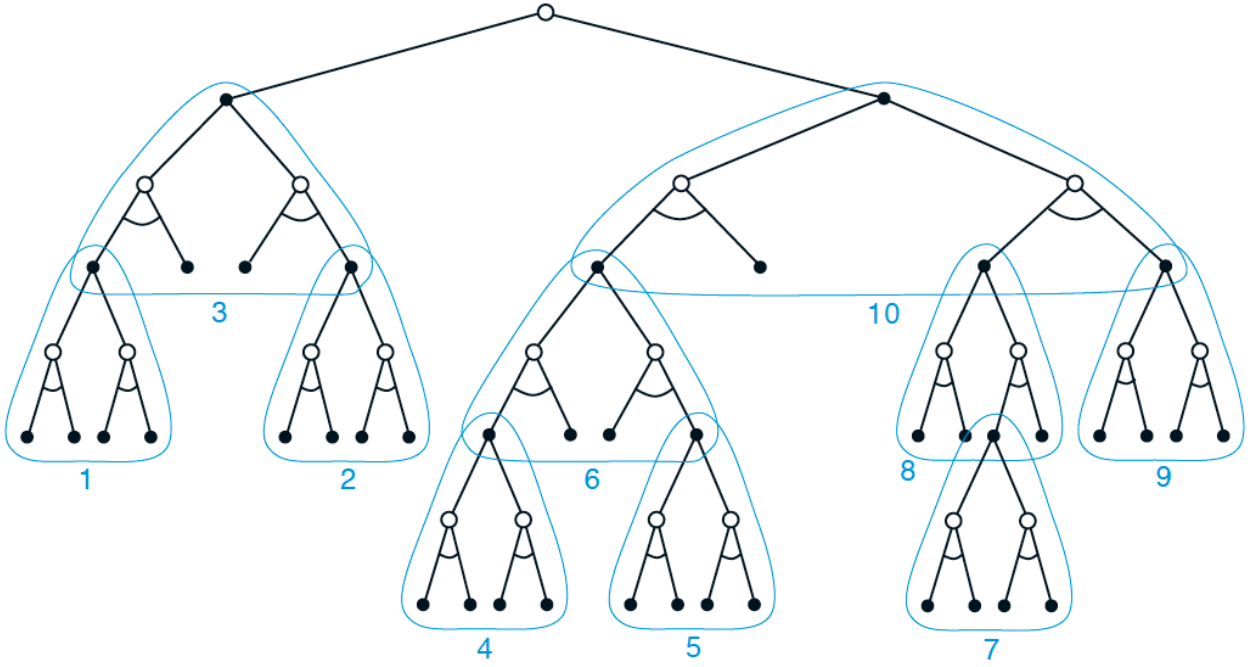


Planning at Decision Time

- DP and Dyna = background planning
- Decision time planning used to select actions from a current state
  - Simplest way is to choose the highest value of the model predicted next states
  - More complex is to consider multi-step lookaheads and evaluate different trajectories
- Most useful in applications in which fast responses are not required

Heuristic Search

- For each state encountered, a large tree of possible continuations is considered and values of leaf nodes are backed up to the root state and the best action is selected (multi-step greedy policy)
- Conventionally the back-up values are discarded but can be used to update the value function
- Computation time depends on depth and branching factor
- Can be implemented as a sequence of one-step updates



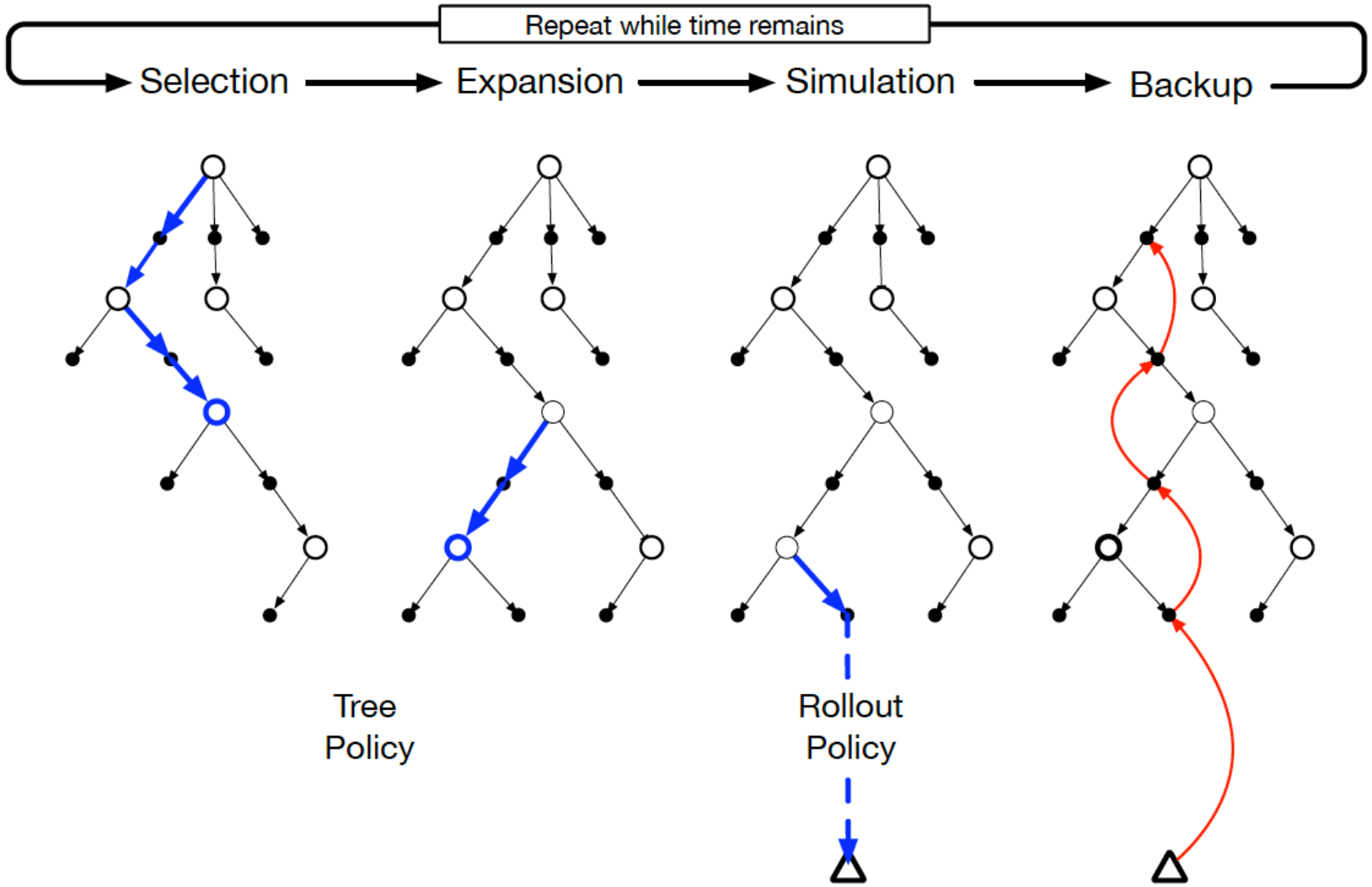
Rollout Algorithms

- Decision-time planning algorithms based on Monte Carlo control applied to simulated trajectories that all begin at the current state and averaging the returns to find the highest value action
- Aim is to improve the rollout policy (not to find the optimal policy)
- No need to store sample outcomes or approximate a value function
- Computation time needed by a rollout algorithm depends on the branching factor, length of trajectory, rollout policy decision making time and the number of trajectories (can parallelize)

Monte Carlo Tree Search

- Rollout algorithm enhanced by the addition of a means for accumulating value estimates obtained from the Monte Carlo simulations in order to successively direct simulations toward more highly-rewarding trajectories
- Rollout policy generates simulated trajectories and then a tree is constructed from the current state for promising state-action pairs. Value estimates are maintained for state-action pairs in the tree and a tree policy is followed (ε-greedy or UCB)

MCTS


Repeat while time remains
Selection
Expansion
Simulation
Backup
Tree
Policy
Rollout
Policy