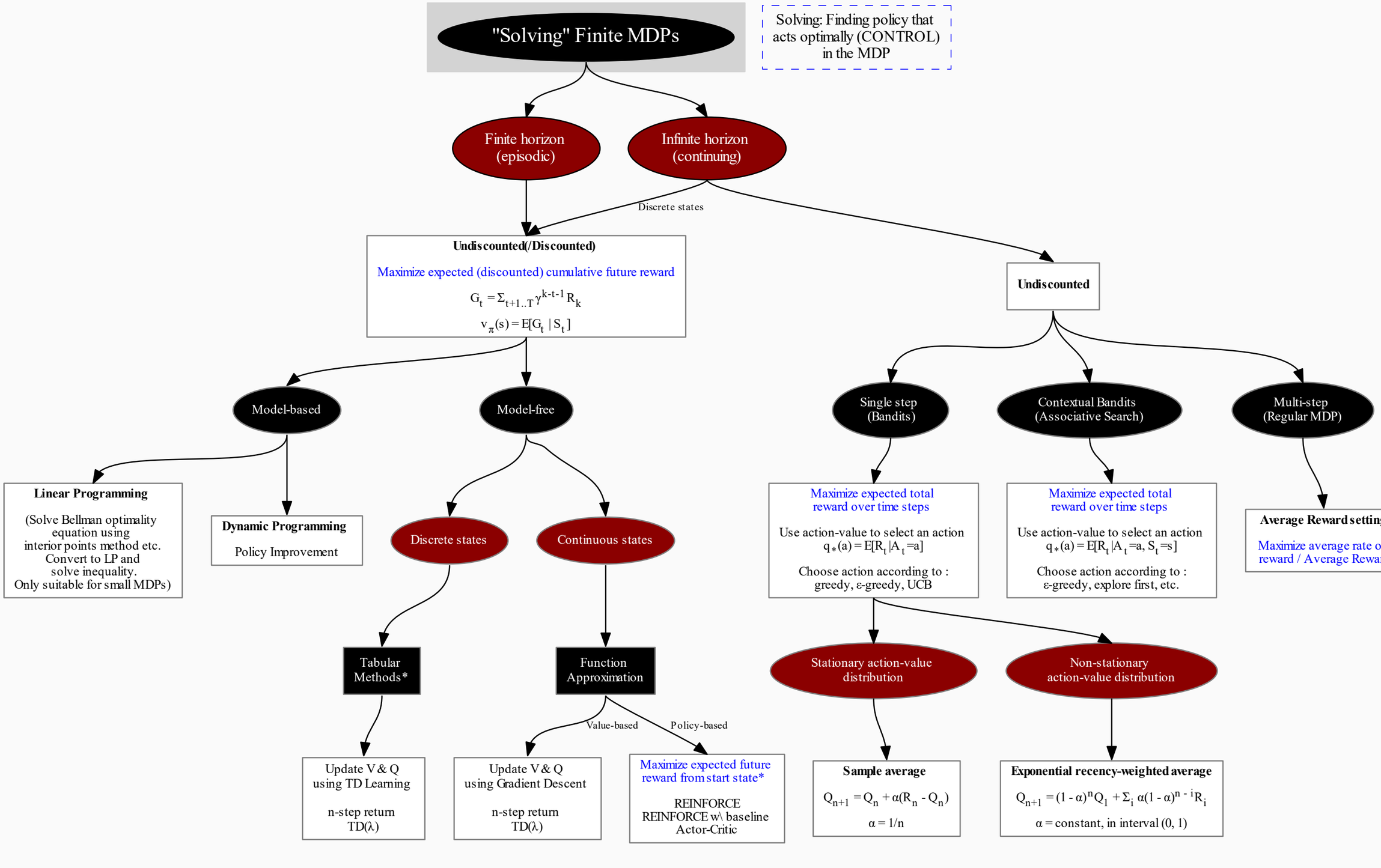
"Solving" Finite MDPs
Solving: Finding policy that acts optimally (CONTROL) in the MDP
Finite horizon (episodic)
Infinite horizon (continuing)
Discrete states
Undiscounted(/Discounted)
Maximize expected (discounted) cumulative future reward
$G_t = \Sigma_{t+1..T} \gamma^{k-t-1} R_k$
$v_\pi(s) = E[G_t \mid S_t]$
Undiscounted
Model-based
Model-free
Single step (Bandits)
Contextual Bandits (Associative Search)
Multi-step (Regular MDP)
Linear Programming
(Solve Bellman optimality equation using interior points method etc. Convert to LP and solve inequality. Only suitable for small MDPs)
Dynamic Programming
Policy Improvement
Discrete states
Continuous states
Maximize expected total reward over time steps
Use action-value to select an action
$q_*(a) = E[R_t \mid A_t = a]$
Choose action according to : greedy, ε-greedy, UCB
Maximize expected total reward over time steps
Use action-value to select an action
$q_*(a) = E[R_t \mid A_t = a, S_t = s]$
Choose action according to : ε-greedy, explore first, etc.
Average Reward setting
Maximize average rate of reward / Average Reward
Tabular Methods*
Function Approximation
Stationary action-value distribution
Non-stationary action-value distribution
Value-based
Policy-based
Update V & Q using TD Learning
n-step return
TD(λ)
Update V & Q using Gradient Descent
n-step return
TD(λ)
Maximize expected future reward from start state*
REINFORCE
REINFORCE w\ baseline
Actor-Critic
Sample average
$Q_{n+1} = Q_n + \alpha(R_n - Q_n)$
$\alpha = 1/n$
Exponential recency-weighted average
$Q_{n+1} = (1-\alpha)^n Q_1 + \Sigma_i \alpha(1-\alpha)^{n-i} R_i$
α = constant, in interval (0, 1)