# Algorithmic and Theoretical Foundations of RL

## Bandit and MCTS

Ke Wei
School of Data Science
Fudan University

With help from Mingxi Hu on MCTS

# Problem Description

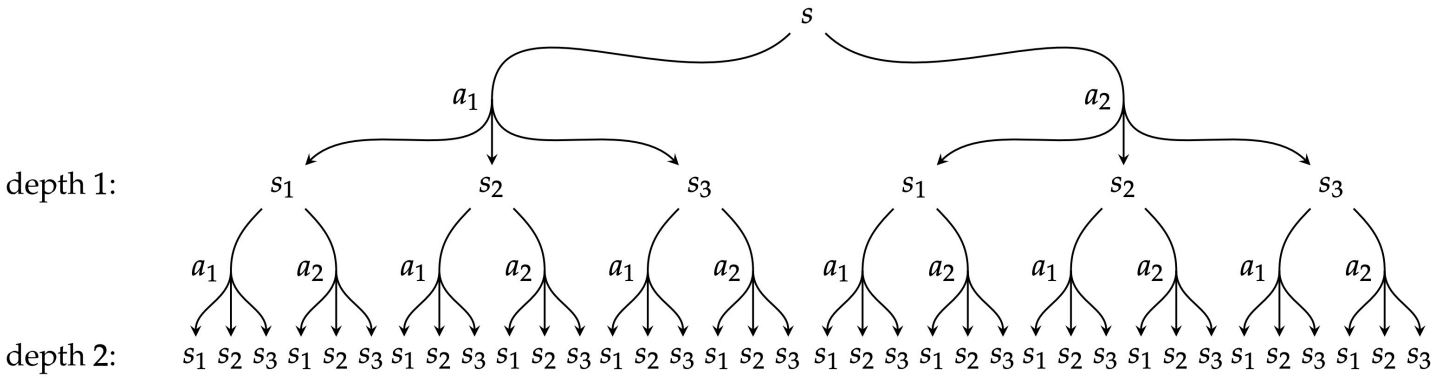
The methods discussed so far focus on solving RL problem **globally in an offline way** via (approximate) dynamic programming or optimization. That is, we would like to find a good action for every state. In contrast, **online planning** methods attempt to find a good action for a single state based on reasoning about states that are reachable from that state. The reachable state space is often orders of magnitude smaller than the full state space, which can significantly reduce storage and computational requirements compared to offline methods.

A typical scheme for online planning is known as **receding horizon planning**, which plans from the current state to a maximum fixed horizon or depth $d$, then executes the action from current state, transitions to the next state, and replans.

---

Materials from “Algorithms for decision making” by Kochenderfer et al., 2022

# Forward Search



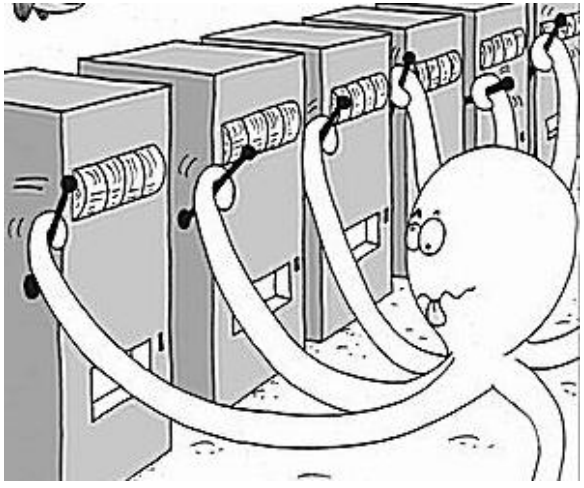

- Forward search builds a search tree with current state as root by expanding all possible transitions up to certain depth via a MDP model, and determines the best action at initial state by for example dynamic programming.
- If it requires planning beyond depth that can be computed online, one can use estimated values obtained using offline RL methods as leaf values.
- In contrast, MCTS is simulation-based search which attempts to reduce computational complexity of forward search by building a tree incrementally based on the balance between exploration and exploitation.

---

Figure from “Algorithms for decision making” by Kochenderfer et al., 2022

# Table of Contents

# MAB

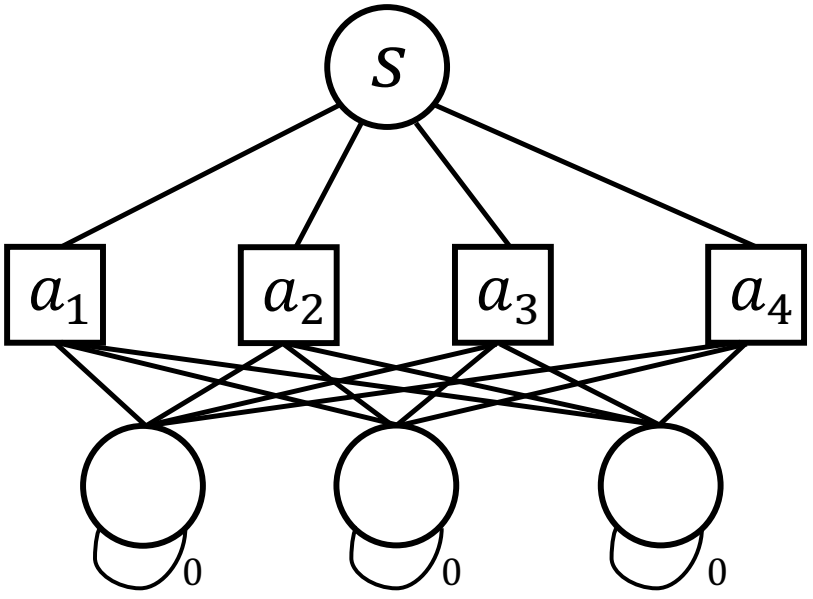


- $K$ actions (arms): $\mathcal{A} = \{1, \cdots, K\}$.
- $r \in [0, 1] \sim \mathcal{D}_a$, unknown probability distribution over rewards for action $a$.
- At time $t$, agent selects an action $a_t$, receives a reward $r_t(a_t) \sim \mathcal{D}_{a_t}$.
- Goal: Maximize cumulative reward $\sum_{t=1}^{T} \mathbb{E}[r_t(a_t)]$.

- There are different settings of bandits, and we only discuss the simplest stochastic setting where rewards of each action at all time steps are i.i.d.
- A fundamental dilemma in online planning is exploration and exploitation tradeoff when facing uncertainty. On one hand, we want to make good decision given current information; on the other hand, uncertainty may mislead and thus requires to explore more decisions before making good decision with high confidence. Overall, a good strategy should count the uncertainty in or we should learn/use the data distribution.

# MAB as RL

MAB is special RL with single state, multiple actions, and random rewards.



# SARSA/Q-Learning for MAB

**Algorithm 1:** SARSA/Q-Learning

**Initialization:** $K$ arms, $Q^0(a) = 0,\ \forall a \in \mathcal{A}$

**for** $t = 0, 1, 2, \ldots$ **do**

Take $a_t \sim \epsilon_t$-greedy$(Q^t(\cdot))$

Observe reward $r_t$

Update $Q^{t+1}(a) = \begin{cases} Q^t(a) + \alpha_t(a) \cdot (r_t - Q^t(a)) & \text{if } a = a_t \\ Q^t(a) & \text{otherwise} \end{cases}$

**end**

- Classic RL algorithms do not focus on the efficient action sampling at each time step. As can be seen later, there exist more efficient algorithms for MAB.

# How to Evaluate an Algorithm? Regret

- $\mu_a$ is mean reward of action $a$: $\mu_a = \mathbb{E}_{r \sim \mathcal{D}_a}[r]$.
- $\mu_* = \mu_{a_*} = \max_a \mu_a$, where $a_* = \operatorname{argmax}_a \mu_a$ is maximum mean reward.

Given a sequential of actions $a_t$ up to time $T$, the total **regret** (or regret for simplicity) is defined as the total loss:

$$R_T = \sum_{t=1}^{T} (\mu_* - \mu_{a_t}).$$

$R_T$ is a random variable, whose randomness comes from the selections of $a_t$.

# Remark about Regret

- Regret characterizes the difference between online performance and offline performance. In the offline setting, we want to choose an action $a$ such that $\sum_{t=1}^{T} \mathbb{E}\left[r_t(a)\right]$ is maximized. It is clear the solution is $a_*$, and the regret is the difference between offline and online cumulative rewards.
- Asymptotically, most algorithms can find action that is close to the best in terms cumulative reward. Then how to compare the performance of different algorithms? Regret provides a micro-level measure based on the loss in the process of applying algorithms, reflecting the speed converging to optimum.

# Probability Tools Needed for Regret Analysis

## Hoeffding Inequality

**Definition 1 (Sub-Gaussian distribution)**
*A random variable X with mean $\mu$ is sub-Gaussian if there exists a $\nu > 0$ such that*

$$\mathbb{E}\left[e^{\lambda(X-\mu)}\right] \leq e^{\frac{\lambda^2\nu^2}{2}}, \quad \forall\lambda \in \mathbb{R}.$$

- Gaussian random variables and bounded random variables are sub-Gaussian.

**Theorem 1 (Hoeffding inequality)**
*Let $\{X_k\}_{k=1}^n$ ($\mathbb{E}[X_k] = \mu$) be i.i.d sub-Gaussian with parameter $\nu$. Then one has,*

$$P\left(\left|\frac{1}{n}\sum_{k=1}^{n}(X_k - \mu)\right| \geq t\right) \leq 2\exp\left(-\frac{nt^2}{2\nu^2}\right).$$

# Overview of Algorithms

- **Explore-First**: Try each arm $N$ rounds first, then pull the empirically best arm.
- **Epsilon-Greedy**: In each round, with an probability $\epsilon_t$, pull all the arms uniformly at random; otherwise pull the best arm so far.
- **UCB:** Be optimism in face of uncertainty. In each round, pull the most promising arm, and this can be done by constructing confidence intervals.

# Explore-First

## Algorithm

There are two phases in Explore-First:

- **Exploration:** Pulls all the arms $N$ rounds;
- **Exploitation:** Pulls the arm with highest empirical mean in remaining rounds.

---

**Algorithm 2:** Explore-First

---

**Initialization:** Parameter $N$.

**for** $a = 1, 2, .., K$ **do**

  Pull arm $a$ for $N$ rounds and collect rewards $\{r_{a,t}\}_{t=1}^{N}$

  Calculate empirical mean reward $\bar{\mu}_a := \frac{1}{N} \sum_{t=1}^{N} r_{a,t}$

**end**

Select the arm $\hat{a} = \underset{a \in [K]}{\operatorname{argmax}} \bar{\mu}_a$ (break ties arbitraily)

Pull arm $\hat{a}$ in all remaining $T - NK$ rounds

---

# Explore-First

## Regret Analysis

The regret of Explore-First consists of two parts,

$$\mathbb{E}[R_T] = \underbrace{\sum_{a \neq a^*} N\left(\mu_* - \mu_a\right)}_{\text{Regret on Exploration Phase}} + \underbrace{(T - NK) \quad (\mathbb{E}[\mu_* - \mu_{\hat{a}}])}_{\text{Regret on Exploitation Phase}}.$$

The choice of $N$ reflects tradeoff between exploration and exploitation. As $N$ increases, regret on exploration increases but regret on exploitation phases decreases with high probability since both $T - NK$ and $P(\hat{a} \neq a^*)$ decreases.

**Theorem 2 (Regret Bound of Explore-First)**

*Explore-First achieves the following bound when* $N = (T/K)^{\frac{2}{3}} \cdot O\left(\log T\right)^{\frac{1}{3}}$,

$$\mathbb{E}\left[R_T\right] \leq T^{\frac{2}{3}} \cdot O\left(K \log T\right)^{\frac{1}{3}}.$$

# Epsilon-Greedy

## Algorithm

**Algorithm 3:** Epsilon-Greedy

**Initialization:** sequence $\{\varepsilon_t\}_{t=1}^{T}$.

**for** $t = 1, 2, .., T$ **do**

Denote $\bar{\mu}_t(a) := \frac{\sum_{i=1}^{t-1} r_i \cdot \mathbf{1}\{a_i = a\}}{\sum_{i=1}^{t-1} \mathbf{1}\{a_i = a\}}$

Toss a coin with success probability $\epsilon_t$;

**if** *success* **then**

Explore: $a_t \sim U([K])$

**else**

Exploit: $a_t = \arg\max_{a \in [K]} \bar{\mu}_t(a)$

**end**

**end**

- Epsilon-Greedy is the same as SARSA/Q-Learning for MAB.

# Epsilon-Greedy

## Regret Analysis

In Epsilon-Greedy, $\epsilon_t$ controls balance between exploration and exploitation. It's natural to let $\epsilon_t$ decrease with $t$ since mean reward of each arm will be estimated more accurately with $t$ increasing.

**Theorem 3 (Regret Bound of Epsilon-Greedy)**

*Epsilon-Greedy achieves the following regret for every $t$ when $\epsilon_t = t^{-\frac{1}{3}} \cdot (K \log t)^{\frac{1}{3}}$,*

$$\mathbb{E}[R_t] \leq t^{\frac{2}{3}} \cdot O(K \log t)^{\frac{1}{3}}.$$

# Optimism in Face of Uncertainty

The key idea of *optimism in face of uncertainty* is to select the most promising action or the action that might have a high reward in an uncertain environment. A random reward might be high if the mean is large or there is more uncertainty in the reward distribution. Thus, a measure should include both information of reward (mean) and uncertainty (more distribution information, e.g., variance).

**Two outcomes** of this scheme:

- Get high reward if the arm really has a high mean reward;
- For arm really having a lower mean reward, pulling it can reduce average reward and mitigate uncertainty.

# Upper Confidence Bound

## Overall Idea

UCB selects arms with highest upper confidence bound: At time step $t$, for each arm $a$, construct the confidence interval (for a fixed confidence) of $\mu_a$ with radius $r_t(a)$ based on the empirical mean $\bar{\mu}_t(a)$. Then UCB selects

$$a_t = \underset{a \in [K]}{\operatorname{argmax}} \ \mathrm{UCB}_t(a) := \bar{\mu}_t(a) + r_t(a) .$$

An arm can have a large $\mathrm{UCB}_t(a)$ for two reasons (or combination thereof):

- $\bar{\mu}_t(a)$ is large: this arm is likely to have a high mean reward;
- $r_t(a)$ is large: this arm has not been explored much.

Either suggests the arm is worth selecting. Thus, $\bar{\mu}_t(a)$ and $r_t(a)$ represent exploitation and exploration. Moreover, UCB counts in effect of finite samples.

# Upper Confidence Bound

## Construction of Upper Confidence Bound

**Lemma 1**

*Let $n_t(a)$ be the number of pulling arm $a$ at time step $t$. For any $0 < \delta < 1$, the following equality holds with probability $1 - \delta$:*

$$|\bar{\mu}_t(a) - \mu(a)| \leq \sqrt{\frac{2}{n_t(a)} \log \frac{2}{\delta}}.$$

- By this lemma, the UCB of arm $a$ at time step $t$ can be constructed as

$$\mathrm{UCB}_t(a) = \bar{\mu}_t(a) + \sqrt{\frac{2}{n_t(a)} \log \frac{2}{\delta}}.$$

# Upper Confidence Bound

## Algorithm

**Algorithm 4:** UCB

**Initialization:** parameter $\delta$

**for** $a = 1, ..., K$ **do**

  Pull arm $a$ and collect reward $r_a$

**end**

**for** $t = 1, 2, .., T - K$ **do**

  $n_t(a) \leftarrow 1 + \sum_{i=1}^{t-1} \mathbf{1}\{a_i = a\}$

  $\bar{\mu}_t(a) = \frac{1}{n_t(a)} \left( r_a + \sum_{i=1}^{t-1} r_t \cdot 1\{a_i = a\} \right)$

  $\mathrm{UCB}_t(a) \leftarrow \bar{\mu}_t(a) + \sqrt{\frac{2}{n_t(a)} \log \frac{2}{\delta}}$

  Select $a_t = \underset{a \in [K]}{\operatorname{argmax}}\ \mathrm{UCB}_t(a)$

**end**

- Typical empirical choice for $\delta$ is $\delta = n_t^{\beta}$, where $n_t$ is total number of simulations, leading to the UCB bound $\mathrm{UCB}_t(a) \leftarrow \bar{\mu}_t(a) + C\sqrt{\frac{\log n_t}{2n_t(a)}}$.

# Upper Confidence Bound

## Regret Analysis

**Theorem 4 (Regret Bound of UCB)**
*UCB achieves the following regret for each round $t \leq T$ when $\delta = \frac{2}{T^4}$,*

$$\mathbb{E}[R_t] \leq O\left(\sqrt{Kt \log T}\right).$$

---

Regret bound for other choice of $\delta$ is also available. Moreover, the lower regret bound for stochastic bandits is $\Omega(\sqrt{KT})$. See "Introduction to Multi-Armed Bandits" by Slivkins 2022 for more details.

# Bayesian Bandits: Probability Matching

Bayesian bandits assume $\{\mu_a\}_{a=1}^K$ obey a prior distribution $Q(\mu_1, \cdots, \mu_K)$. Given history $H_{t-1} = \{(a_1, r_1, \cdots, a_{t-1}, r_{t-1})\}$, the idea of **probability matching** is:

- Compute posterior distribution $P(\mu_1, \cdots, \mu_K | H_{t-1})$ by Bayes law;
- Compute $p_a = P(\operatorname{argmax}_{a \in [K]} \mu_a = a | H_{t-1})$ and select $a$ with largest $p_a$.

Compute $p_a$ from posterior $P$ is difficult. Thompson sampling implements this by sampling: Sample $(\mu_1, \cdots, \mu_K) \sim P(\cdot | H_{t-1})$ and choose the arm with largest $\mu_a$.

# Bayesian Bandits: Thompson Sampling

In the independent setting, $P(\mu_1, \cdots, \mu_K | H_{t-1})$ is decomposable and we can compute the posterior of each arm independently.

**Algorithm 5:** Thompson Sampling

**Initialization:**

**for** $t = 1, 2, .., $ **do**

  Observe the history $H_{t-1} = \{(a_1, r_1), ..., (a_{t-1}, r_{t-1})\}$.

  Compute posterior for each arm $P(\mu_a | H_{t-1})$

  Sample $\bar{\mu}_t(a) \sim P(\mu_a | H_{t-1})$

  Choose the best arm $\hat{a}_t = \underset{a}{\operatorname{argmax}}\, \bar{\mu}_t(a)$ and collect reward $r_t$

**end**

- Thompson sampling achieves nearly optimal Bayesian regret $O(\sqrt{KT \log T})$.

See "Introduction to Multi-Armed Bandits" by Slivkins 2022 for more details.

# Bayesian Bandits: A Special Case

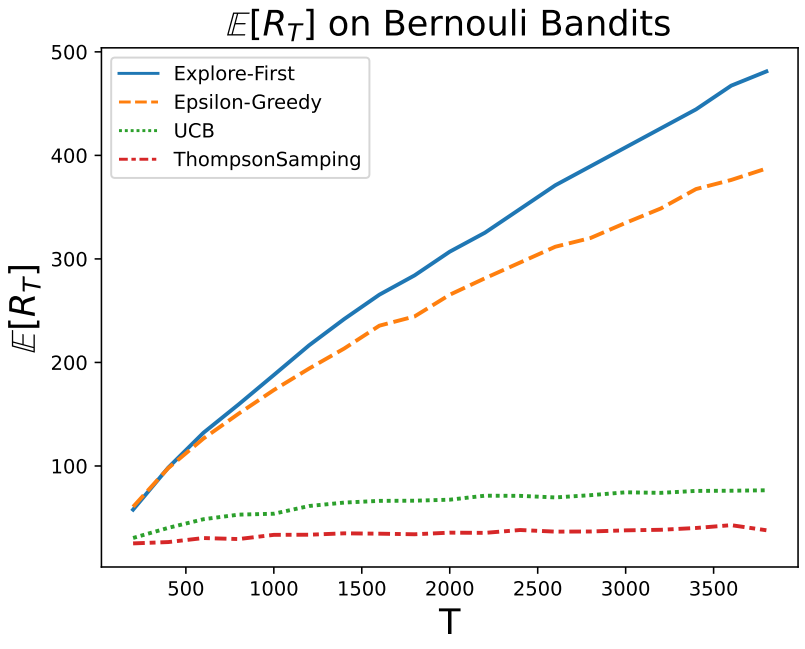
Consider Bernoulli reward case $r|\mu_a \sim \text{Bernoulli}\,(\mu_a)$, where $\mu_a$ indicates the probability $r = 1$ (also mean of $r$). Assume uniform prior distribution for $\mu_a$, i.e., $\mu_a \sim U([0,1])$. Given independent random rewards $r_1, \cdots, r_n$ sampled for arm $a$, by Bayes law, pdf for posterior distribution $P(\mu_a|r_1, \cdots, r_n)$ is

$$\begin{aligned} p(\mu_a|r_1, \cdots, r_n) &\propto p(r_1, \cdots, r_n|\mu_a)p(\mu_a) \\ &= \prod_{k=1}^{n} \mu_a^{r_k}(1-\mu_a)^{1-r_k} = \mu_a^{\sum_{k=1}^n r_k}(1-\mu_a)^{\sum_{k=1}^n (1-r_k)}. \end{aligned}$$

It follows that $P(\mu_a|r_1, \cdots, r_n) = \textbf{Beta}(1+m_1, 1+m_2)$, where $m_1$ is the number of rewards that $r_k = 1$ and $m_2$ is the number of rewards that $r_k = 0$.

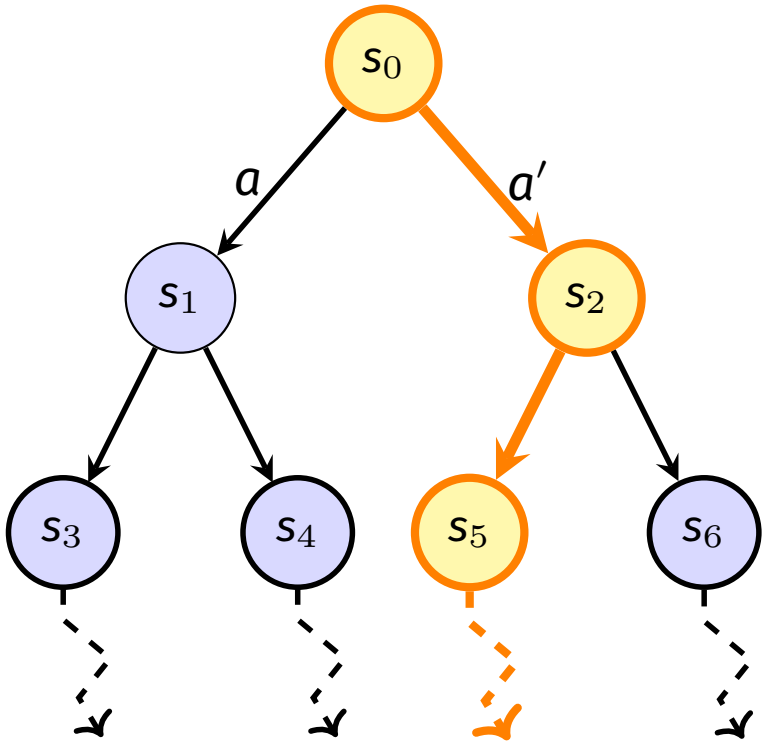
- Note $\mathbb{E}\left[\textbf{Beta}(\alpha, \beta)\right] = \frac{\alpha}{\alpha+\beta}$ and $U[0,1] = \textbf{Beta}(1,1)$. Thus, prior and posterior distributions are in the same distribution family, known as **conjugate prior**.

# Illustrative Example



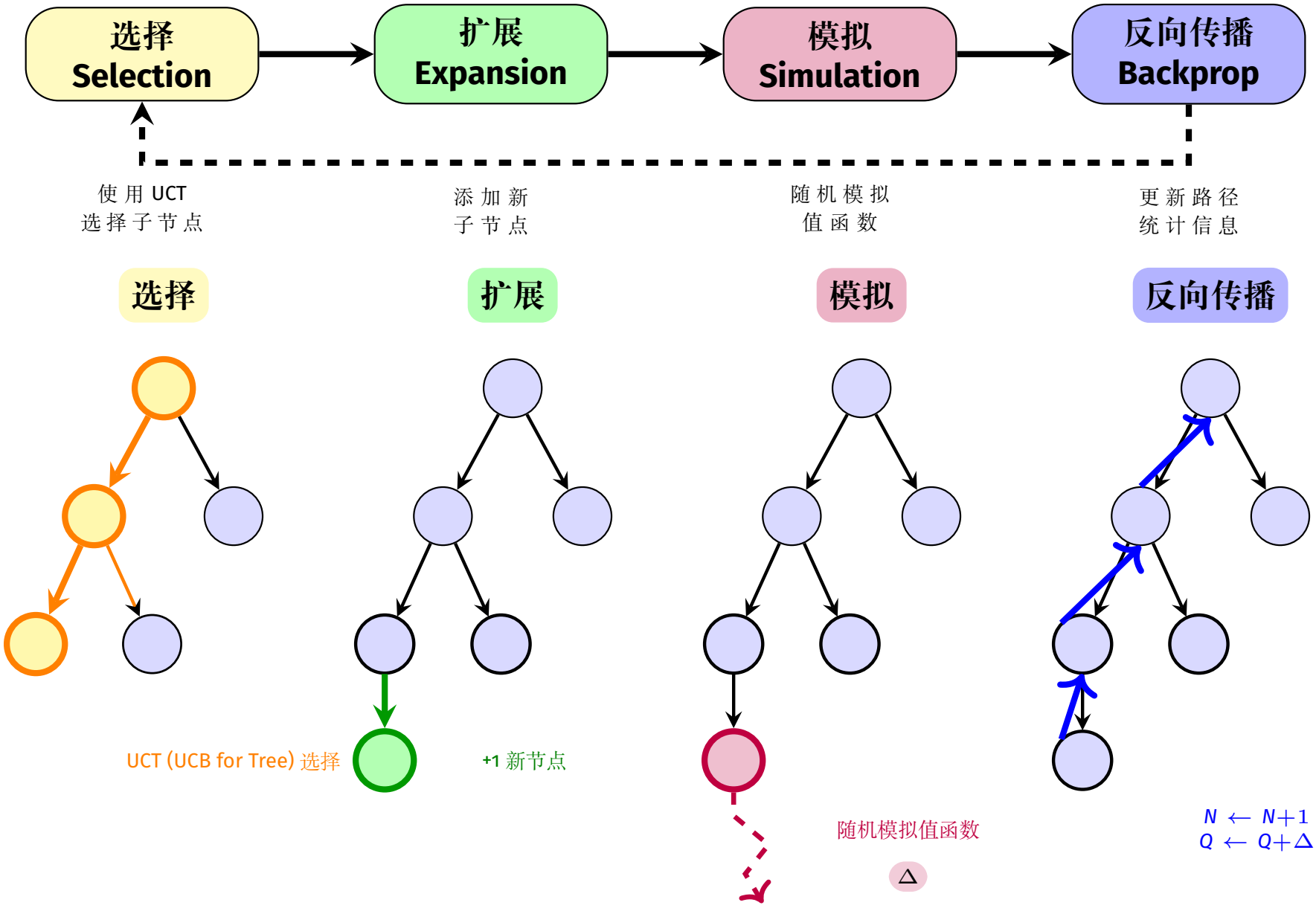
# Table of Contents

# Problem Setup




- The goal is to find next action which leads to path with the maximum reward under finite computation budget (consider deterministic transition here).
- Instead of searching all paths in depth, MCTS builds a tree incrementally and only does in-depth search to those branches with high potential.

# MCTS: Four Phases



# MCTS: Main Search

**Algorithm 6:** MCTS-SEARCH($s_0$)

create root node $v_0$ with state $s_0$

**while** *within computational budget* **do**

  $v_l \leftarrow$ TREEPOLICY($v_0$) ▷ Selection + Expansion

  $\Delta \leftarrow$ DEFAULTPOLICY($s(v_l)$) ▷ Simulation

  BACKPROP($v_l, \Delta$) ▷ Backpropagation

**end**

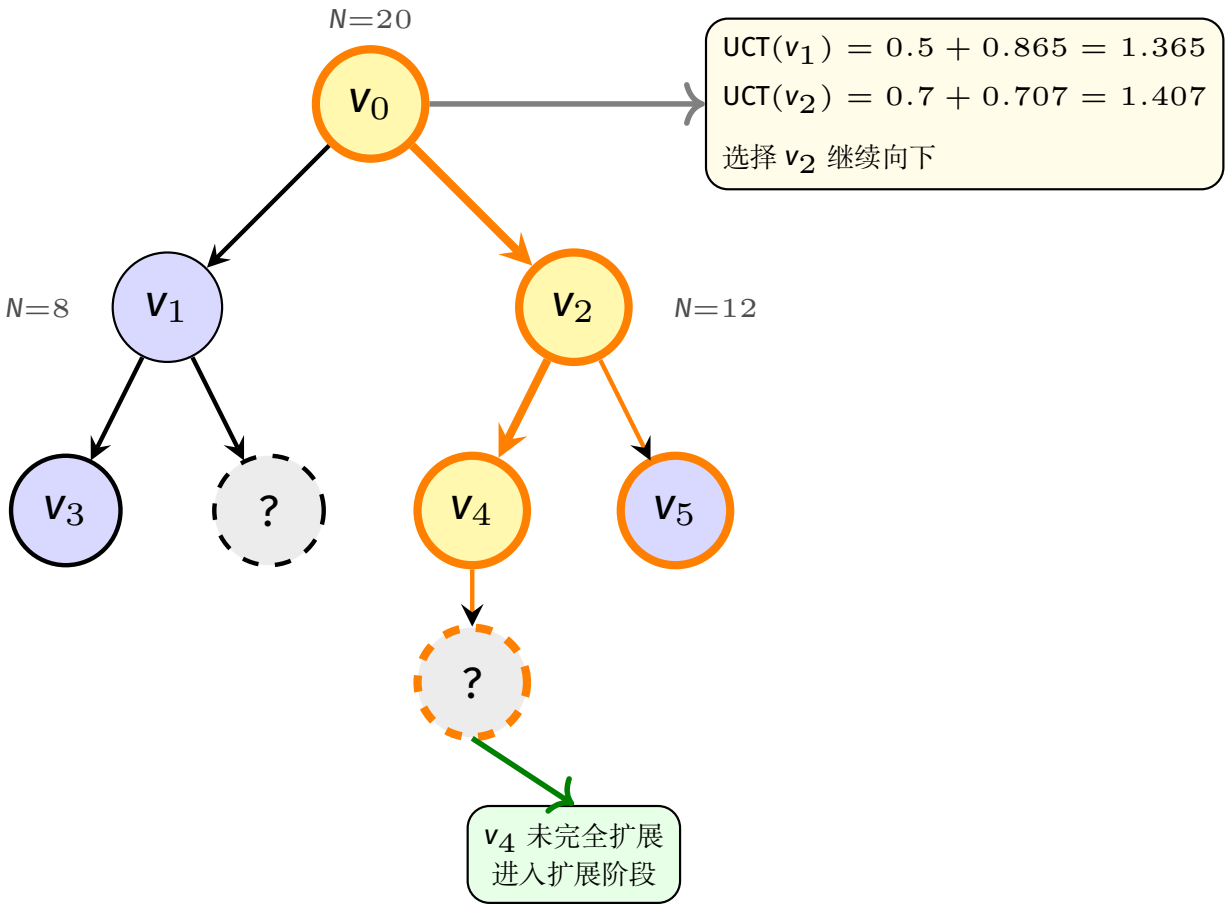
**return** $a(\text{BESTCHILD}(v_0, 0))$ ▷ Return best action

- $v_l$ is representation of $s_l$, $s(v)$ is state associated with $v$.
- $a(v)$ is the action that takes to reach $s(v)$ from father node.

**Algorithm 7:** BESTCHILD($v$,$c$)

**return** $\underset{v_j \in \text{children}(v)}{\operatorname{argmax}} \left[ \frac{Q(v_j)}{N(v_j)} + c\sqrt{\frac{2\ln N(v)}{N(v_j)}} \right]$

# MCTS: Selection

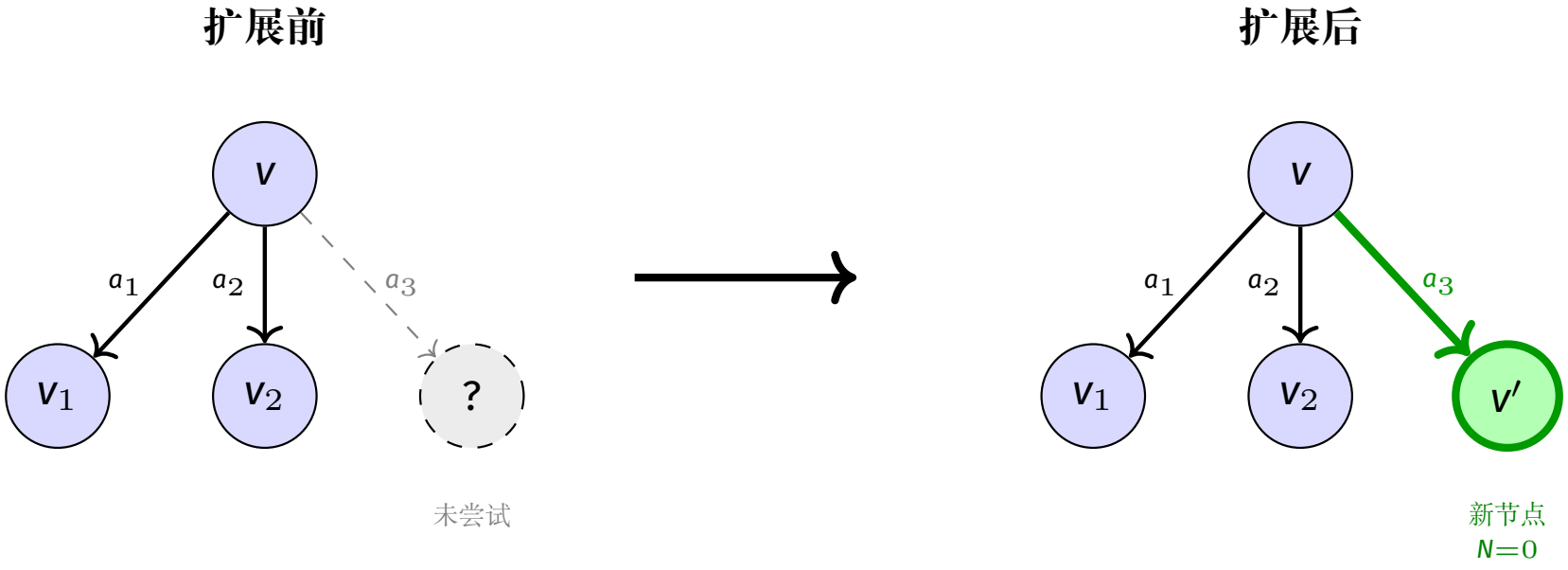


# MCTS: Selection

**Algorithm 8:** TreePolicy($v$)

**while** *$v$ is nonterminal* **do**
    **if** *$v$ is not fully expanded* **then**
        **return** Expand($v$)
    **else**
        $v \leftarrow$ BestChild($v, C_p$)
    **end**
**end**
**return** $v$

# MCTS: Expand




**Algorithm 9:** EXPAND($v$)

choose $a \in$ untried actions from $s(v)$

create child node $v'$

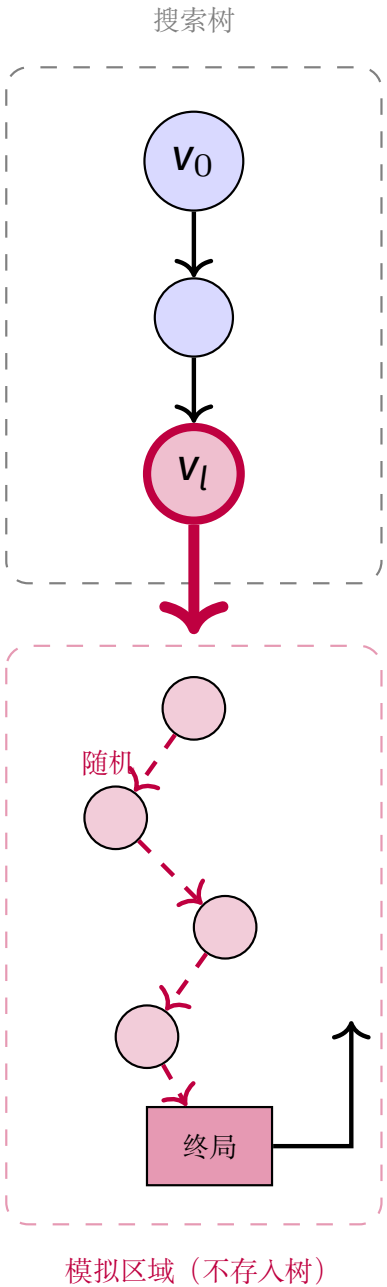
$s(v') \leftarrow (s(v), a)$ ▷ Transition to new state and associate it with $v'$

$a(v') \leftarrow a$ ▷ Record action

add $v'$ to children of $v$

**return** $v'$

# MCTS: Simulation



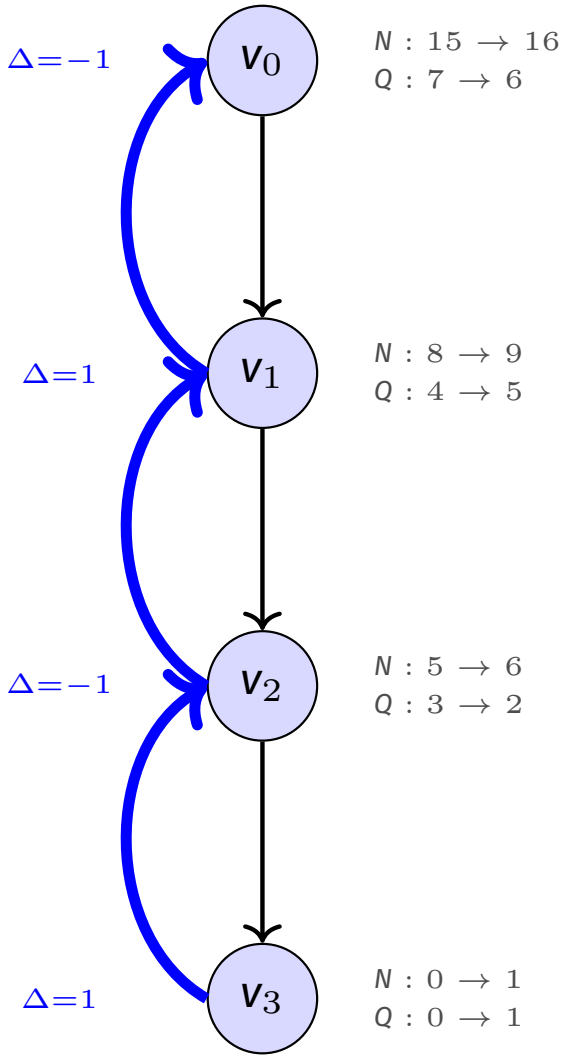

模拟结果
$\Delta = 1$
(当前玩家胜)

**Algorithm 10:** DEFAULTPOLICY(s) [Random Rollout]

**while** *s is nonterminal* **do**
  $a \leftarrow$ uniform random action at $s$
  $s \leftarrow (s, a)$ ▷ State transition
**end**
**return** reward for terminal state $s$

- This phase relies on the particular task. Here we consider zero-sum game where father and child nodes represent two different players.

# MCTS: Back Propagation




$N(v) \leftarrow N(v) + 1$

$Q(v) \leftarrow Q(v) + \Delta$

$\Delta \leftarrow -\Delta$ (双人博弈)

**Algorithm 11:** BACKPROP($v, \Delta$)

**while** $v \neq null$ **do**

  $N(v) \leftarrow N(v) + 1$

  $Q(v) \leftarrow Q(v) + \Delta$

  $\Delta \leftarrow -\Delta$ ▷ 交替取反

  $v \leftarrow \text{parent}(v)$

**end**

- This phase also relies on particular task, and here considers zero-sum game where father and child nodes represent two different players.

# References


- Coulom, R. (2006). Efficient Selectivity and Backup Operators in Monte-Carlo Tree Search. *Computers and Games*, LNCS 4630, 72-83.
- Kocsis, L., & Szepesvŕi, C. (2006). Bandit Based Monte-Carlo Planning. *ECML 2006*, LNCS 4212, 282-293.
- Browne, C., et al. (2012). A Survey of Monte Carlo Tree Search Methods. *IEEE Trans. CIAIG*, 4(1), 1-43.

# Questions?