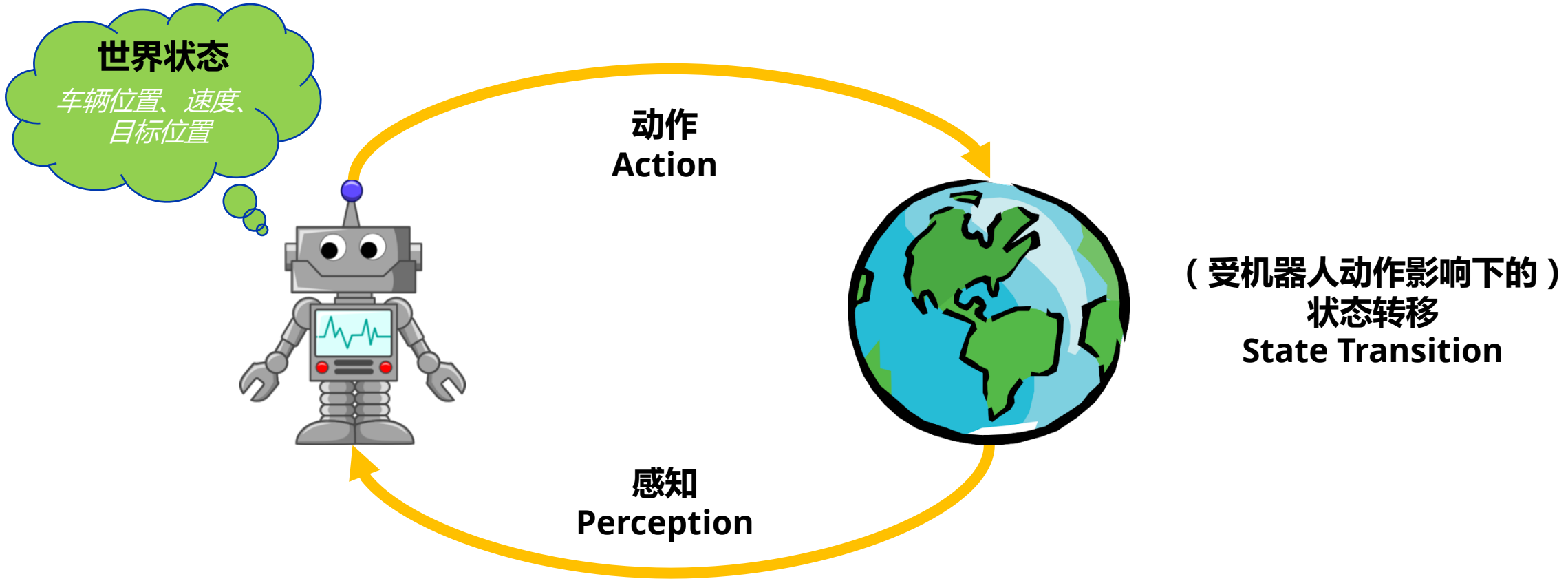
CS7355
自动驾驶前沿技术

2024年5月8日

机器人与随机环境的交互

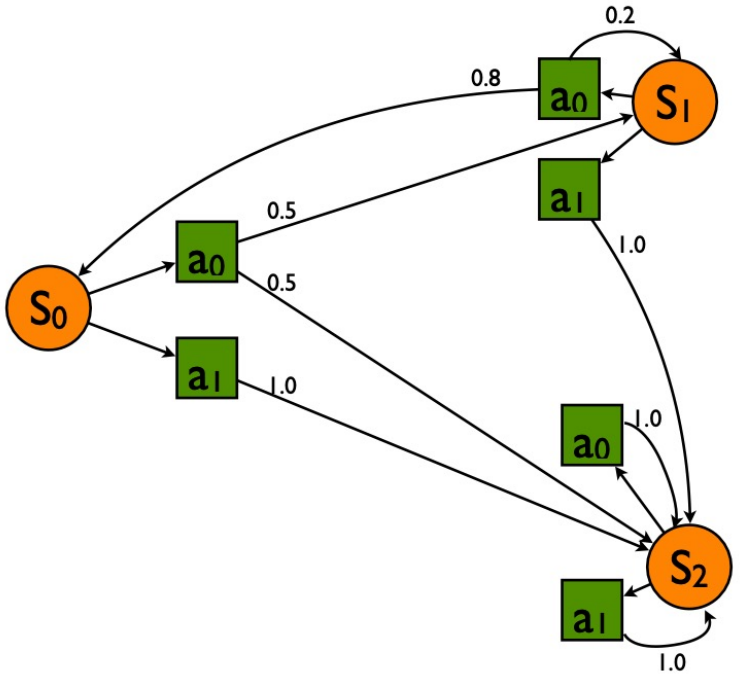
主动交互：

- 车辆观察其他车辆和行人的行为（感知），并找机会穿过十字路口（动作）



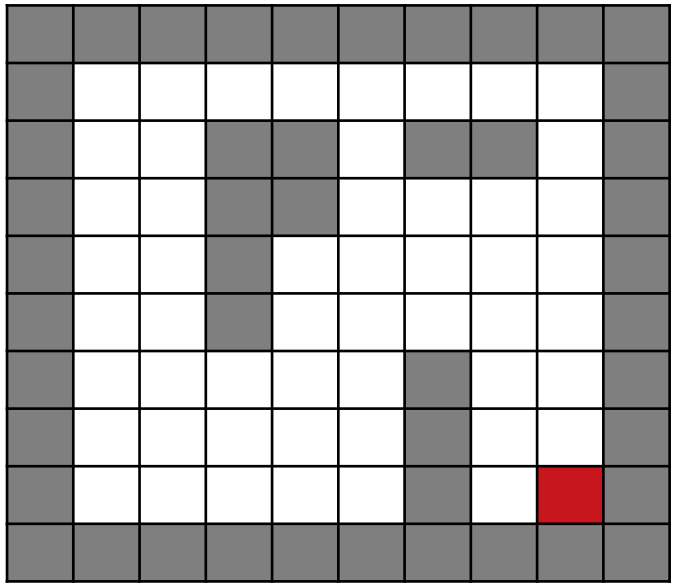
马尔科夫决策过程（Markov Decision Process）

- **MDP 包含了一下四个元素：**
  - S: 状态空间
    - 状态 s 可以表示场景中所有人、车所在的几何、行为状态
  - A: 动作空间
    - 动作 a 可以表示自车的行为
    - 底层行为：方向盘角度、加速度
    - 高层行为：跟车、变道、避让，...
  - $T(s, a, s') = p(s'|s, a)$：状态转移函数
    - 表达机器人行为带来的不确定性结果：若当前世界处在状态 s, 机器人执行动作 a, 世界下一步转移到状态 s' 的概率是多少？
  - $R(s, a)$ ：奖励函数
    - 表达机器人任务：若机器人在世界状态 s 执行动作 a, 所获得的即时奖励是多少？
    - 例：碰撞（-30000），到达终点（+100），变道（-4），...



思考题：MDP问题建模

- 机器人的工作空间是一个 10x10 栅格：
  - 灰色代表障碍物
  - 终点在右下角（红点）
- 没有障碍物的情况下，机器人在每一步可以选择移动到相邻4个格子，成功率为3/4；失败时，机器人会以同等的概率滑到其他3个格子；机器人也可以选择停在原地不动
- 到达终点取得+1奖励，其余时间没有奖励
- $\gamma = 0.9$



S？
A？
T？
R？

Bellman最优性条件

- **最优值**（optimal value）：从状态 s 出发，所能达到的最优价值（期望累积奖励）

$$V^*(s) = \max_a R(s,a) + \gamma \sum_{s'} T(s,a,s') V^*(s') \qquad \text{(Bellman-V)}$$

- **最优 Q 值**（optimal q-value）：从状态 s 出发，执行动作 a 之后，所能达到的最优价值

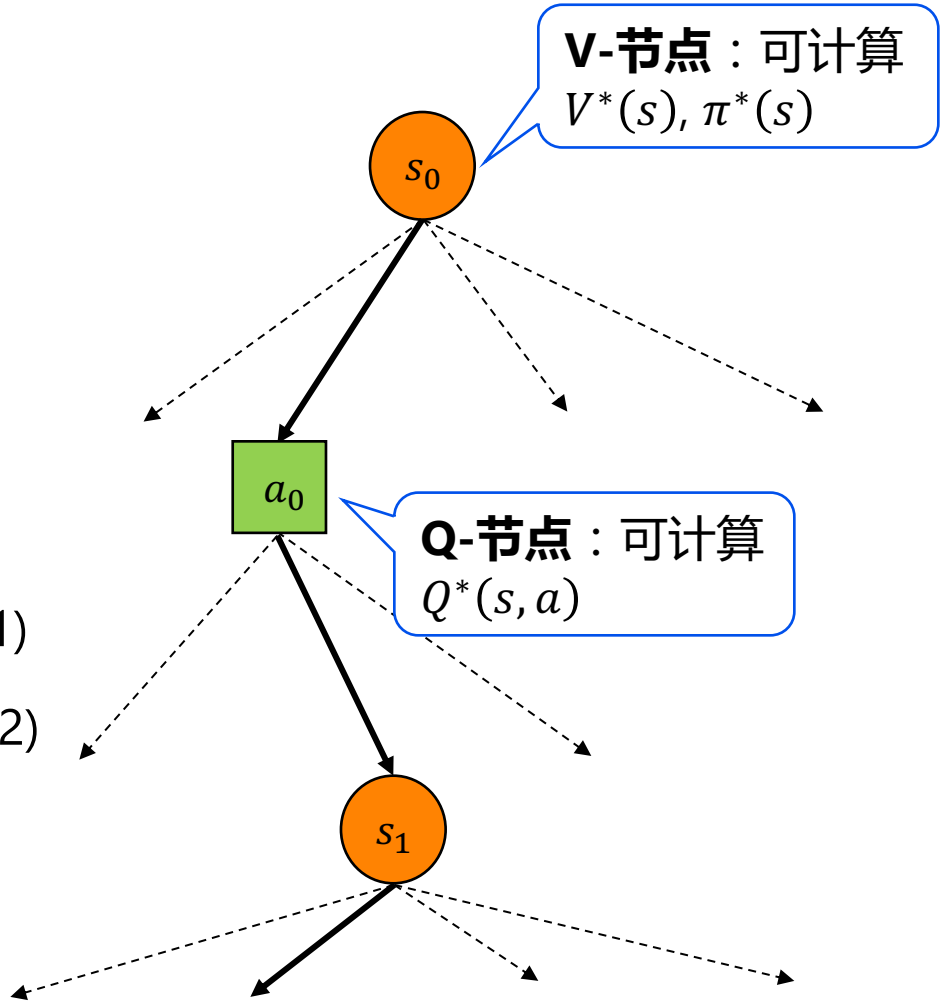
$$Q^*(s,a) = R(s,a) + \gamma \sum_{s'} T(s,a,s')\, V^*(s') \qquad \text{(Bellman-VQ1)}$$

$$V^*(s) = \max_a Q^*(s,a) \qquad \text{(Bellman-VQ2)}$$

- **最优策略**（optimal policy）：使价值最大化的策略

$$\pi^*(s) = \operatorname{argmax}_a Q^*(s,a), \text{或} \qquad \text{(Bellman-Pi1)}$$

$$\pi^*(s) = \operatorname{argmax}_a R(s,a) + \gamma \sum_{s'} T(s,a,s')\, V^*(s') \qquad \text{(Bellman-Pi2)}$$



思考题

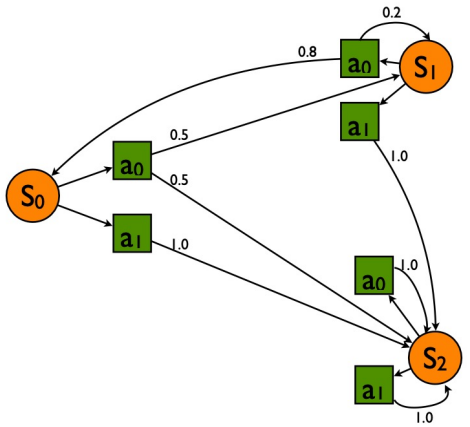
- **奖励函数 $R(s,a)$**：是否能表达以下类型的奖励?
  1. 碰撞时 R =-3000
     - 奖励只跟当前状态相关: $R(s)$
  2. 变道时 R =-4
     - 奖励只跟当前动作相关: $R(a)$
  3. 变道对他车速度的影响
     - 奖励跟当前状态、动作、下一步状态都相关：$R(s,a,s')$
- $R(s,a,s')$ 与 $R(s,a)$ 等价！
  - $V^*(s) = \max_a R(s,a) + \gamma \sum_{s'} T(s,a,s')V^*(s')$ (1)
  - $V^*(s) = \max_a \sum_{s'} T(s,a,s')[R(s,a,s') + \gamma V^*(s')]$ (2)
  - 只需定义 $R(s,a) = \sum_{s'} T(s,a,s')R(s,a,s')$ ，则 (1) 与 (2) 等价

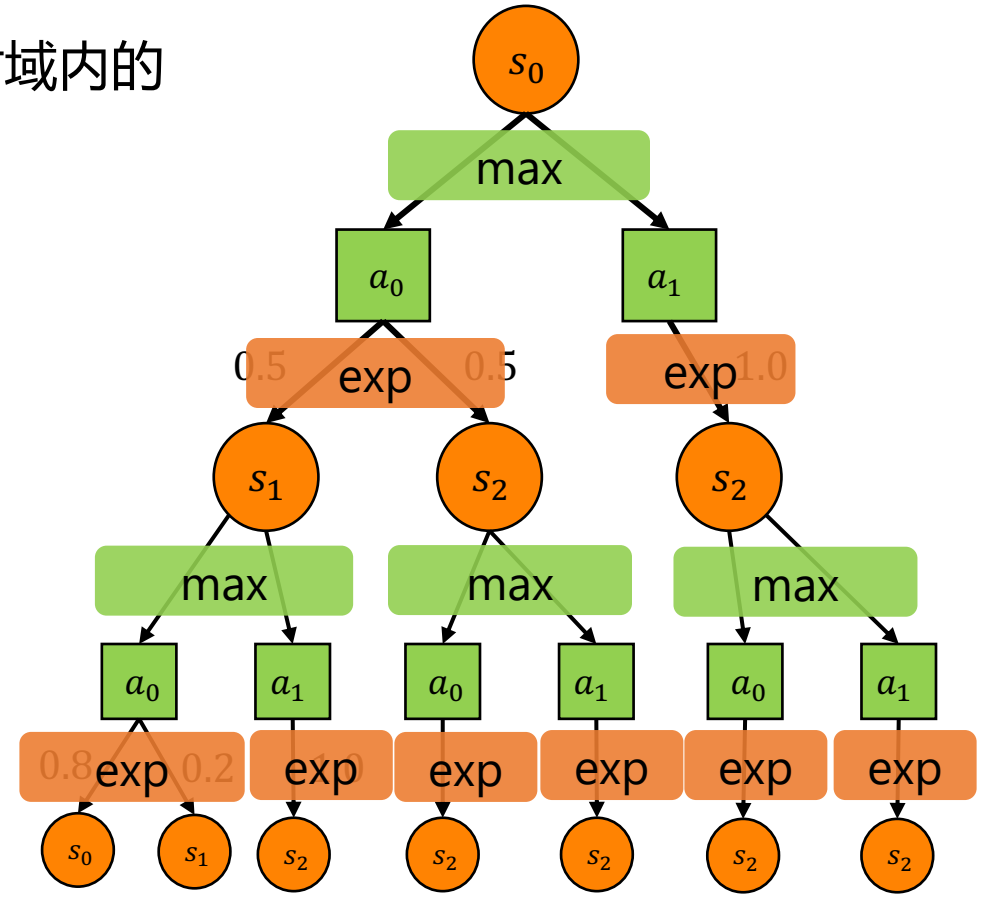
动态规划算法

- 优点：
  - 为所有未来可能遇到的状态计算了在时域内的最优策略，策略树可以运行 H 步
- 缺点：
  - 计算复杂度：

$$O(|A|^H|S|^H)$$

  - 对于示例问题：重复的分支太多



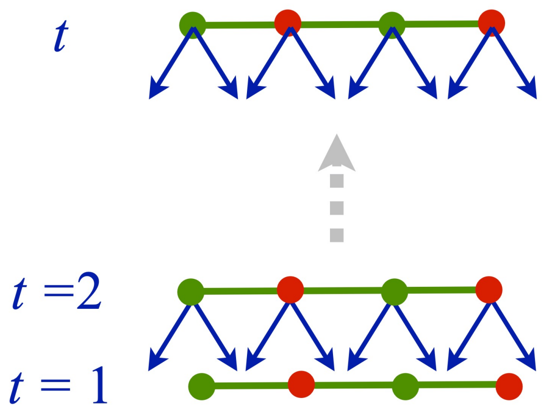


H=2

$$V^*(s) = \max_a R(s,a) + \sum_{s'} T(s,a,s')V^*(s')$$

值迭代（Value Iteration）

- 同时进行从所有状态出发的动态规划



- 初始化时时，对**所有状态 s** 赋值：

$$V^0(s)=0$$

- 每次迭代时，对**所有状态 s** 计算：

$$V^t(s)=\max_a R(s,a)+\gamma\sum_{s'}T(s,a,s')V^{t-1}(s')$$

- 迭代至所有 V(s) 收敛（即Bellman等式满足）

Initialization

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	[illegible]	0
0	0	0	0	0	0	0	0	0	0

Iteration 1

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.75	0
0	0	0	0	0	0	0	0.75	1	0
0	0	0	0	0	0	0	0	0	0

Iteration 2

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0.51	0
0	0	0	0	0	0	0	0.56	1.43	0
0	0	0	0	0	0	0	1.43	1.9	0
0	0	0	0	0	0	0	0	0	0

MDP规划问题（例）

值迭代（Value Iteration）

- 初始化时时，对所有状态 s 赋值：

  $V^0(s) = 0$

- 每次迭代时，对所有状态 s 计算：

  $V^t(s) = \max_a R(s,a) + \gamma \sum_{s'} T(s,a,s') V^{t-1}(s')$

- 迭代至所有 V(s) 收敛

Iteration 50

0	0	0	0	0	0	0	0	0	0
0	0.44	0.54	0.59	0.82	1.15	0.85	1.09	1.52	0
0	0.59	0.69	0	0	1.52	0	0	2.13	0
0	0.75	0.90	0	0	2.12	2.55	2.98	3.00	0
0	0.95	1.18	0	2.00	2.70	3.22	3.80	3.88	0
0	1.20	1.55	1.87	2.41	2.92	3.51	4.52	5.00	0
0	1.15	1.47	1.74	2.05	2.25	0	5.34	6.47	0
0	0.99	1.26	1.49	1.72	1.74	0	6.69	8.44	0
0	0.74	0.99	1.17	1.34	1.27	0	7.96	9.94	0
0	0	0	0	0	0	0	0	0	0

如何提取最优策略？

最优策略：

- $\pi^t(s) = \operatorname{argmax}_a R(s,a) + \gamma \sum_{s'} T(s,a,s') V^t(s')$

Value iteration 仍然是动态规划算法

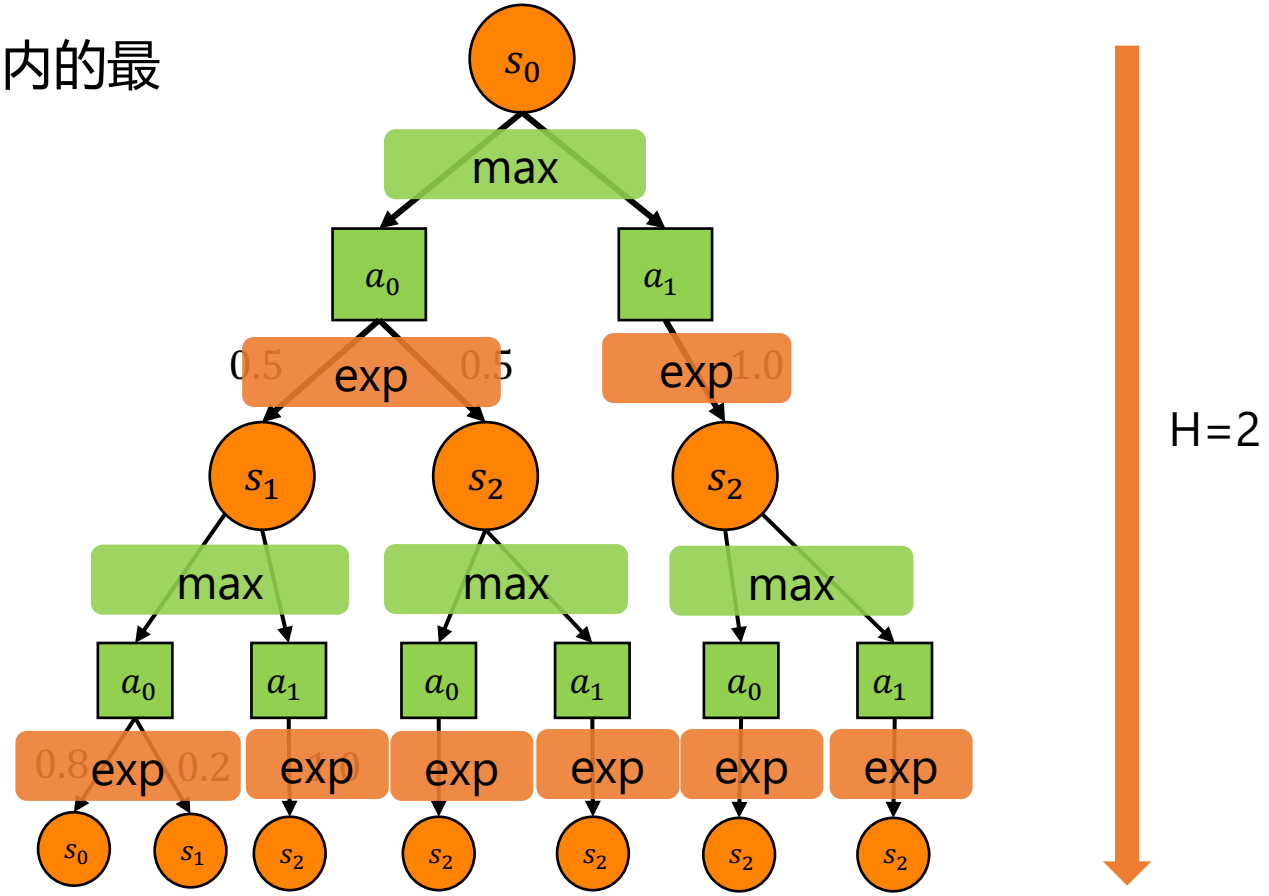
- 在整个状态空间中迭代（而不是在状态树中自下而上迭代）
- 节省了树中大量可能的重复运算

动态规划算法的优缺点

- 优点：
  - 计算了所有未来可能遇到的状态在时域内的最优策略，策略树可以运行 H 步
- 缺点：
  - 计算复杂度：

$$O(|A|^H|S|^H)$$

  - 值迭代: $O(H|A||S|^2)$
  - 对于自动驾驶问题：
    - |A|=?
    - |S|=?



$$V^*(s) = \max_a R(s,a) + \sum_{s'} T(s,a,s')V^*(s')$$

蒙特卡洛仿真（Monte Carlo Simulation）

- **估算随机函数期望值**：给定一个随机变量 $X \sim P(X)$，以及一个随机函数 $f(X)$，计算 $\mathbb{E}[f(X)]$

1. 根据定义计算：
   - $\mathbb{E}[f(X)] = \int p(x)f(x)\,dx$
2. 通过随机采样计算：
   - $\mathbb{E}[f(X)] \approx \frac{1}{N}\sum_{i=1}^{N} f(x_i)$，$x_i \sim P(X)$ 是 X 的样本（sample）
   - **大数定理**：在进行重复实验时，随着实验次数趋于无穷，样本的平均值将收敛于期望值
   - 这种计算 $\mathbb{E}[f(X)]$ 的方式称作 Monte Carlo Estimation
   - 采样 x_i 的过程称作 Monte Carlo Simulation

与 MDP 规划的联系？

- X: 采用策略 π 时未来可能的轨迹
- $f(X)$: 轨迹上的累积奖励
- $\mathbb{E}[f(X)]$：π 的策略价值 V^{π}

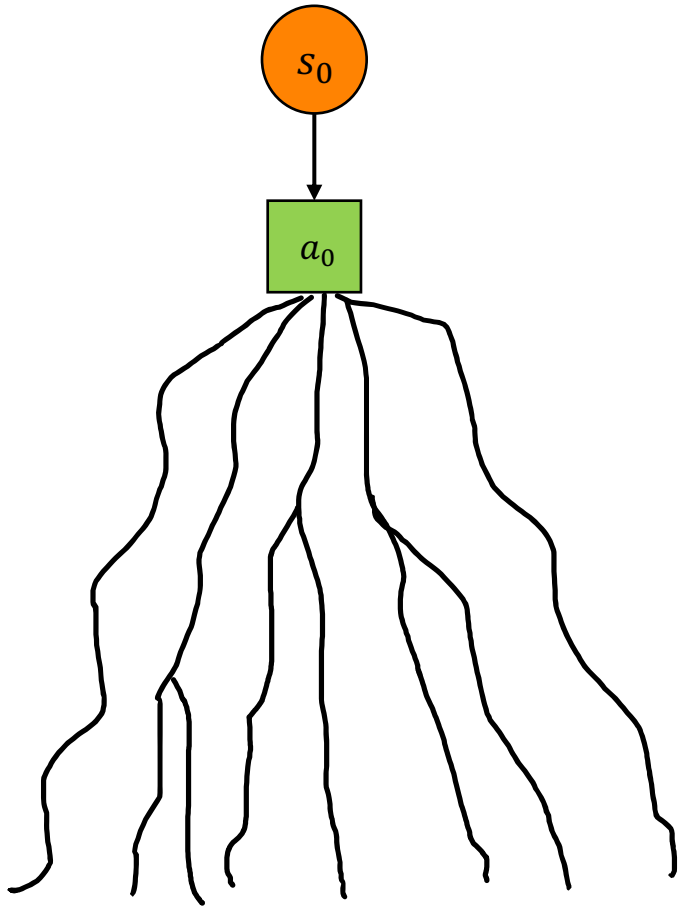
基于 MC simulation 的 MDP 规划

- 通过模拟策略 π 对未来轨迹和轨迹上的累积奖励进行随机采样
- 采样的次数越多，累积奖励的均值越接近策略 π 的价值

Monte Carlo Tree Search（MCTS）

MCTS-v1

- 给定一个起始状态 s_0 和一个策略 π
- 重复N次：
  - 从 s_0 开始进行 MC simulation
  - 根据 $a = \pi(s)$ 选择在该状态执行的动作
  - 执行动作后，从条件概率分布 $P(s'|s,a) = T(s,a,s')$ 中采样状态转移
  - 循环上述模拟，直到轨迹长度达到 H
  - 计算轨迹上的累积奖励：$\sum_{t=0}^{H} \gamma^t R(s_t, a_t)$
- 计算 N 次 MC simulation 的平均累积奖励，作为策略 π 的价值输出
- 算法计算了策略 π 的价值，而不是最优价值！



如何修改算法，以计算最优价值？

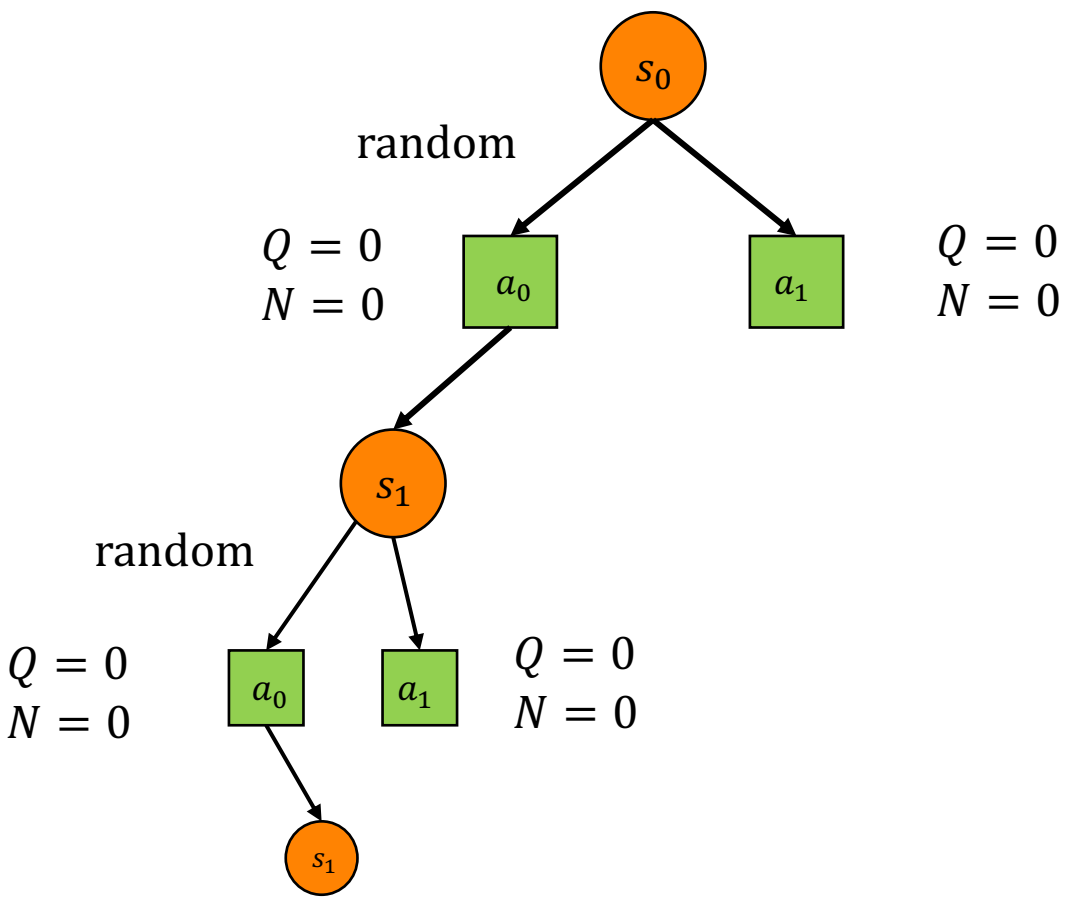
Monte Carlo Tree Search

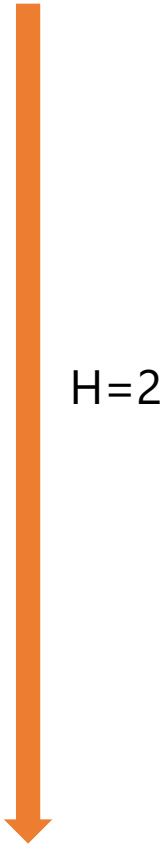
MCTS-v2

- 给定一个起始状态 s_0, 重复N次：
  - **Simulate**: 从 s_0 开始进行 MC simulation，直到轨迹长度达到 H
    - 最优动作选择：$a^* = \underset{a}{\text{argmax}}\, Q(s,a)$
    - 状态转移从 $P(s'|s,a) = T(s,a,s')$ 采样
  - **Expand:** 若 s' 对应的子节点不存在，创建叶节点
  - **Initialize:** 每扩展一个新节点，进行初始化：
    - $Q(s,a) = 0, N(s,a) = 0, \forall a \in A$
  - **Backup:** 从轨迹尾端（叶节点）开始向上回溯，途径每个节点 s 时计算：
    - 累积奖励：$q = R(s,a) + q'$ (回传自子节点)
    - 平均Q值：$Q(s,a) \leftarrow \frac{N(s,a)}{N(s,a)+1} Q(s,a) + \frac{1}{N(s,a)+1} q$
    - 累积访问次数：$N(s,a) \leftarrow N(s,a) + 1$
- 输出: $\pi^*(s_0) = \underset{a}{\text{argmax}}\, Q(s_0,a)$

演示：MCTS-V2

Trial 1: Simulate & Initialize



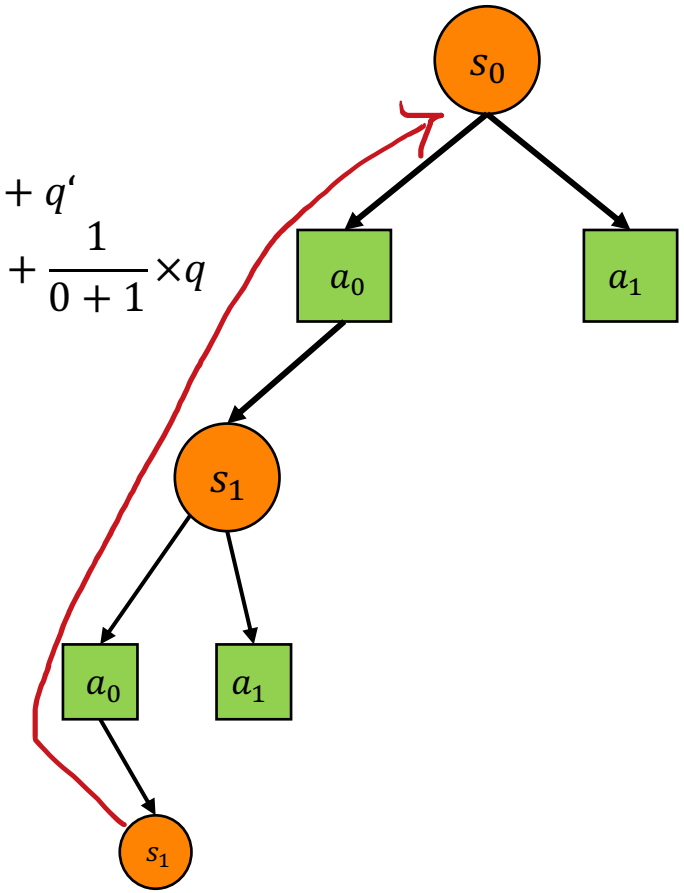


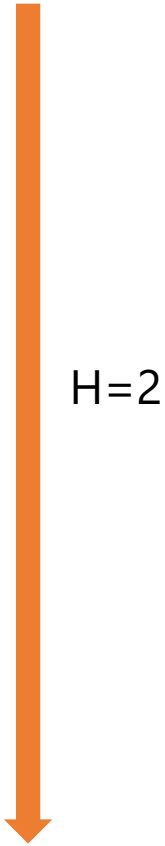
MCTS-V2

Trial 1: Backup

$$q = R(s_0, a_0) + q'$$
$$Q = \frac{0}{0+1} \times 0 + \frac{1}{0+1} \times q$$
$$N = 1$$

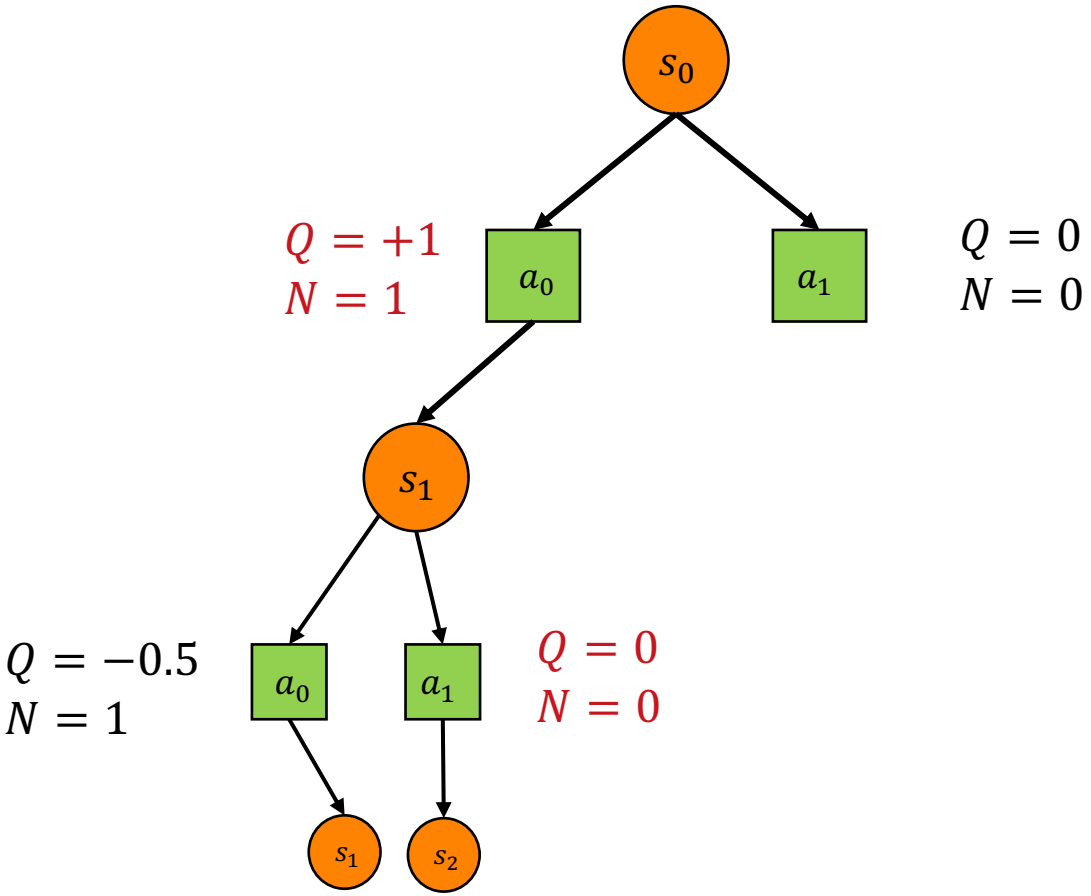
$$q = R(s_1, a_0)$$
$$Q = \frac{0}{0+1} \times 0 + \frac{1}{0+1} \times q$$
$$N = 1$$





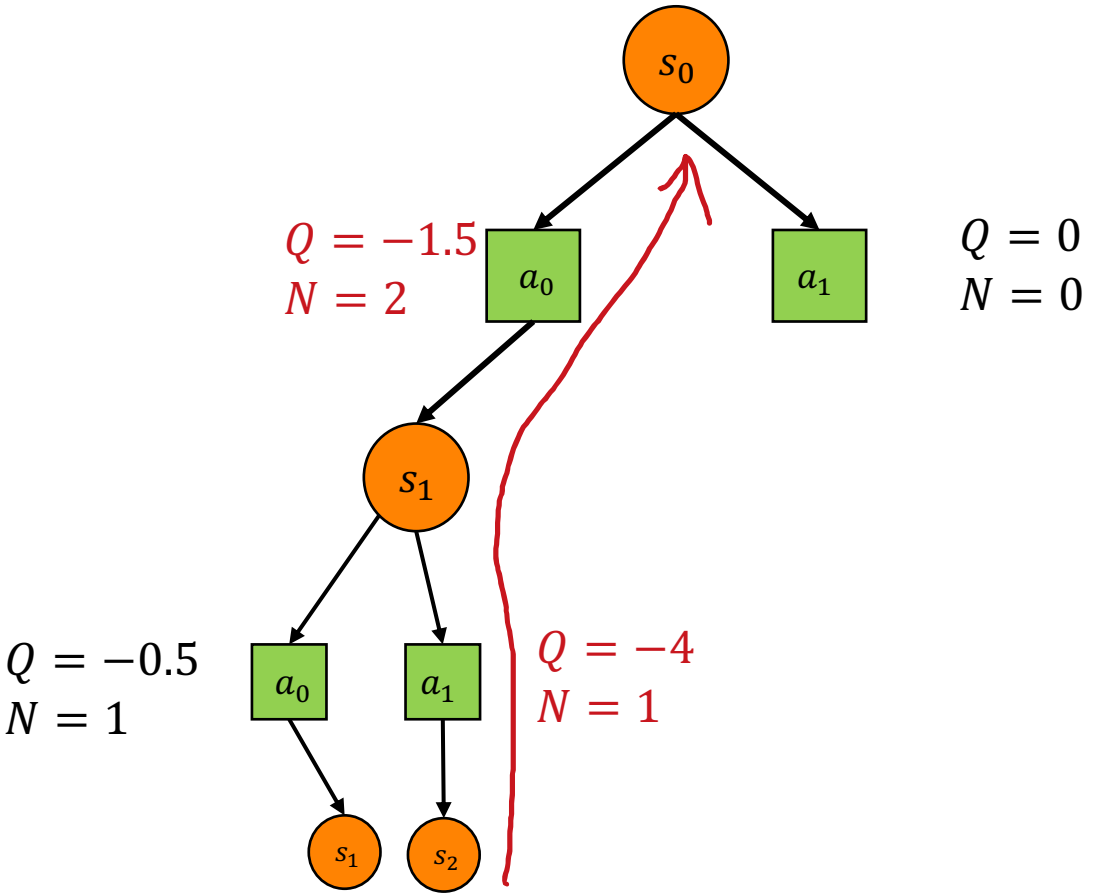
MCTS-V2

Trial 2: Simulate & Initialize



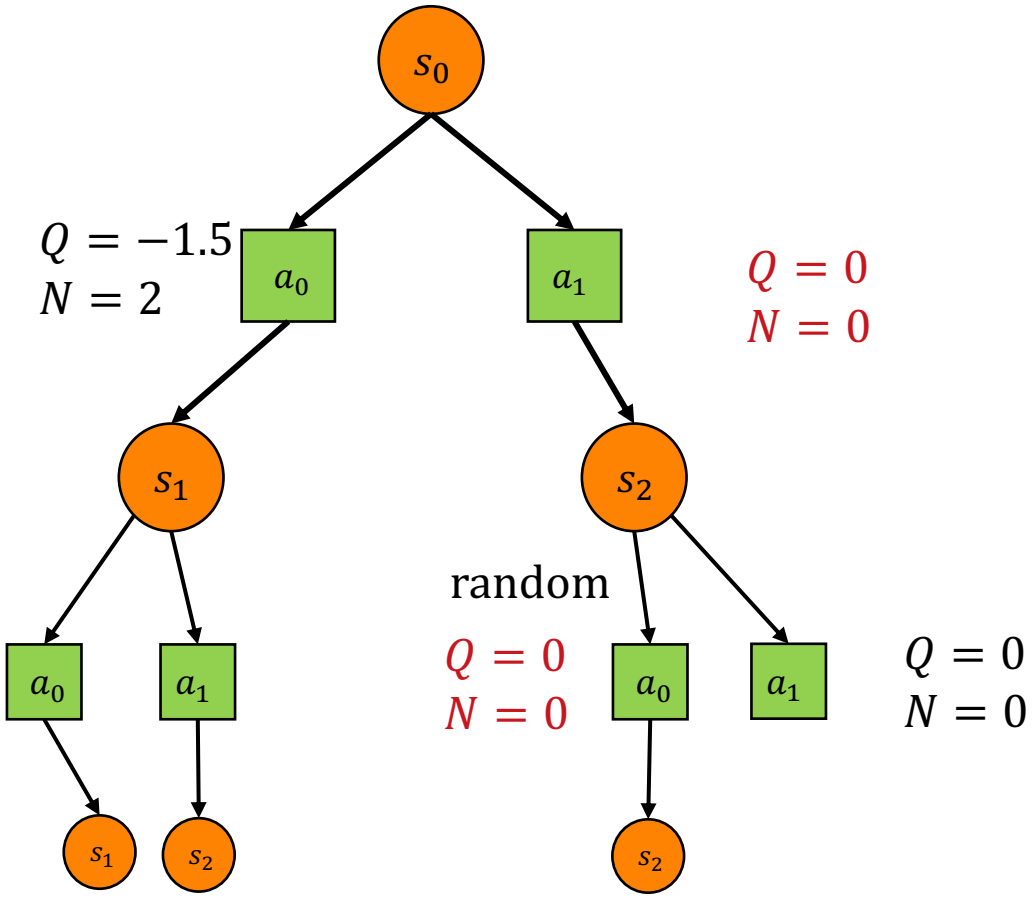
MCTS-V2

Trial 2: Backup



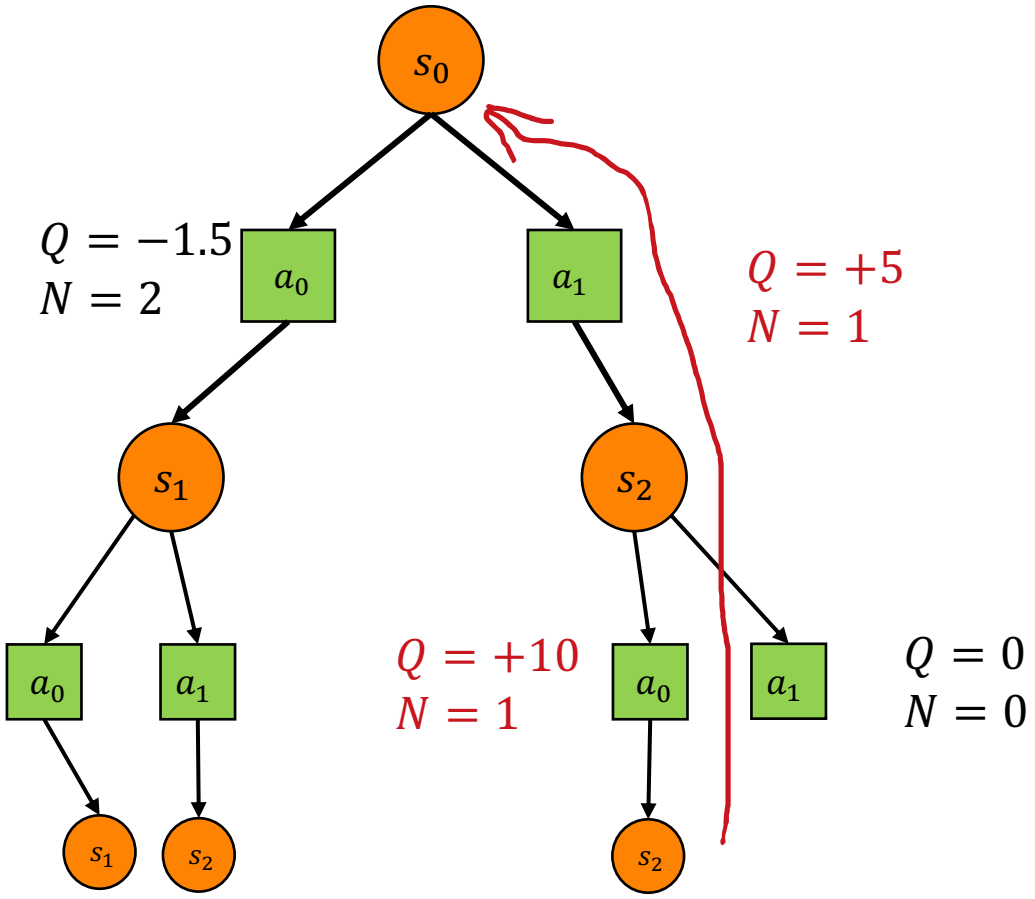
MCTS-V2

Trial 3: Simulate & Initialize



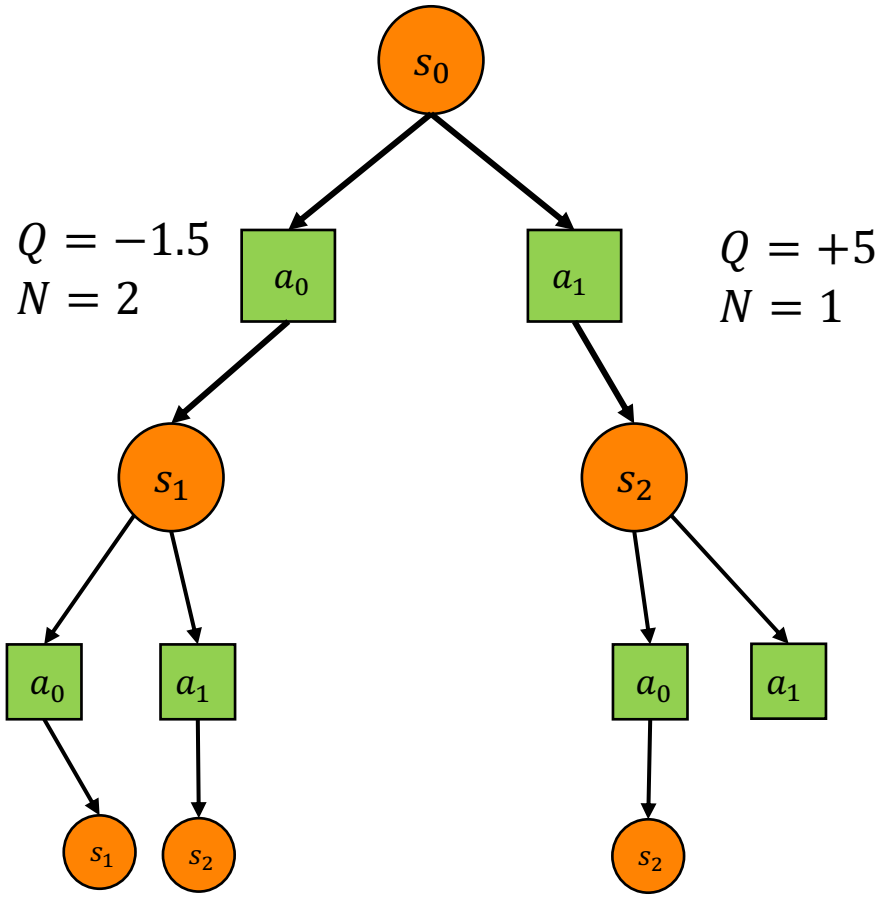
MCTS-V2

Trial 3: Backup



MCTS-V2

- 通过三次 MC simulation 估计的最优策略 $\pi^*(s_0)$？



MCTS-V2

可能存在的问题？

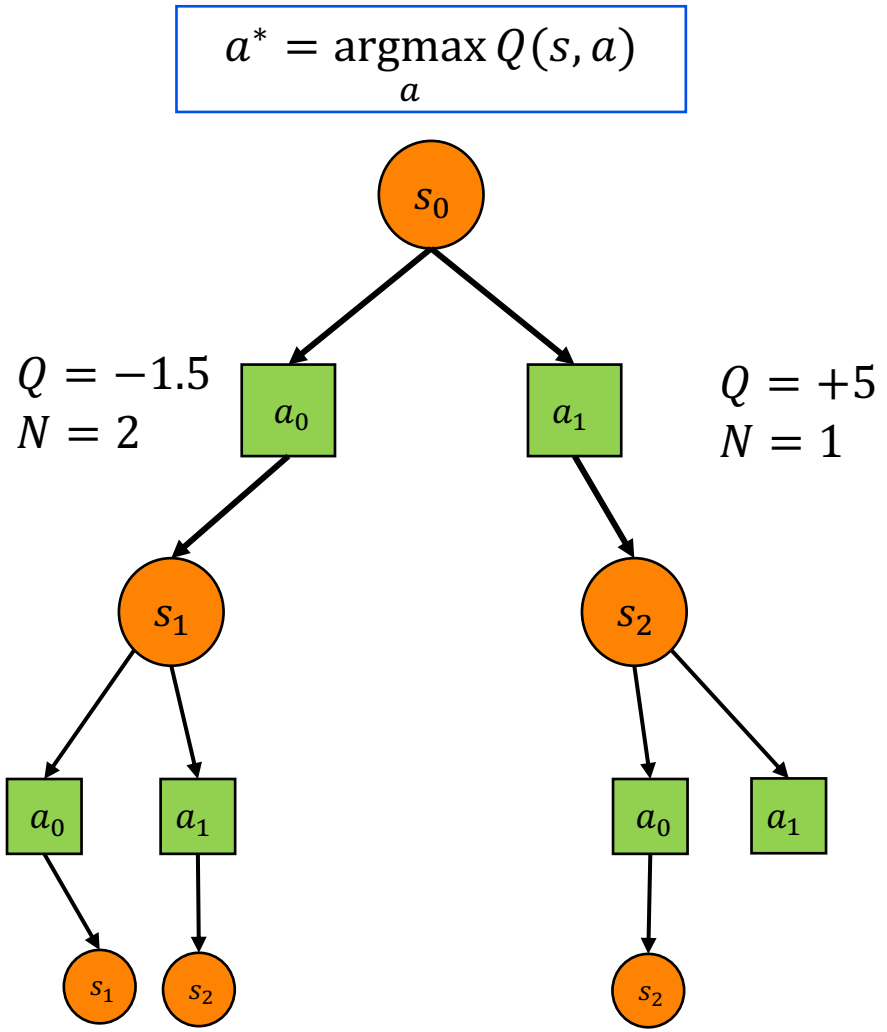
- 当采样轨迹少的时候，对 Q 值的估计非常不准确，而树搜索探索的方向由 Q 值直接决定。
- 当出现严重underestimate时，可能导致好的分支永远不被探索。

解决办法：鼓励非最优探索

- Upper confidence bound (UCB)

$$a^* = \underset{a}{\operatorname{argmax}}\, Q(s,a) + \alpha \sqrt{\frac{\log N(s)}{N(s,a)}}$$

Exploration bonus（具体形式推演自中心极限定理）



MCTS-V2

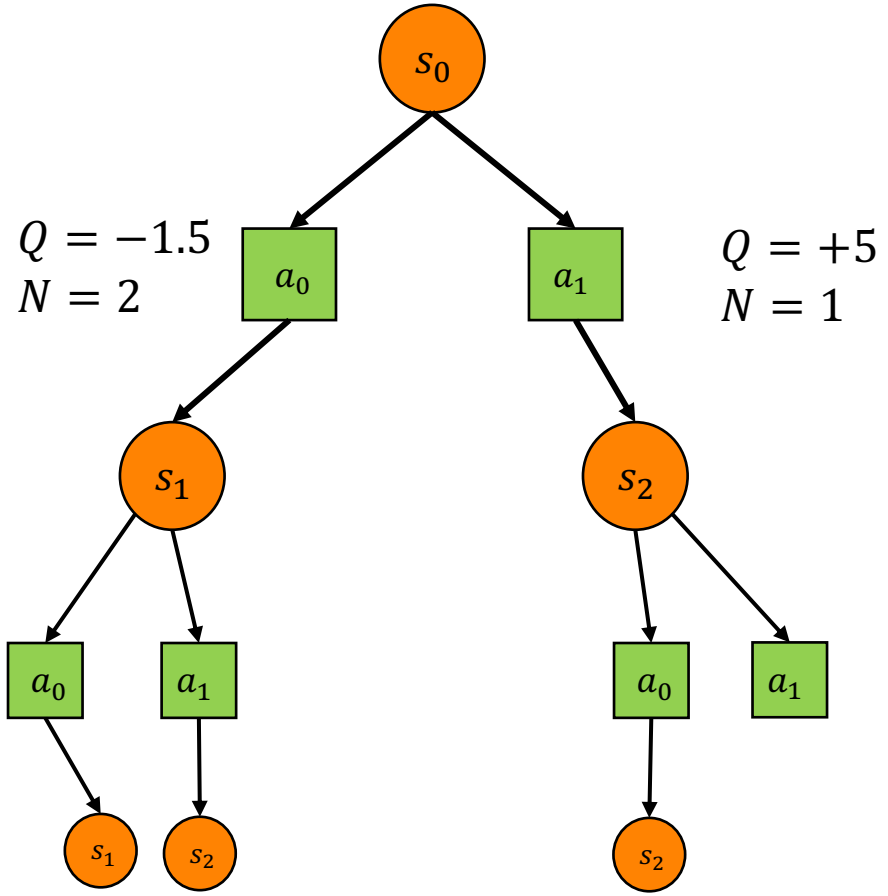
还可能存在的问题？

- 当 H 增大，进行一次完整模拟的时间将线性增长，同时可能轨迹的数量也将组合式增长
- 给定有限的规划时间有限（1s），规划结果将越来越不可靠

解决办法：逐渐深入的搜索（progressive deepening）

- 每扩展一个节点，都触发 backup
- 在初始阶段，exploration bonus 的影响较显著，MCTS 将广泛探索不同的动作分支（搜索树更多地被横向扩展）
- 随着访问数的增加，Q 值的差异越来越占据主导，MCTS将更优的分支进行更深入的探索
- **问题**：可能导致搜索过浅，失去对长期未来的理解

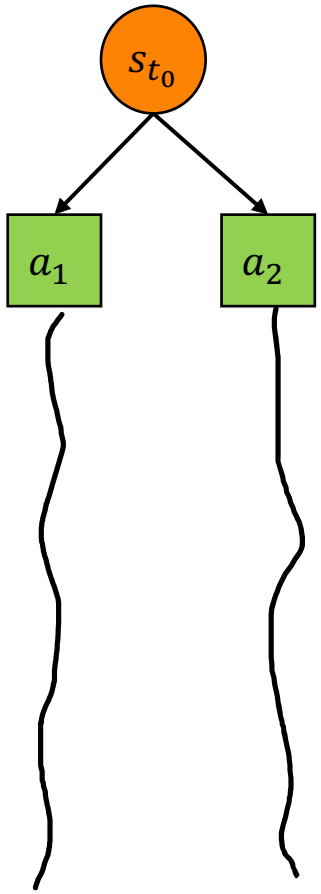
$$\text{UCB}: a^* = \underset{a}{\operatorname{argmax}}\, Q(s,a) + \alpha \sqrt{\frac{\log N(s)}{N(s,a)}}$$



MCTS-V2

- **搜索启发**：在新节点初始化的时候，不是简单设Q=0，而是利用对 Q* 值的简单估计
  - 用某个（基础）策略 π_0 的 Q^{π_0} 作为 Q^* 的估计值
  - 通过对 π_0 的MC模拟估计其 Q^{π_0}
- **Rollout 搜索启发** (类似MCTS-v1，但通常仅进行一次模拟):
  - 从新叶节点 s_{t_0} 开始进行 MC simulation
  - 除第一步外，总是根据 π_0 选择动作
  - 从 $P(s'|s,a) = T(s,a,s')$ 中采样状态转移
  - 循环上述模拟，直到深度达到 H
  - 计算轨迹上的累积奖励，作为s_{t_0}的初始Q值：
    $\sum_{t=t_0}^{H} \gamma^t R(s_t, a_t)$

π_0 通常称做 Rollout policy



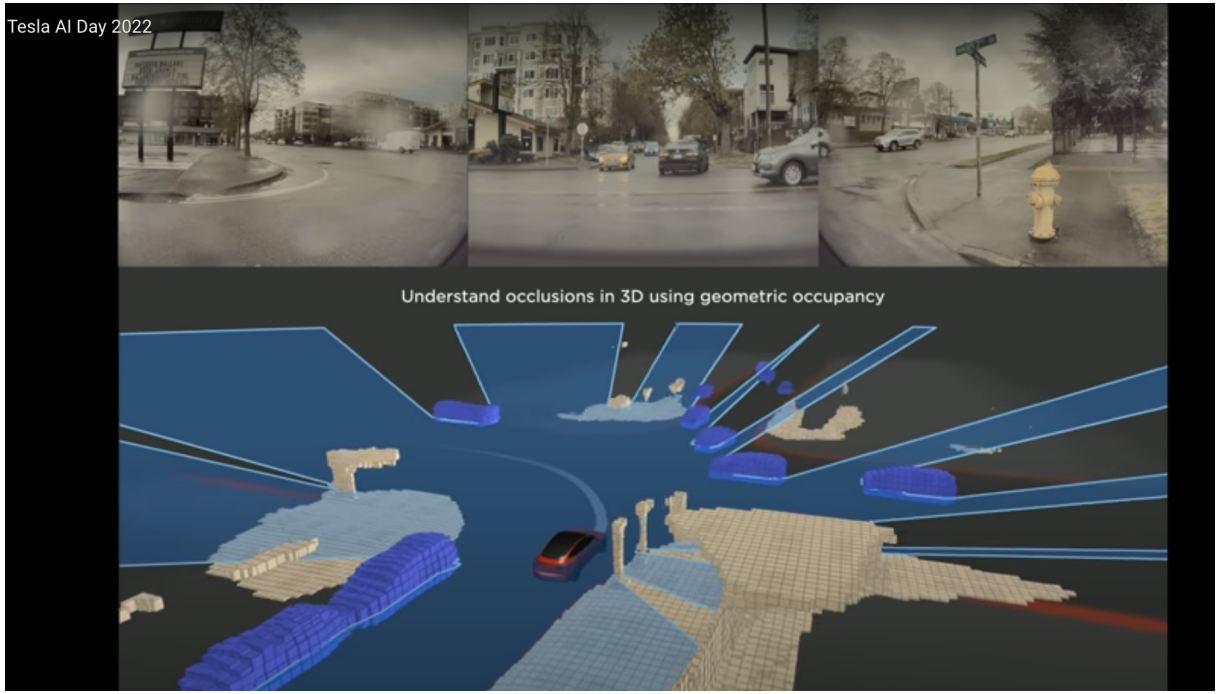
Monte Carlo Tree Search（MCTS）

最终版

- 给定起始状态 s_0, 重复N次：
  - **Simulate**: 从 s_0 开始进行 MC simulation
    - 动作选择：$a^* = \underset{a}{\text{argmax}}\, Q(s,a) + \alpha\sqrt{\frac{\log N(s)}{N(s,a)}}$
    - 状态转移从 $P(s'|s,a) = T(s,a,s')$ 采样
  - **Expand:** 若 s' 对应的子节点不存在，创建叶节点
  - **Initialize:** 每扩展一个叶节点，进行初始化：
    - $Q(s,a) = \textbf{Rollout}(s,a,\pi_0)$, $N(s,a) = 0, \forall a \in A$
  - **Backup:** 立即从叶节点开始向上回溯，途径每个节点 s 时计算：
    - 累积奖励：$q = R(s,a) + q'$ (回传自子节点)
    - 平均Q值：$Q(s,a) \leftarrow \frac{N(s,a)}{N(s,a)+1} Q(s,a) + \frac{1}{N(s,a)+1} q$
    - 累积访问次数：$N(s,a) \leftarrow N(s,a) + 1$
- 输出: $\pi^*(s_0) = \underset{a}{\text{argmax}}\, Q(s_0,a)$

MCTS应用于自动驾驶（Tesla 2022）

- 结构化道路上，自动驾驶汽车与多个个体的实时交互



MCTS应用于自动驾驶（Tesla 2022）

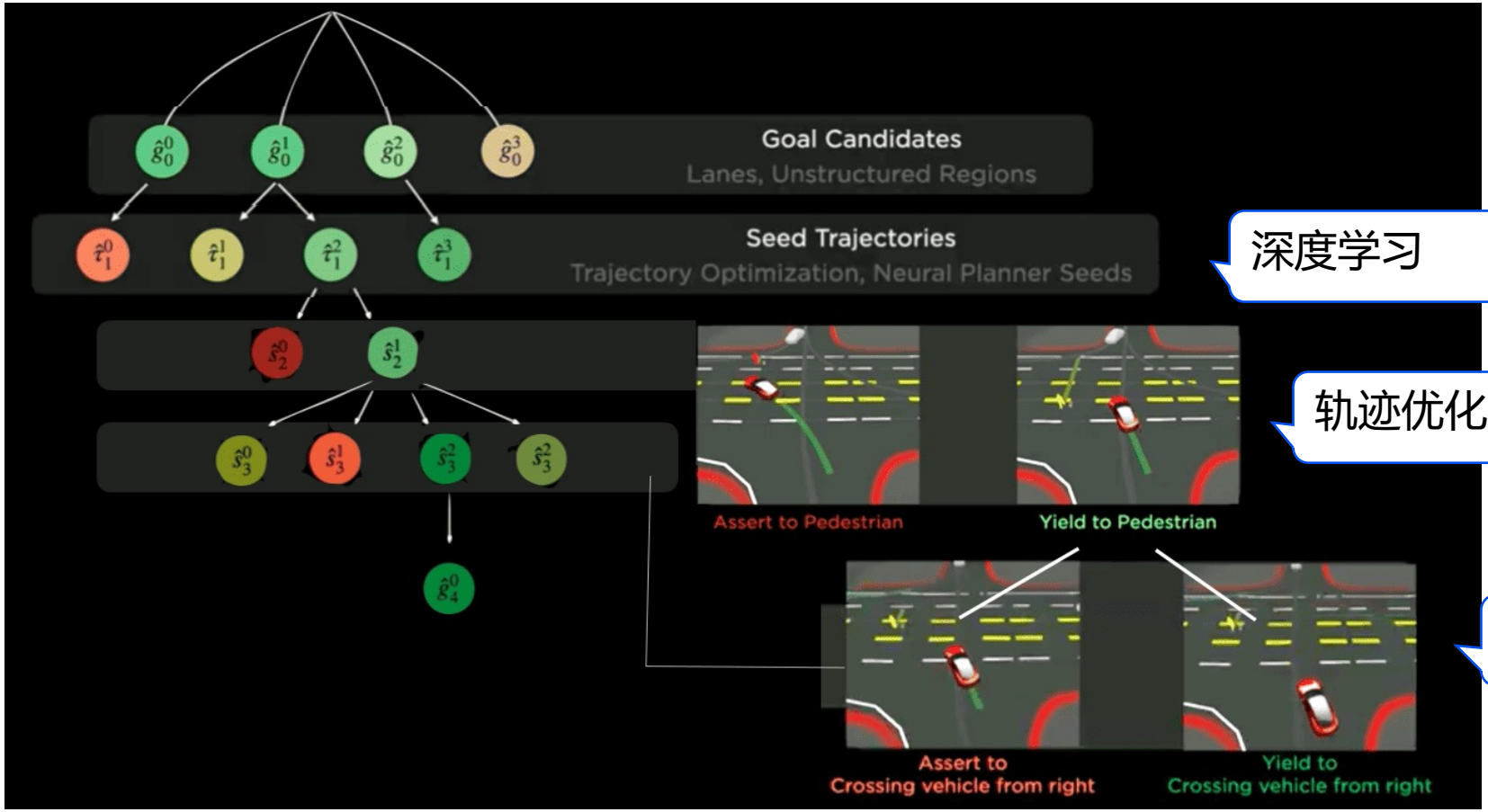
Tesla 的自动驾驶 MDP

- **状态**：自车与他车的运动学状态
- **动作**：分层决策
  - 第一层：局部车道图上的不同目标位置
  - 第二层：是否对第一个个体进行避让
  - 第三层：（在第一层决策的基础上）是否对第二个个体进行避让
  - …
- **状态转移**：（确定性的）运动轨迹
  - 对自车与他车进行联合轨迹优化：给定当前状态、目标位置与交互方式，由神经网络生成初始轨迹，由优化方法生成最终轨迹

MCTS应用于自动驾驶（Tesla 2022）

自动驾驶 MDP

- **状态**：自车与他车的运动学状态
- **动作与状态转移**：到达不同 goal 的、以不同方式他人交互的操作轨迹



深度学习

轨迹优化（人）

轨迹优化（人+车）

MCTS应用于自动驾驶（Tesla 2022）

自动驾驶 MDP

- **状态**：自车与他车的运动学状态
- **动作**：
  - 第一层：到达局部车道图上的不同目标位置
  - 第二层：是否对第一个个体进行避让
  - 第三层：（在第一层决策的基础上）是否对第二个个体进行避让
  - …
- **状态转移**：运动轨迹
  - 对自车与他车进行联合轨迹优化：给定目标位置与交互方式，由神经网络生成初始轨迹，由优化方法生成最终轨迹
- **奖励函数**：轨迹打分（trajectory scoring）
  - 碰撞检测、舒适度分析
  - 深度学习打分：接管可能性、行为的类人程度

认为可以准确知道他人的运动学状态

对他人/车运动轨迹的确定性假设

第十讲总结

- 马尔科夫决策过程（MDP）
  - 数学模型 $< S, A, T, R, \gamma >$
  - 开环、闭环计划
  - 价值 (Value)、策略 (Policy)
  - Bellman 最优性条件
- MDP 决策规划算法
  - 动态规划算法
  - 值迭代算法
  - MCTS（自动驾驶常用）
  - Tesla 的 MCTS 决策算法

第十讲扩展阅读

- **MDP 基础**： Kochenderfer, Mykel J. Decision making under uncertainty: theory and application. MIT press, 2015.
- **MCTS**： Browne, Cameron B., Edward Powley, Daniel Whitehouse, Simon M. Lucas, Peter I. Cowling, Philipp Rohlfshagen, Stephen Tavener, Diego Perez, Spyridon Samothrakis, and Simon Colton. "A survey of monte carlo tree search methods." IEEE Transactions on Computational Intelligence and AI in games 4, no. 1 (2012): 1-43.
- **Tesla AI day video**： https://youtu.be/ODSJsviD_SU

第十一讲：决策规划 II

蔡盼盼 副教授

上海交通大学清源研究院

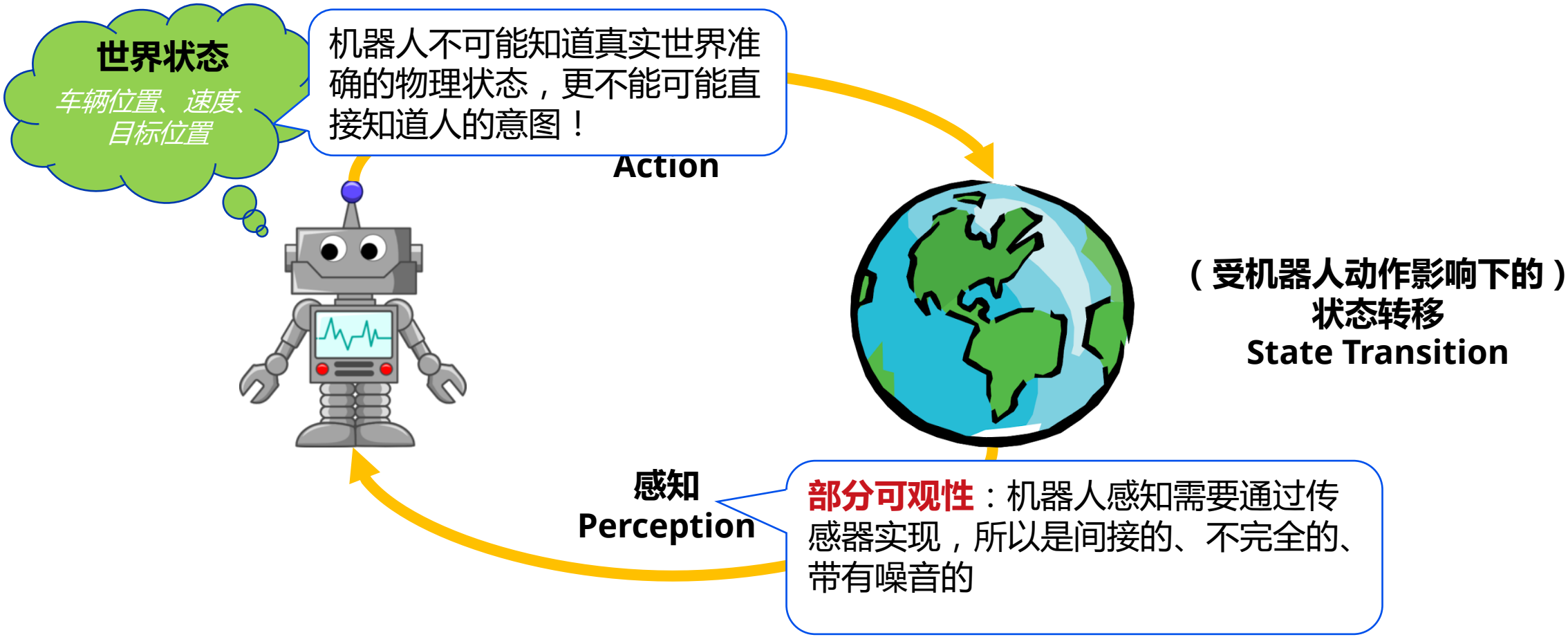
研究领域：机器人规划、机器人学习、自动驾驶

邮箱：Cai_panpan@sjtu.edu.cn

网站：https://cindycia.github.io/

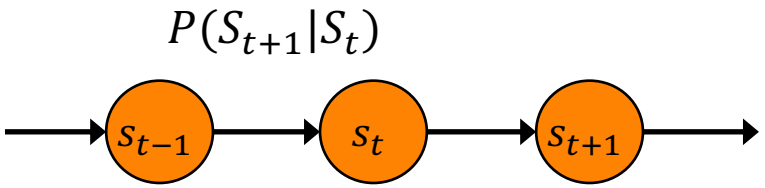
本讲材料与设计借鉴了David Hsu在NUS的课程 CS5478

机器人与随机环境的交互（真实版本）

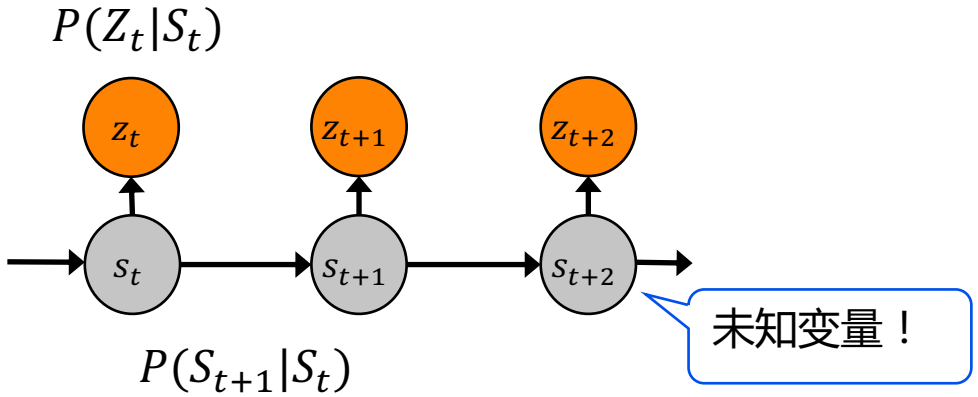


被动感知（部分可观版本）

- Markov chain

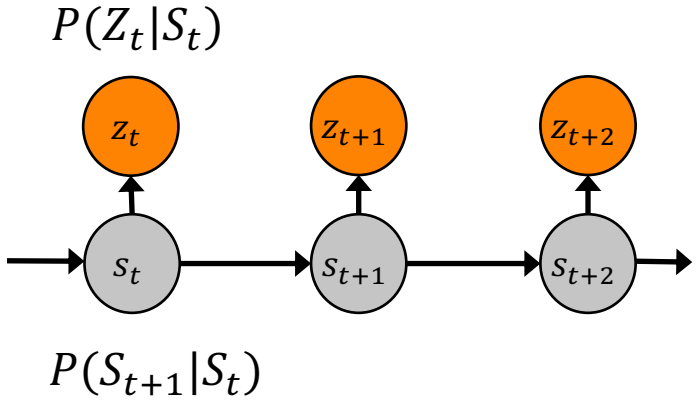


- Hidden Markov Model (HMM)



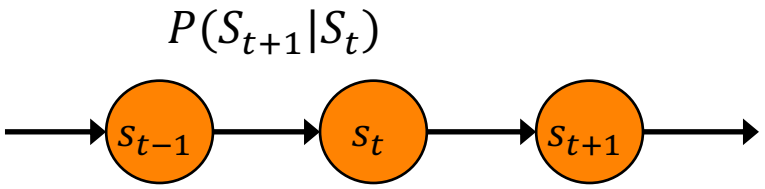
隐性马尔科夫模型（Hidden Markov Model）

- **HMM** 包含了一下 5 个元素：
  - S: 状态空间
    - 状态 s 可以表示场景中所有人、车所在的几何、行为状态
  - Z: 观察空间
    - 观察 z 可以表示传感器检测到的信息，如检测到的他人、他车位置、朝向、速度等
  - $T(s,s') = p(s'|s)$：状态转移函数
    - 表达世界变化的随机性：若当前世界处在状态 s, 下一步转移到状态 s′ 的概率是多少？
  - $O(s,z) = p(z|s)$ ：观察函数
    - 表达机器人感知的随机性：若当前世界处在状态 s, 获得观察 z 的概率是多少？
  - $b_0(s)$：初始时刻世界状态的概率分布

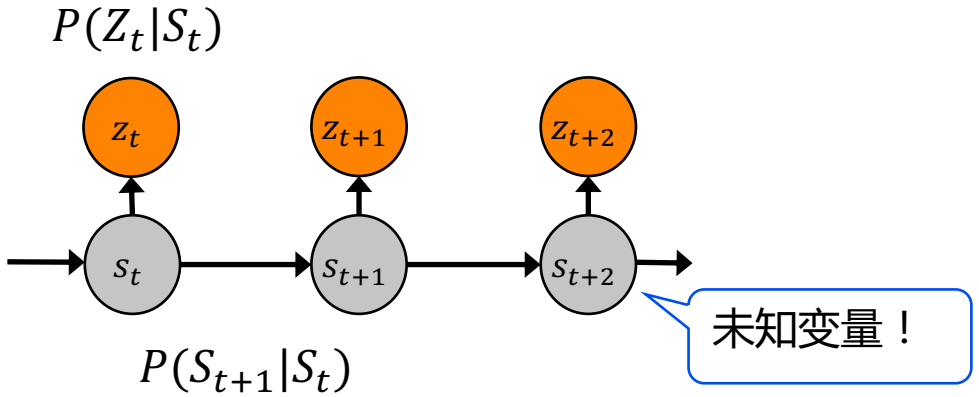


被动感知（部分可观版本）

- Markov chain



- Hidden Markov Model (HMM)



贝叶斯网络（Bayes Net）

- 节点：代表随机变量
- 有向边：代表变量对变量的直接影响
- 变量之间的影响由条件概率指定：
  $\boldsymbol{P(X \mid \text{parents}(X))}$

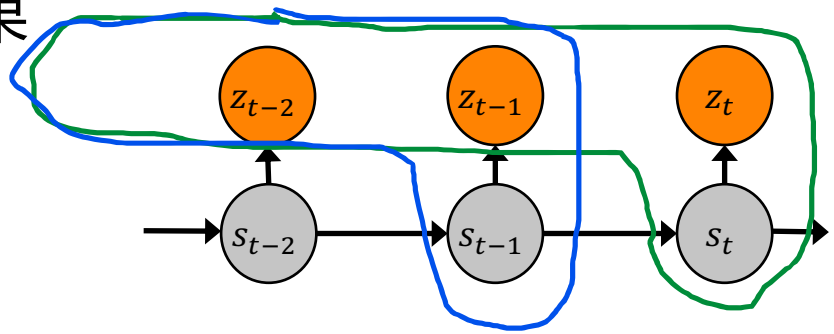
概率推断（probabilistic inference）

- 利用从部分节点观察到的信息，推断未知节点的概率分布
- 概率推断基于贝叶斯公式，利用图中的条件概率函数开展

$$p(x|y) = \frac{p(y|x)p(x)}{p(y)} = \eta p(y|x)p(x)$$

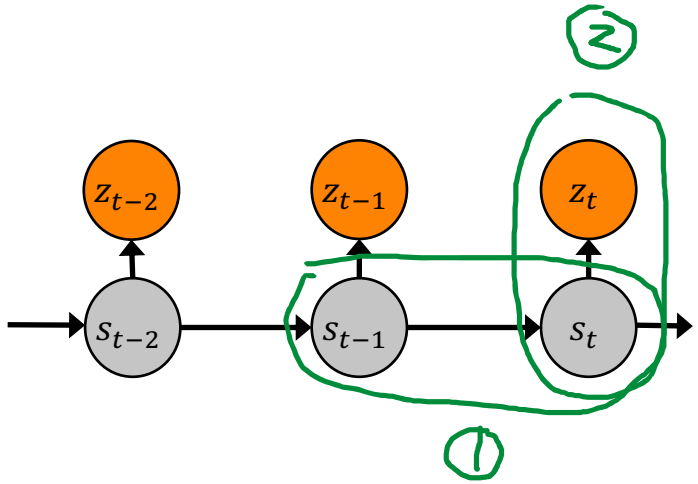
推断 $P(s_t|z_{1:t})$

迭代贝叶斯推断算法：在每一步，通过更新上一步的结果 $P(s_{t-1}|z_{1:t-1})$ ，得到 $P(s_t|z_{1:t})$



推断 $P(s_t|z_{1:t})$

迭代贝叶斯推断算法：在每一步，通过更新上一步的结果 $P(s_{t-1}|z_{1:t-1})$ ，得到 $P(s_t|z_{1:t})$



- **预测（Prediction）**：考虑随机状态转移

$$p(S_t = s') = \sum_{s \in S} p(S_t = s', S_{t-1} = s)$$

全概率公式

$$= \sum_{s \in S} p(S_t = s' | S_{t-1} = s)\, p(S_{t-1} = s)$$

Product rule

- **观察（Observation）**：考虑 z_t 中的信息

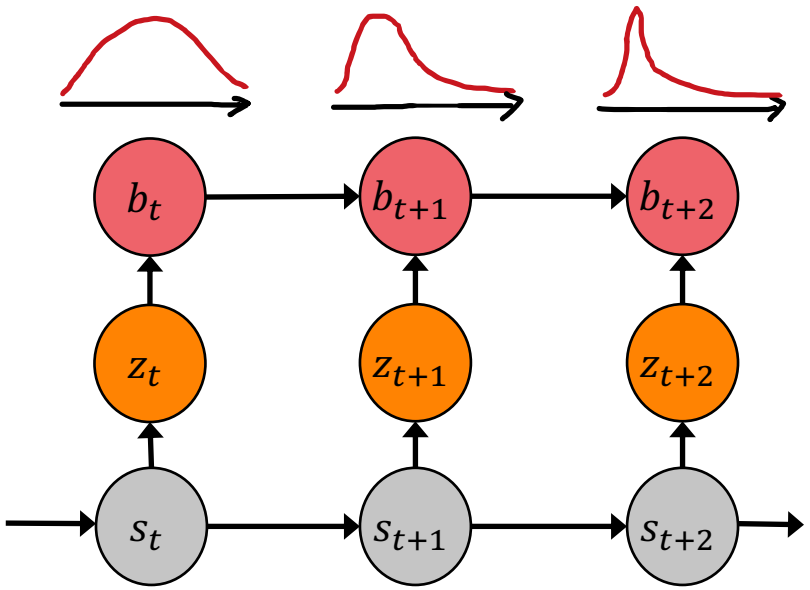
$$p(S_t = s | Z_t = z) = \eta\, p(Z_t = z | S_t = s)\, p(S_t = s)$$

Bayes rule

- 总结：
  - 上述过程称作滤波（filtering）
  - 推断的 $P(s_t|z_{1:t})$ 通常也称作**信念（Belief）**，因此该迭代推断过程也称作 Belief update 或 Belief Tracking

信念（Belief）

- **定义**：当前世界状态的概率分布，写作 b
  - $b(s)$ 代表世界处于状态 s 的概率



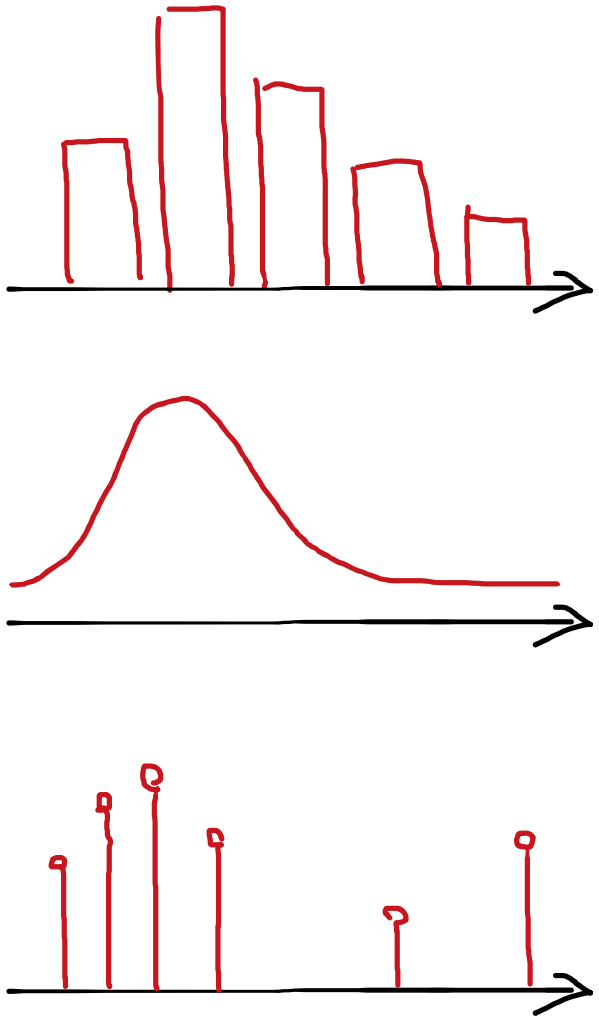
机器人认为世界发生了什么？（推断）

机器人观察到什么变化？（可知）

真实世界发生了什么事？（不可知）

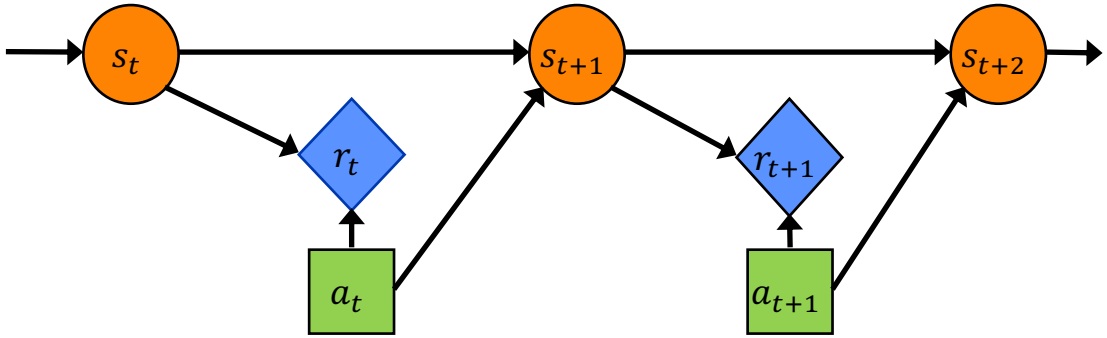
信念（Belief）的表达形式

- 直方图信念（**Histogram Belief**）
  - 状态空间 S 为包含有限个元素的离散集合
  - 给每个状态赋予一个概率值
  - 概率总和为 1
- 高斯信念（ **Gaussian belief** ）
  - 状态空间 S 为连续空间
  - 利用高斯分布表达不同状态的概率密度
- 粒子信念（ **Particle Belief** ）
  - 状态空间 S 为离散或连续空间
  - 利用粒子（对状态的采样）近似整个状态空间
  - 利用粒子的权重和粒子分布的密度表达状态概率



主动交互（部分可观版本）

- MDP

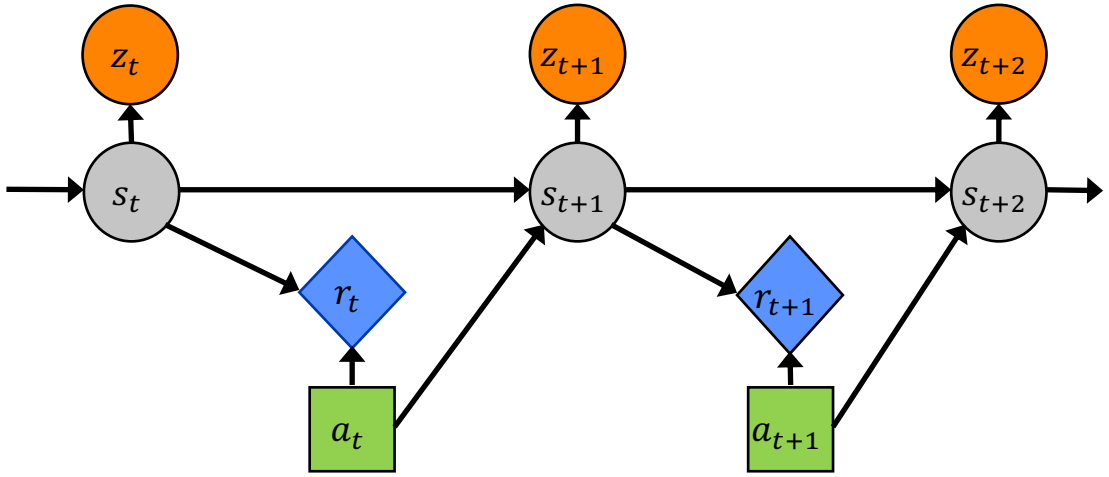


真实世界发生了什么事？（可知）

导致了什么奖励/惩罚？（可知）

机器人做了什么？（过去：可知；未来：求解）

- Partially Observable MDP (POMDP)



机器人观察到什么变化？（可知）

真实世界发生了什么事？（不可知）

导致了什么奖励/惩罚？（可知）

机器人做了什么？（过去：可知；未来：求解）

部分可观马尔科夫决策过程（POMDP）

- **POMDP** 包含 7 个元素，其中涉及 3 个空间 ……
  - S: 状态空间
    - 状态 s 可以表示场景中所有人、车所在的几何、行为状态
  - A: 动作空间
    - 动作 a 可以表示自车的行为
    - 底层行为：方向盘角度、加速度
    - 高层行为：跟车、变道、避让，…
  - Z: 观察空间
    - 观察 z 可以表示传感器检测到的信息，如检测到的他人、他车位置、朝向、速度等

MDP / HMM 皆有

MDP 具有

HMM 具有

部分可观马尔科夫决策过程（POMDP）

- **POMDP** 还包含 2 个条件概率函数 ……
  - $T(s,a,s') = p(s'|s,a)$：状态转移函数
    - 表达机器人行为带来的不确定性结果：若当前世界处在状态 s, 机器人执行动作 a, 世界下一步转移到状态 s′ 的概率是多少？
  - $O(a,s',z) = p(z|s',a)$ ：观察函数
    - 表达机器人感知的随机性：若机器人通过动作 a 使世界状态转移到 s′, 获得观察 z 的概率是多少？

MDP / HMM 皆有

HMM 具有

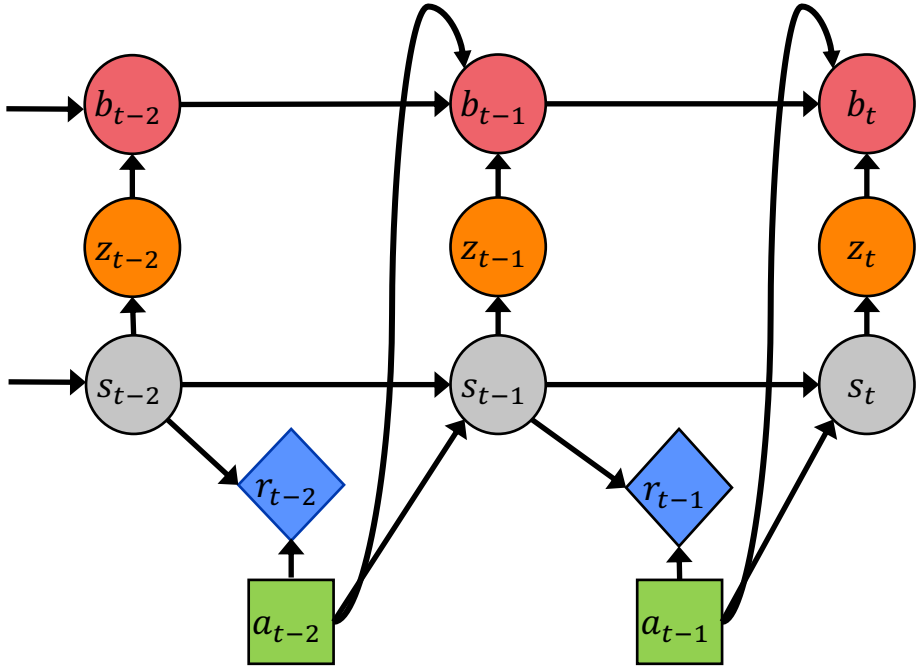
- 以及 1 个实函数和 1 个常量
  - $R(s,a)$ ：奖励函数
    - 表达机器人任务：若机器人在世界状态 s 执行动作 a, 所获得的即时奖励是多少？
    - 例：碰撞（ -30000 ），到达终点（ +100 ），变道（ -4 ），…
  - b_0：初始信念

MDP 具有

HMM 具有

POMDP 中的信念更新（Belief Update）

基于迭代贝叶斯推断，计算 $P(s_t|z_{1:t}, a_{1:t-1})$ ：



POMDP 中的信念更新（Belief Update）

基于迭代贝叶斯推断，计算 $P(s_t|z_{1:t}, a_{1:t-1})$：若机器人在第 $t-1$ 步执行了动作 a, 随后在第 t 步观察到 z：

- **预测（Prediction）**：考虑 a 导致的随机状态转移

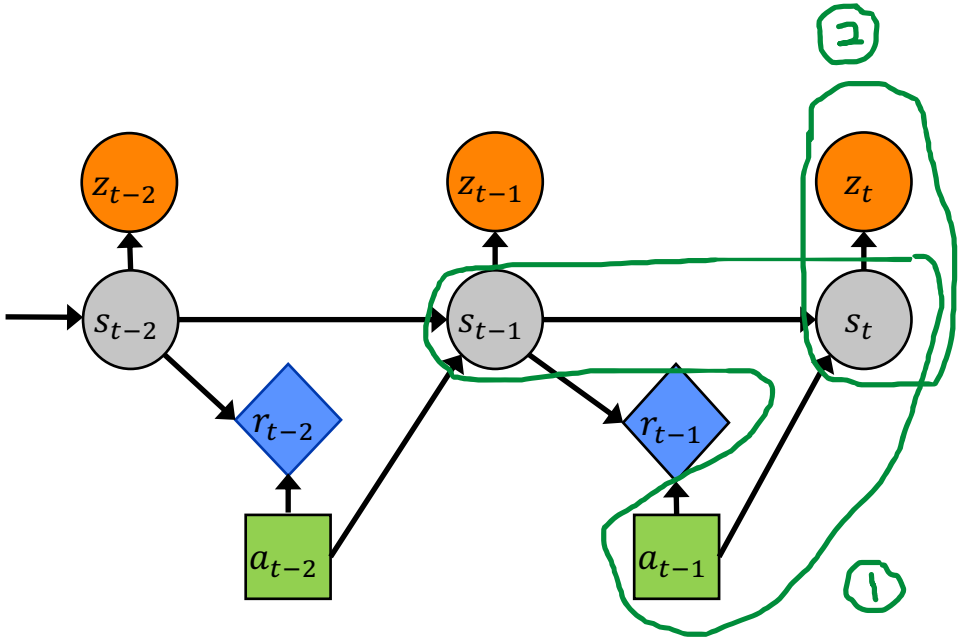
$$b_t(s'|a) = \sum_{s \in S} p(s', s|a)$$ 全概率公式

$$= \sum_{s \in S} p(s'|s, a)\, b_{t-1}(s|a)$$ Product rule

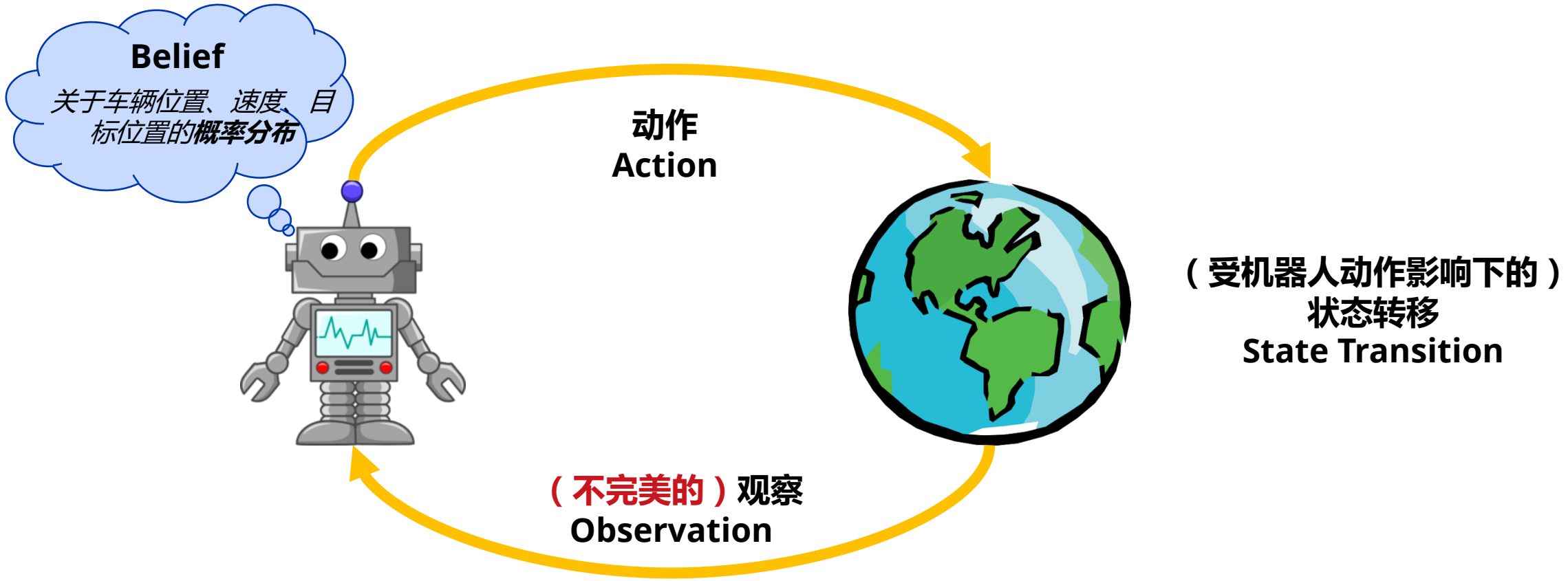
- **观察（Observation）**：考虑 z 中的信息

$$b_t(s'|a, z) = \eta\, p(z|s)\, b_t(s'|a)$$ Bayes rule

- 以上过程可以简写为：$\boldsymbol{b' = \tau(b, a, z)}$

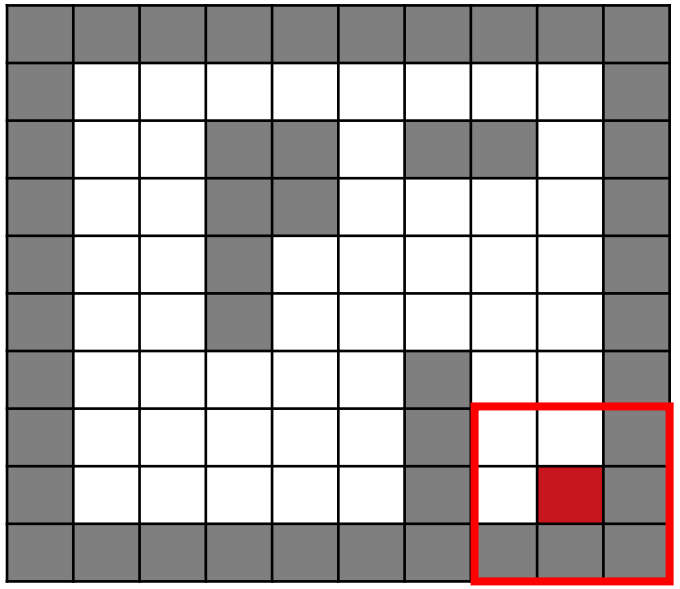


机器人与随机环境的交互（POMDP）

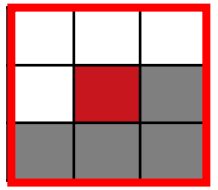


思考题：POMDP 建模

- 机器人的工作空间是一个 10x10 栅格：灰色代表障碍物，终点在右下角。机器人知道终点位置，但不知道障碍物的具体分布。
- 机器人的传感器只能观察到周围8个格子的占用状态，并且只有80%的正确率。
- 机器人在每一步可以选择移动到相邻4个格子。在没有障碍物的情况下，成功率为3/4。失败时，机器人会以同等的概率滑到其他3个格子。机器人也可以选择停在原地不动。
- 到达终点取得+1奖励，其余时间没有奖励。
- **问**：
  - 世界状态 s？
  - 观察 z？
  - 观察函数 O？
  - 初始信念？



$p(z|s,a)?$

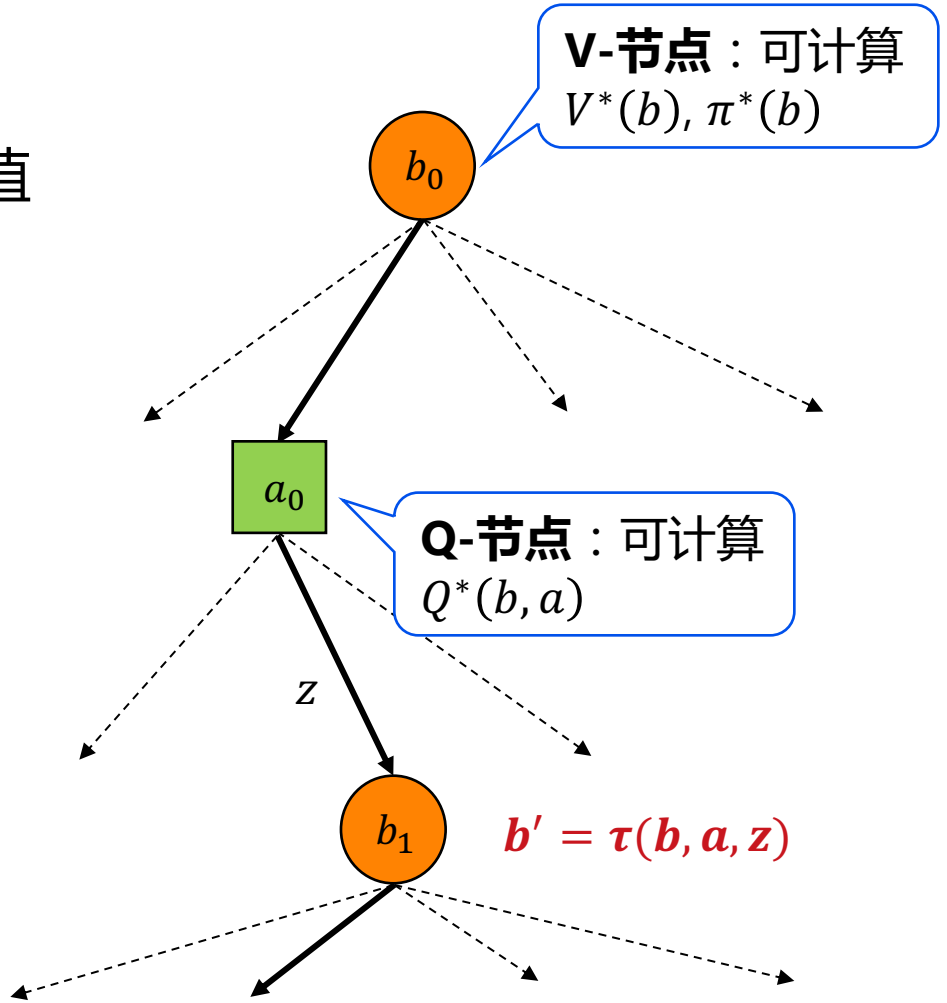


POMDP 规划

- **定义**： 找到一个**最优**策略，使得某个前瞻时域内的价值最大化
  - **价值（value）**： 执行策略可获得的累积奖励的期望值

$$V^{\pi}(b) = \mathbb{E}\left[\sum_{t=0}^{H} \gamma^t R(b_t, a_t) \,|b_0 = b, a_t = \pi(b_t)\right]$$

  - **策略（policy）**：$\boldsymbol{a_t = \pi(b_t)}$
    - 策略图
    - 策略树
    - 显式映射：公式、代码、DNN



最优量与 Bellman 最优性条件

- **最优值**（optimal value）：从信念 b 出发，所能达到的最优策略价值

$$V^*(b) = \max_a R(b,a) + \gamma \sum_{b'} T(b,a,b') V^*(b') \qquad \text{(Bellman-V)}$$

- **最优 Q 值**（optimal q-value）：从信念 b 出发，执行动作 a 之后，所能达到的最优价值

$$Q^*(b,a) = R(b,a) + \gamma \sum_{b'} T(b,a,b') V^*(b') \qquad \text{(Bellman-VQ1)}$$

$$V^*(b) = \max_a Q^*(b,a) \qquad \text{(Bellman-VQ2)}$$

- **最优策略**（optimal policy）：使价值最大化的策略

$$\pi^*(b) = \operatorname{argmax}_a Q^*(b,a), \text{或} \qquad \text{(Bellman-Pi1)}$$

$$\pi^*(b) = \operatorname{argmax}_a R(b,a) + \gamma \sum_{b'} T(b,a,b') V^*(b') \qquad \text{(Bellman-Pi2)}$$

$R(b,a)$ 怎么计算?

$T(b,a,b')$ 怎么计算?

POMDP中的 Bellman 最优性条件

- **$R(b,a)$ 怎么计算?**

$$R(b,a)=\sum_{s\in S}R(s,a)b(s)$$

- **$T(b,a,b')$ 怎么计算?**

$$T(b,a,b')=p(b'|b,a)=\sum_{z\in Z}p(b'|b,a,z)\,p(z|b,a)$$

  - **$p(b'|b,a,z)$ 是什么？**
    - b' 由 Belief update 唯一确定： $b'=\tau(b,a,z)$
    - 转移概率分布是狄利克雷函数：$p(b'|b,a,z)=\delta_{b'}(\tau(b,a,z))$
  - **$p(z|b,a)$ 是什么？**

$$p(z|b,a)=\sum_{s'\in S}p(z|b,a,s')p(s'|b,a)$$

全概率公式：引入 s′

$$=\sum_{s'\in S}p(z|a,s')\sum_{s\in S}p(s'|s,a)b(s)$$

给定 s′ 时，z 与 b 条件独立
全概率公式：引入 s

$$=\sum_{s'\in S}O(a,s',z)\sum_{s\in S}T(s,a,s')b(s)$$

POMDP中的 Bellman 最优性条件

- 代入上述推导至:

$$V^*(b) = \max_a R(b,a) + \gamma \sum_{b' \in B} T(b,a,b') V^*(b') \quad \text{(Bellman-V)}$$

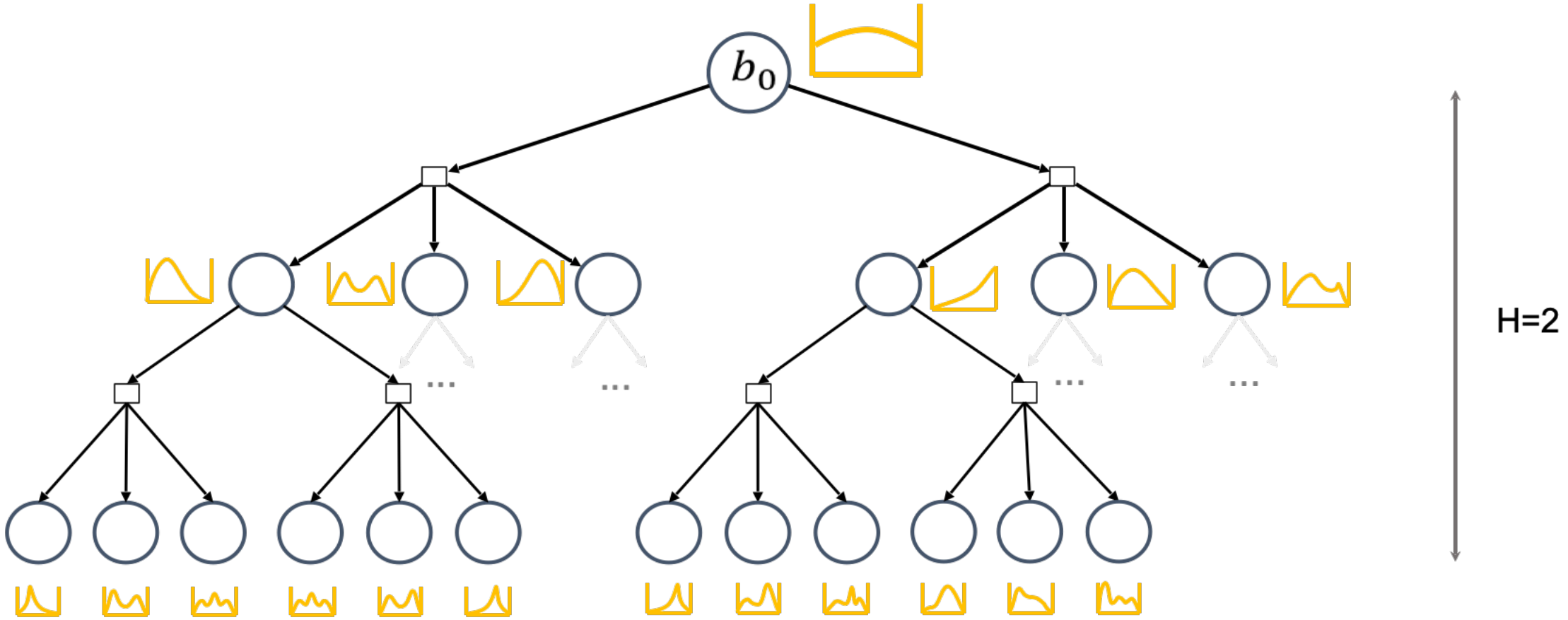
后得到：

$$V^*(b) = \max_a R(b,a) + \gamma \sum_{z \in Z} p(z|b,a) V^*(b') \quad \textbf{(Bellman-POMDP)}$$

其中：

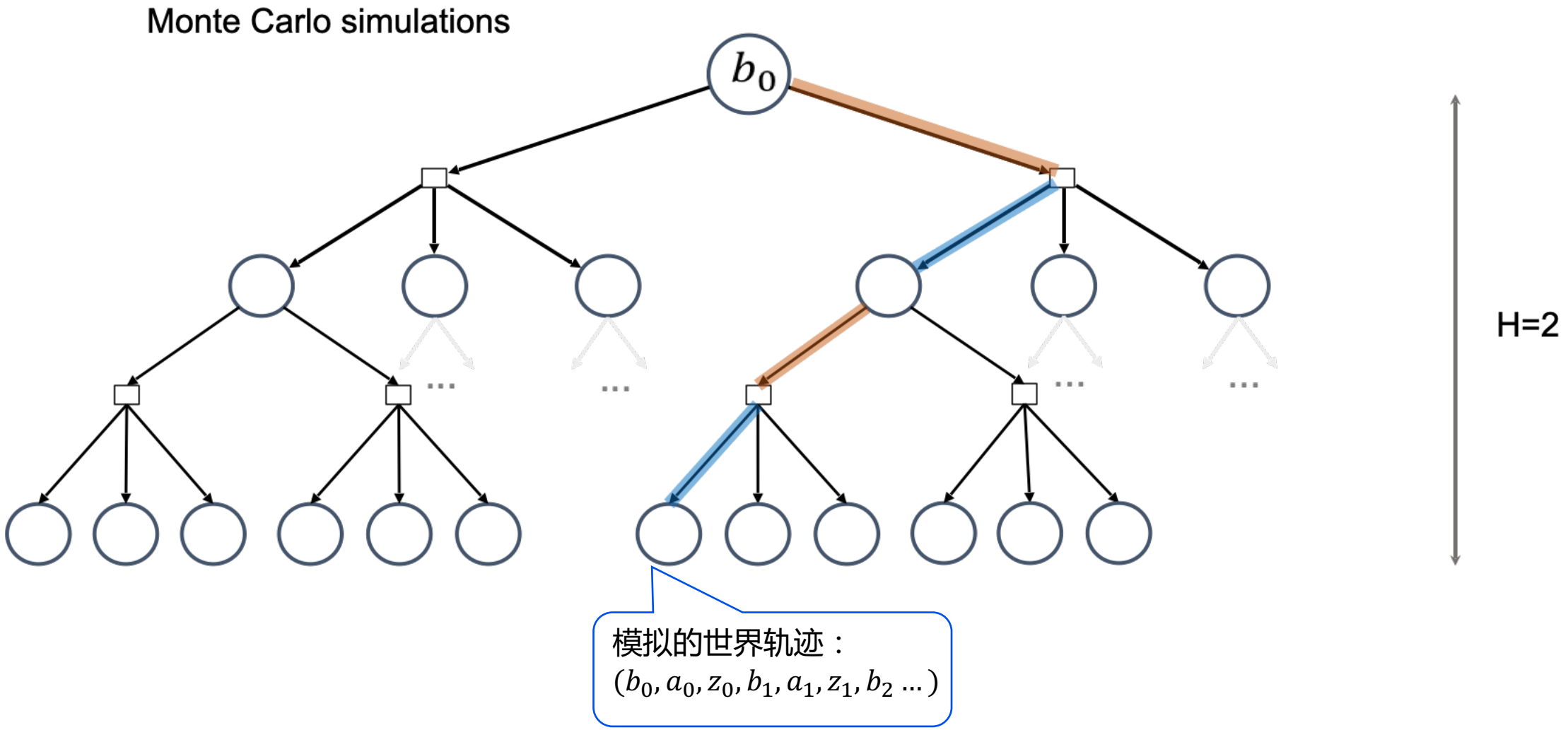
- $b' = \tau(b,a,z)$
- $R(b,a) = \sum_{s \in S} R(s,a) b(s)$
- $p(z|b,a) = \sum_{s' \in S} O(a,s',z) \sum_{s \in S} T(s,a,s') b(s)$

信念树（Belief Tree）

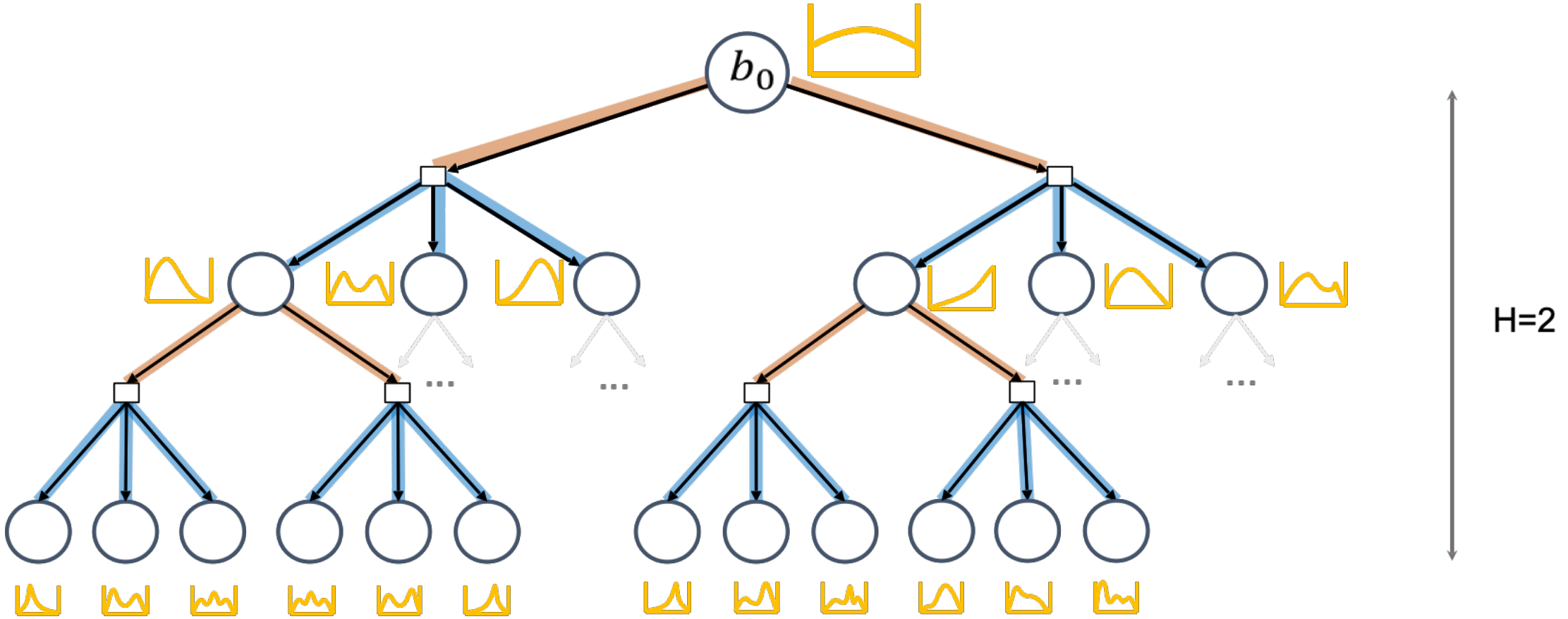


Belief tree with 2 actions and 3 observations

信念树（Belief Tree）

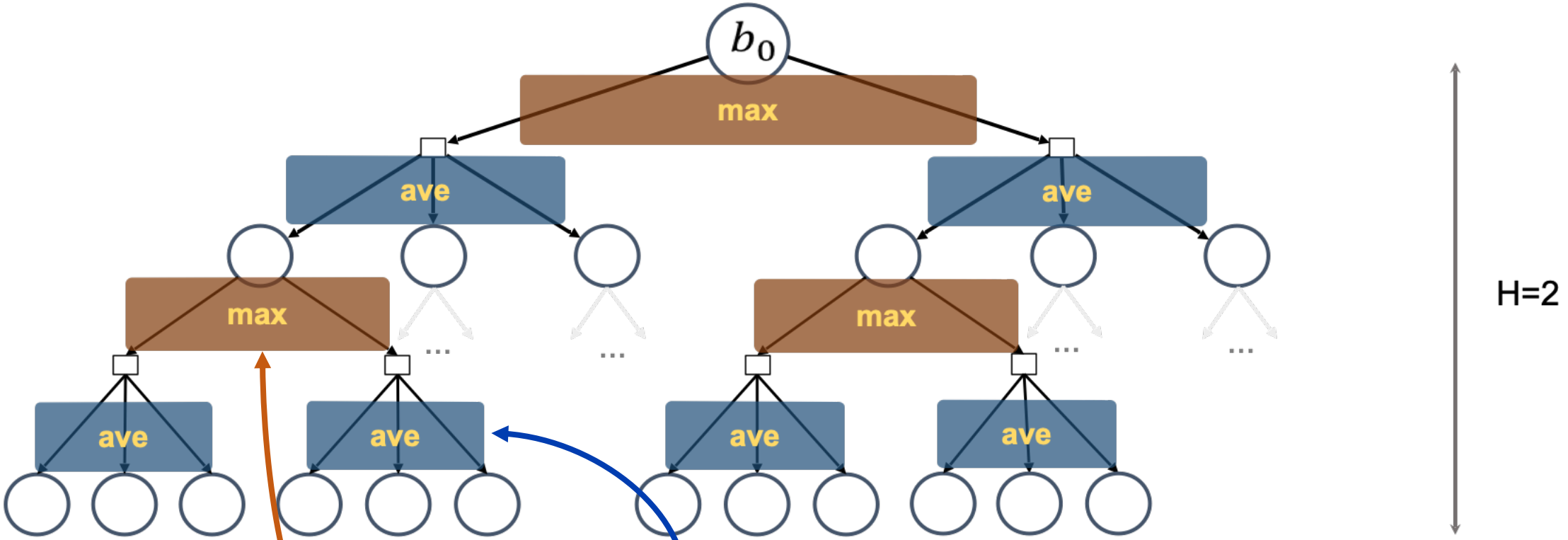


信念树（Belief Tree）



Belief tree with 2 actions and 3 observations

信念树中的动态规划



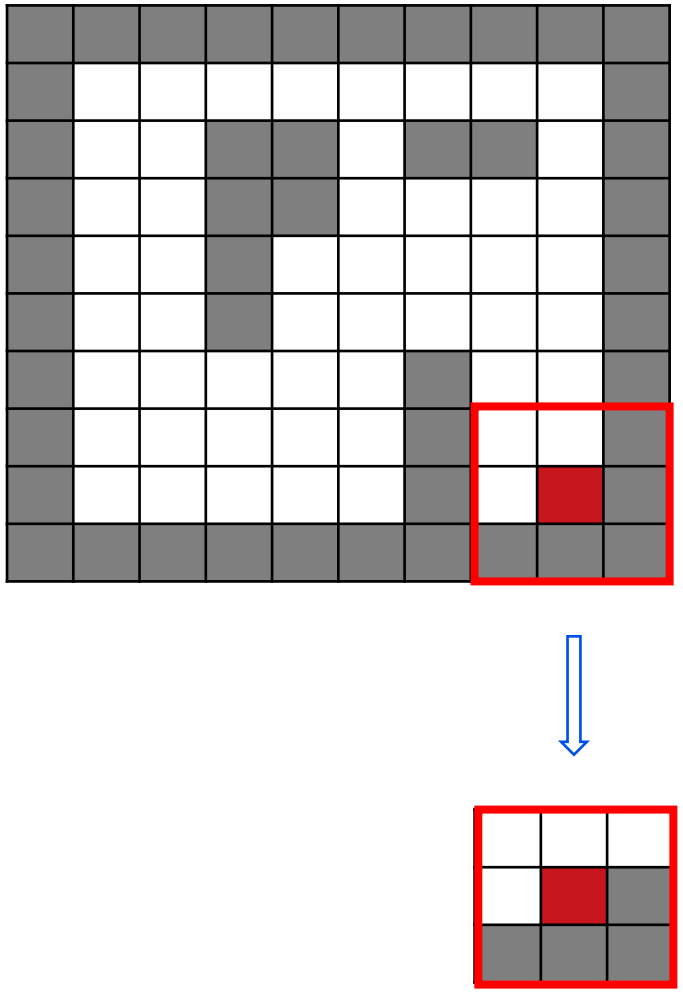
$$V^*(b) = \max_a R(b,a) + \gamma \sum_{z \in Z} p(z|b,a) V^*(b') \quad \textbf{(Bellman-POMDP)}$$

计算复杂度：$O(|A|^H |Z|^H)$

观察空间 Z 的大小？

- 机器人的工作空间是一个 10x10 栅格：灰色代表障碍物，终点在右下角。机器人知道终点位置，但不知道障碍物的具体分布。
- 机器人的传感器只能观察到周围8个格子的占用状态，并且只有80%的正确率。
- 机器人在每一步可以选择移动到相邻4个格子。在没有障碍物的情况下，成功率为3/4。失败时，机器人会以同等的概率滑到其他3个格子。机器人也可以选择停在原地不动。
- 到达终点取得+1奖励，其余时间没有奖励。

- **|Z| = 256 x 100 !**



实用 POMDP 规划算法

QMDP：利用启发函数近似计算

$$V^*(b) = \max_a R(b,a) + \gamma \sum_{z\in Z} p(z|b,a)V^*(b')$$

- 只在选择第一步动作时进行分支，其余全部由启发函数 h 计算：

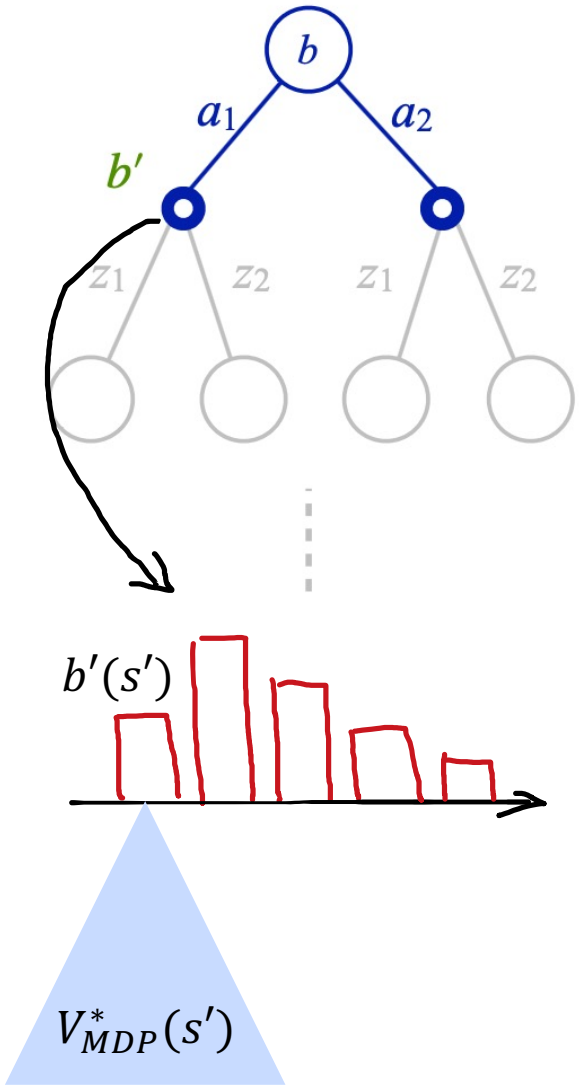
$$V^*(b) \approx \max_a R(b,a) + h(b')$$

  - 这里 $b'(s') = p(s'|b,a)$ 只考虑了动作，不考虑后续观察得到的信息
  - **启发函数 h？**

$$h(b) = \sum_{s\in S} b(s)\, V^*_{MDP}(s)$$

  - **思考**：$h(b)$ 与 $V^*(b)$ 的关系是？

(1) 上界　　(2) 下界　　(3) 没有界关系



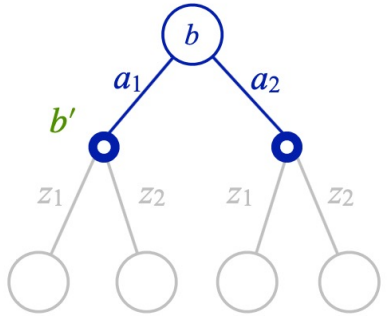
QMDP 算法

- 给定 $b_0 = b$, 对每个动作 a 进行：
  - 对所有 $s' \in S$ 计算：
    - $b'(s') = p(s'|s,a)b(s)$
    - 利用 VI 或者 MCTS 计算 $V^*_{MDP}(s')$
  - 计算 Q 值：

$$Q_{MDP}(b,a) = R(b,a) + \sum_{s' \in S} b'(s')\, V^*_{MDP}(s')$$

- 计算最优策略：

$$\pi^*(b) = \arg\max_a Q_{MDP}(b,a)$$



优点：

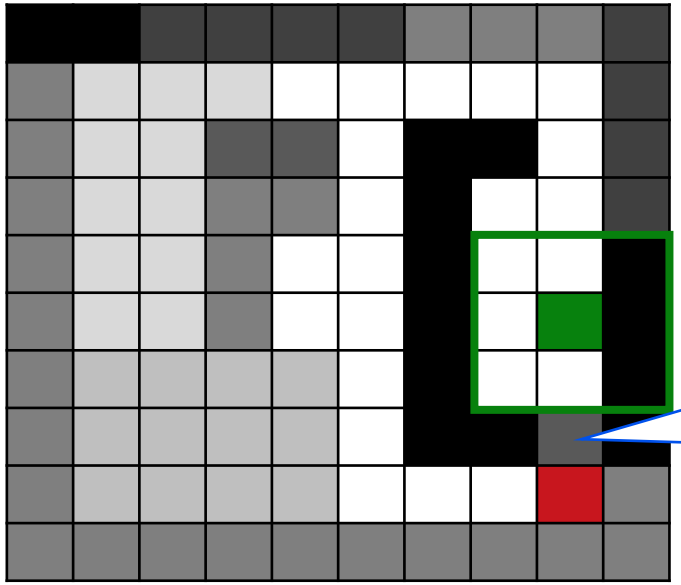
- 快: $O(|A||S|C_{MCTS})$

缺点：

- 不考虑未来观察带来的信息，以及这些信息对决策的助益

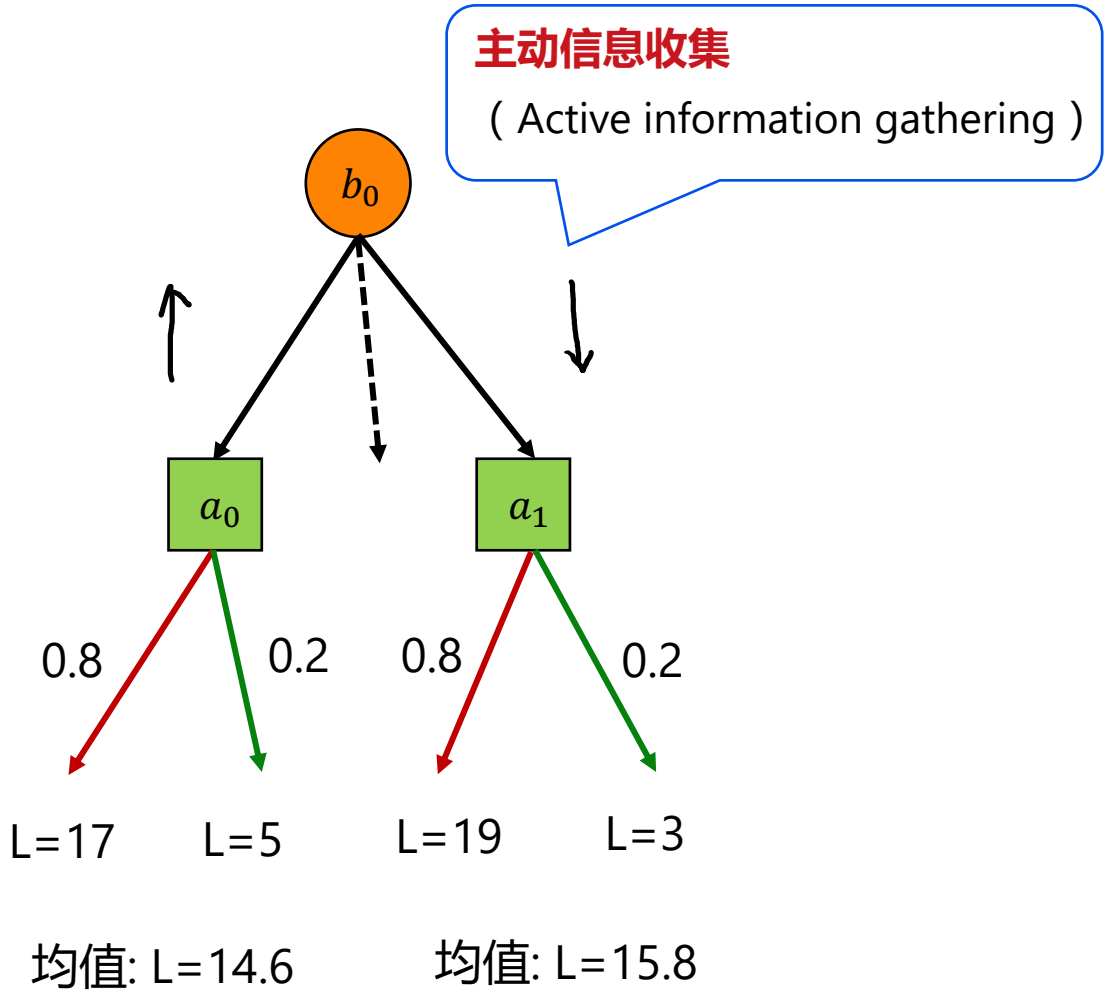
QMDP 算法

缺点：不考虑未来观察带来的信息，以及这些信息对决策的助益



未来信息也具有价值！
（Value of Information）

感知失误率：0.3
奖励：每移动一步 -0.1，不移动 0，到达终点 10，碰撞 -100
状态转移：以概率 1 成功
折扣因子：1.0



基于 MC 采样的 POMDP 规划

POMCP 算法：

- 给定起始信念 b_0, 重复 N 次：
  - **Sample**: 从根节点 b_0 采样 s_0
  - **Simulate**: 从 s_0 开始进行 MC simulation
    - 最优动作选择：$a^* = \underset{a}{\operatorname{argmax}}\, Q(b,a) + \alpha\sqrt{\frac{\log N(b)}{N(b,a)}}$
    - 从 $p(s'|s,a) = T(s,a,s')$ 采样转移到的状态 s_1
    - 从 $p(z|a,s') = O(a,s',z)$ 采样观察 s_1 的结果 z
  - **Expand:** 若观察 z 对应的子节点不存在，创建叶节点 b'
  - **Initialize:** 每扩展一个新节点，进行初始化：
    - $\forall a \in A, Q(b',a) = \mathbf{Rollout}(b',a,\pi_0), N(b',a) = 0$
  - **Backup:** 立即从叶节点开始向上回溯，途径每个节点 b 时计算：
    - 累积奖励：$q = R(b,a) + q'$ (回传自子节点)
    - 平均Q值：$Q(b,a) \leftarrow \frac{N(b,a)}{N(b,a)+1} Q(b,a) + \frac{1}{N(b,a)+1} q$
    - 累积访问次数：$N(b,a) \leftarrow N(b,a) + 1$
- 输出: $\pi^*(b_0) = \underset{a}{\operatorname{argmax}}\, Q(b_0,a)$

可能存在的问题？

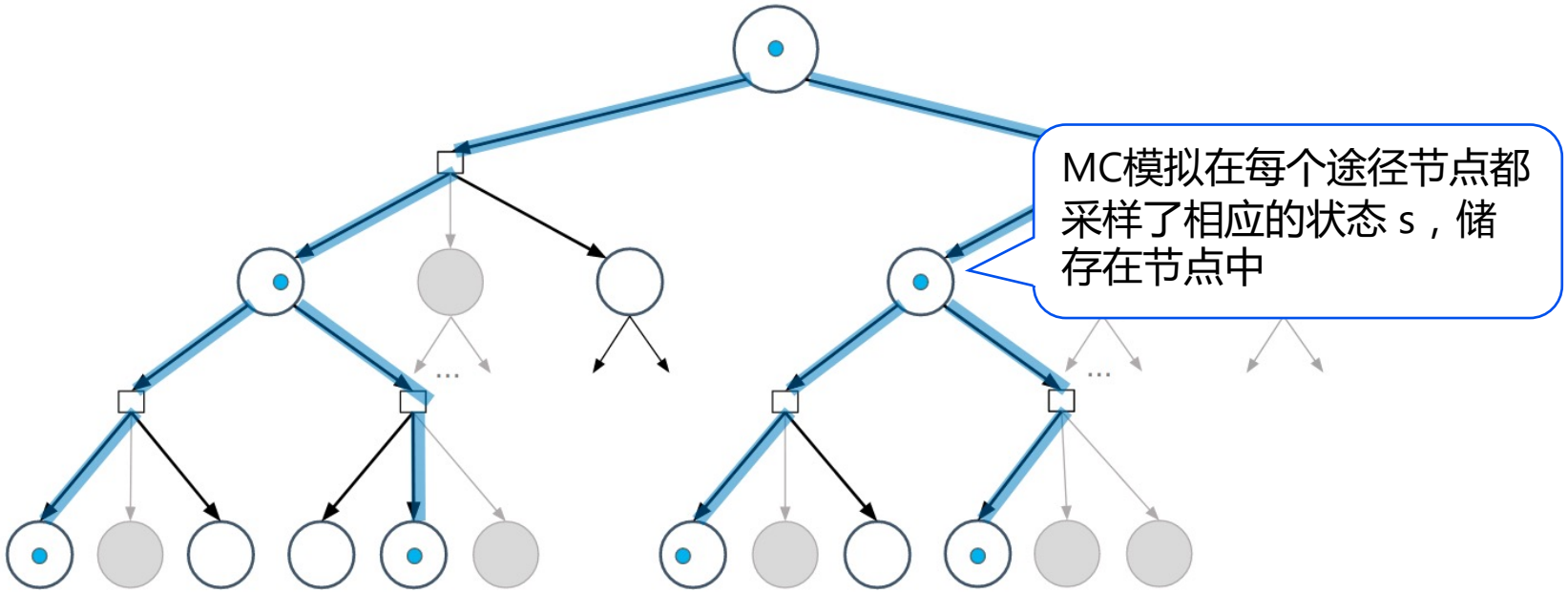
- 当观察空间很大时，很难采样到同一个 z, 因此搜索很难突破第一层

搜索树只考虑采样得到的observation，而不是所有可能性，从而实现快速规划

rollout policy 可以是 $\pi_0(b)$, 也可以是基于历史轨迹的策略 $\pi_0(h_b)$

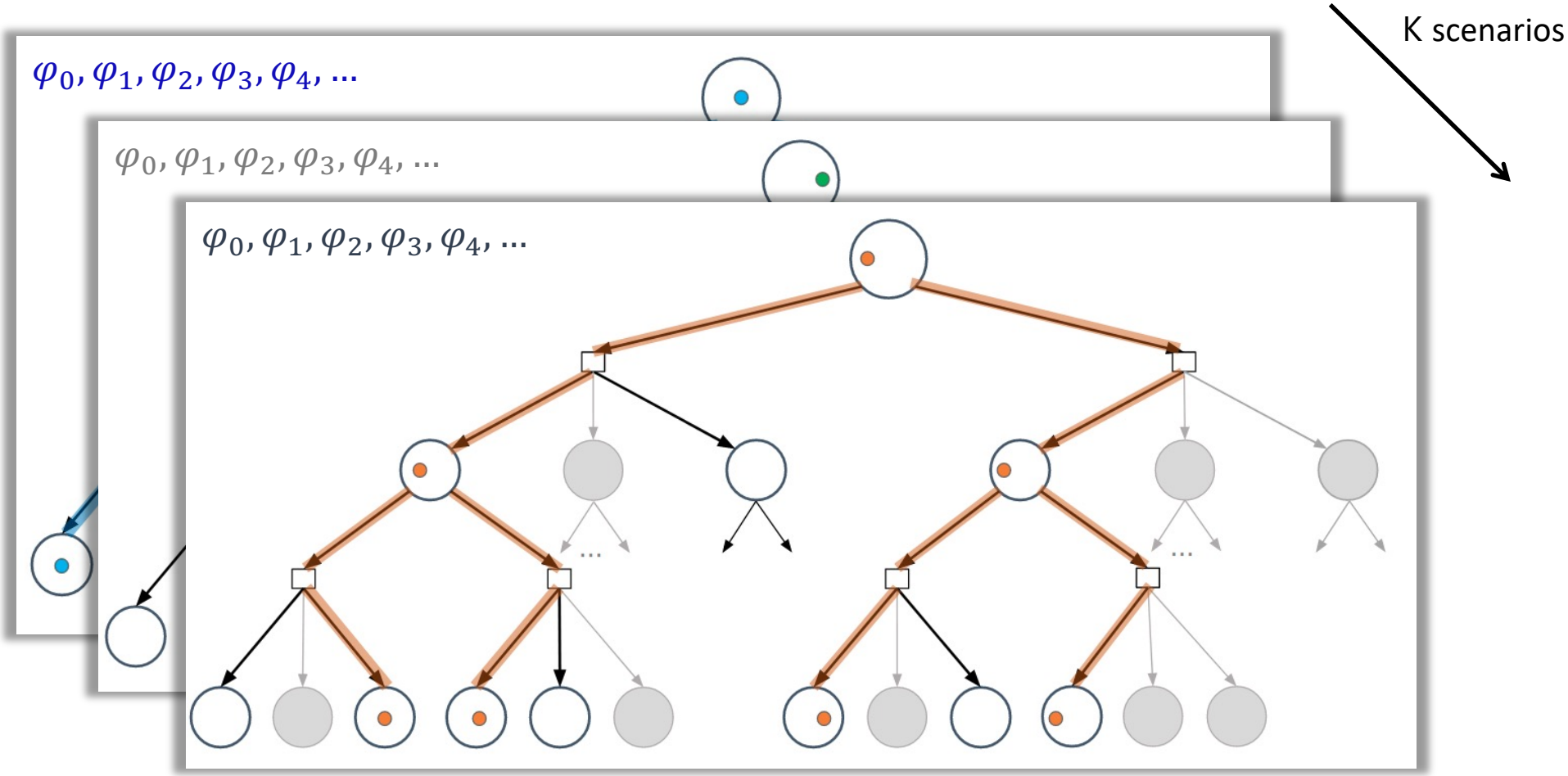
DESPOT 算法

- 用 k 个代表性情形（scenario）构成的稀疏信念树（sparse belief tree）代表完整的信念树
  - **Scenario**: 使用固定的 random seed $\{\varphi_0, \varphi_1, \varphi_2, \varphi_3, \varphi_4, \dots\}$ 进行 MC 模拟



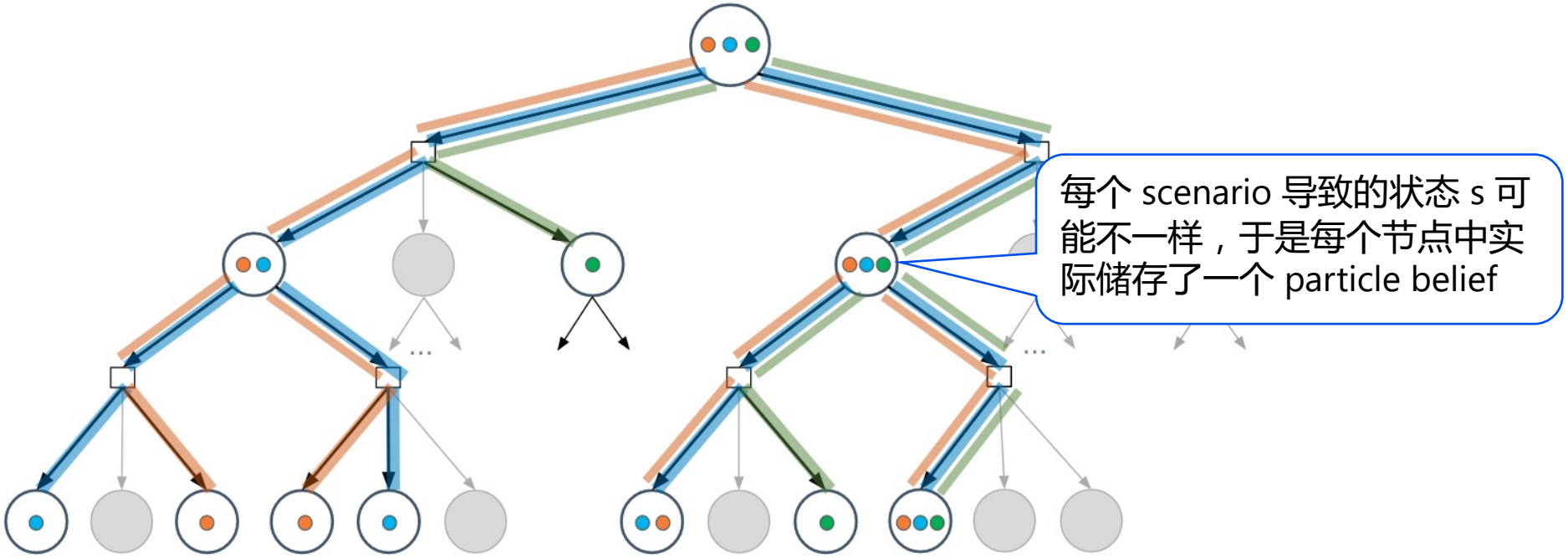
DESPOT

- 用 k 个代表性情形（scenario）构成的稀疏信念树（sparse belief tree）代表完整的信念树



DESPOT

- 由 k 个代表性情形（scenario）构成的稀疏信念树（sparse belief tree）代表完整的信念树



DEterminized Sparse Partially Observable Tree

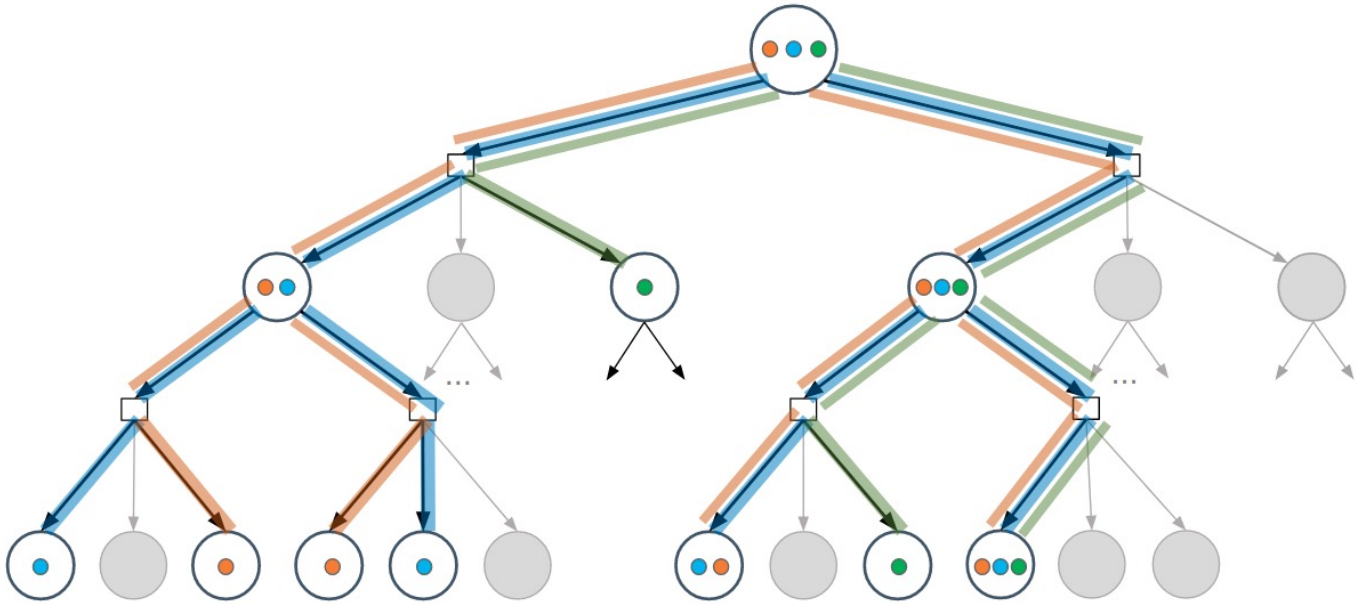
复杂度：$O(|A|^H K)$

启发式 DESPOT 树搜索

- 给每个节点 b 维持一个上界估计 $U(b)$ 与下界估计 $L(b)$，使得： $L(b) \leq V^*(b) \leq U(b)$
- 利用 $U(b)$ 和 $L(b)$ 构造搜索启发，引导搜索：
  - 动作选择：$a^* = \underset{a}{\operatorname{argmax}}\, U(b,a)$
  - 观察选择： $z^* = \underset{z}{\operatorname{argmax}}\ f(|U(b') - L(b')|) = \underset{z}{\operatorname{argmax}}\ WEU(b')$

POMCP：概率上界
DESPOT：启发上界

POMCP：随机采样 z
DESPOT：主动选择最有信息价值的 z 进行优先探索！

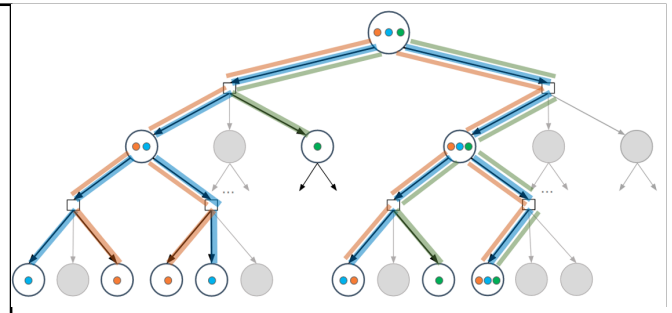


DESPOT 算法

- 给定起始信念 b_0：
  - **Sample Scenario:** 采样 K个初始状态与随机数序列 $\phi_k = \{s_0, \varphi_0, \varphi_1, \varphi_2, \varphi_3, \varphi_4, \dots, \varphi_H\}$
  - **Forward**: 从 b_0 开始向下搜索
    - 最优动作选择：$a^* = \underset{a}{\operatorname{argmax}}\, U(b,a)$（若 b 已经被扩展）
    - 最优观察选择：$z^* = \underset{z}{\operatorname{argmax}}\, WEU(b')$（若 b 已经被扩展）
  - **Expand:** 若 b 未被扩展，根据 b 中的 scenario 集合 Φ_b 进行扩展：
    - 动作分支：对所有 a ，从 $T(s,a,s'|\Phi_b)$ 采样状态转移 {s'}
    - 观察分支：从 $O(a,s',z|\Phi_b)$ 采样观察 $\{z\}$，逐一构造子节点 b′ ，并储存 $\Phi_{b'}$
  - **Initialize:** 每扩展一个新节点，利用启发函数进行初始化：
    - 上界：$U(b') = U_0(b')$　　（ U_0 是启发函数）
    - 下界：$L(b') = \mathbf{Rollout}(b', \pi_0)$（ π_0 是 rollout policy ）
  - **Backup:** 当深入搜索不再有益（WEU<=0）时向上回溯，途径每个节点 b 时计算：

$$V(b) = \max_{a \in A} \left\{ \frac{1}{|\boldsymbol{\Phi}_b|} \sum_{\phi \in \boldsymbol{\Phi}_b} R(s_\phi, a) + \gamma \sum_{z \in Z_{b,a}} \frac{|\boldsymbol{\Phi}_{b'}|}{|\boldsymbol{\Phi}_b|} V(b') \right\}$$

, 其中 V={U, L}



理论性质：

- 随着搜索的进展，所有节点的上下限值都将收敛性于 K 种情形下的最优值
- 随着 K 的增大，K 种情形下的最优值将概率收敛于最优值

=> 概率最优性

POMDP 应用于自动驾驶

1. 构建 POMDP 模型
   - $< S, A, Z, T, O, R, b_0 >$
   - 超参数：H, γ
2. 选择 POMDP 规划算法

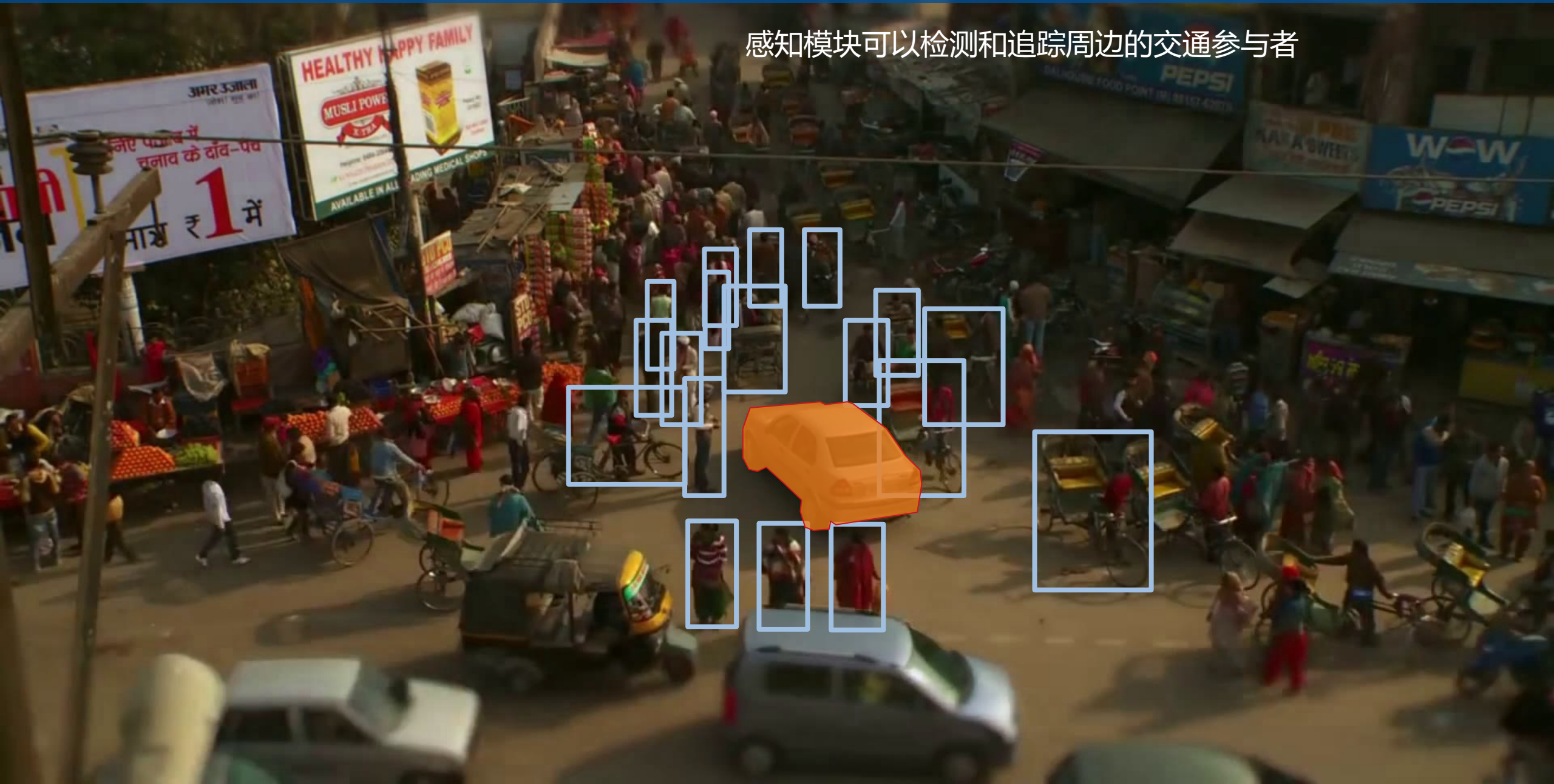

HEALTHY HAPPY FAMILY
MUSLI POWE
AVAILABLE IN ALL
PEPSI
W-W
PEPSI
₹1 में

自车

在避免与他车碰撞的同时，尽快穿过这个混乱的十字路口。

感知

感知模块可以检测和追踪周边的交通参与者



观察

通过处理传感器数据，可以得到自车与他人/车的运动学信息

不确定性

是否能依据历史轨迹准确预测此人未来的运动轨迹？

他人的运动轨迹具有不确定性。

如何建模这些不确定性？

意图（Intention）

若我们知道一个人的意图（比如目的地），那么运动预测将变得容易。我们可以利用运动模型，预测他人的长期交互轨迹。

Luo, Y., Cai, P., Bera, A., Hsu, D., Lee, W. S., & Manocha, D. (2018). PORCA: Modeling and planning for autonomous driving among many pedestrians. IEEE Robotics and Automation Letters, 3(4), 3418-3425.
Luo, Y., Cai, P., Hsu, D., Lee, W.S. (2022). GAMMA: A General Agent Motion Prediction Model for Autonomous Driving. IEEE Robotics and Automation Letters, 7(2), 3499 - 3506.

意图的不确定性

该行人可能有不同的意图，对应地图上的不同目的地。有些可能性高（但并非肯定），有些可能性低（但并非不可能）。

如何获取准确地概率分布？

贝叶斯推断

信念更新公式：

$$b'(s') = \eta O(o|s',a) \sum_{s \in S} T(s'|s,a)\, b(s)$$

历史轨迹　　转移概率：GAMMA

关于个体意图的信念

信念

若对每个个体进行贝叶斯推断，则可以获得关于场景内所有个体意图的信念。

该信念总结了场景内蕴含的不确定性，是自车进行决策规划的基础。

复杂路况自动驾驶 POMDP 模型

Belief b

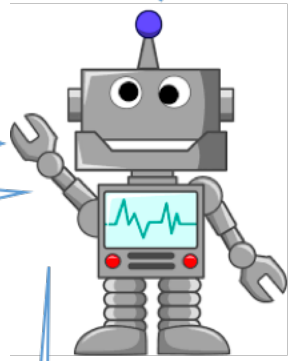
Distribution over states

Transition function T

GAMMA

Observation function O

Gaussian noise



Action a

State s

Pos, vel, headings,

human behavioral states

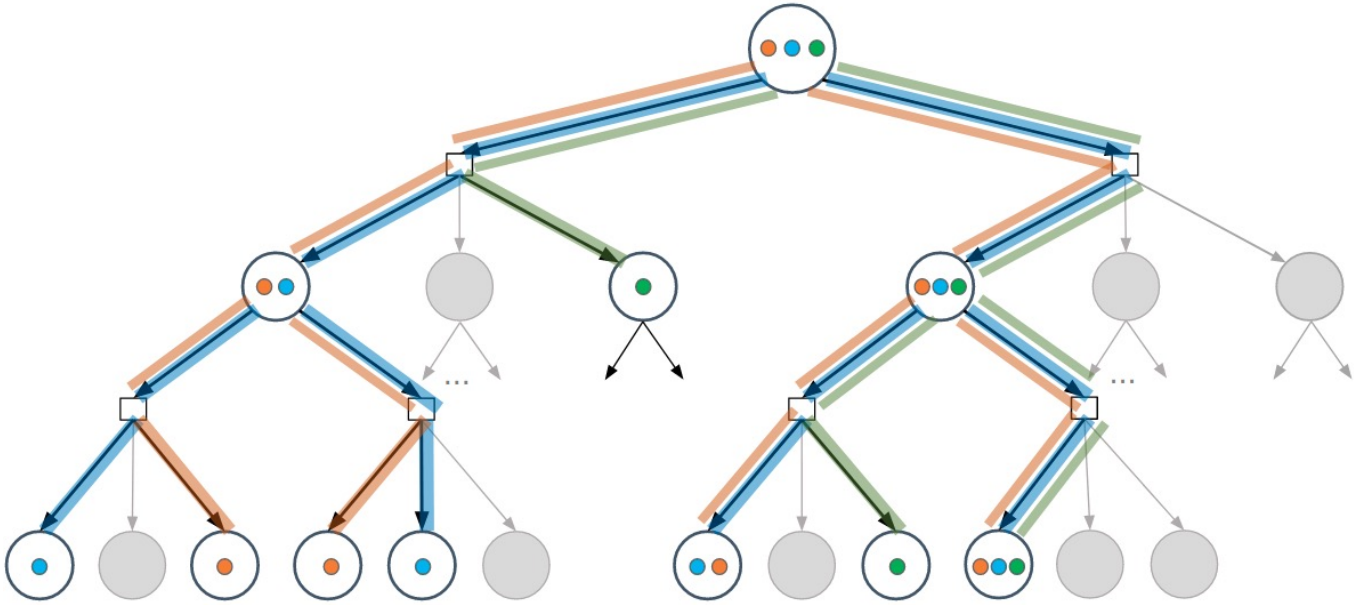
Observation z

Reward r

Safety, efficiency, smoothness

自动驾驶 POMDP 规划

DESPOT[1]: O(|A|^H|Z|^H) -> O(|A|^{H}K)



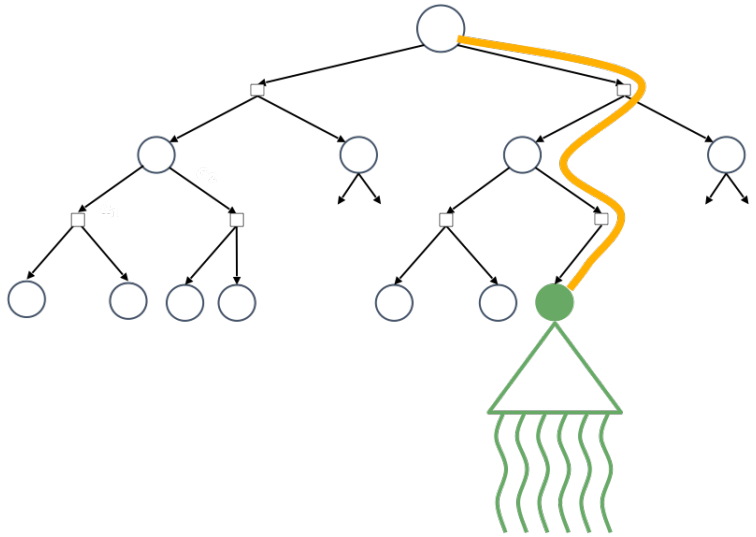
|A| = 9, H = 20, K = 100
DESPOT树大小: O(100*9^{20})

Hybrid Parallel DESPOT

- 利用大规模并行化加速决策规划

任务拆解与重整合

DESPOT



计算复杂度

$O(|A|^H K)$

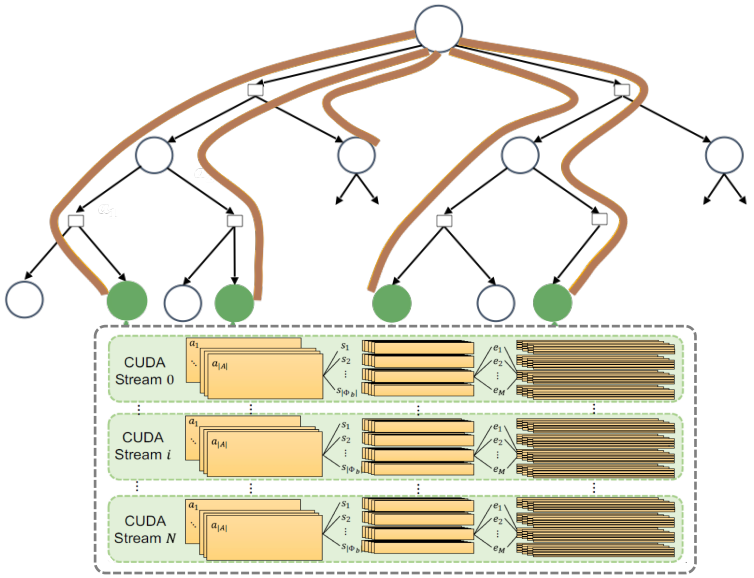
CPU并行
信念树搜索

- 灵活的数据结构
- 频繁的数据共享

GPU并行
蒙特卡洛 Rollout

- 独立的未来情形
- 相似的运算逻辑

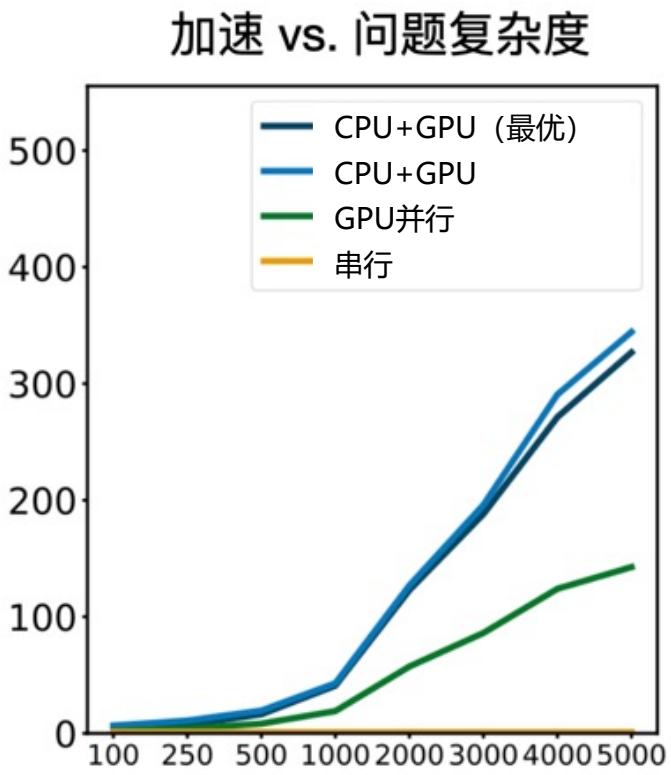
HyP-DESPOT

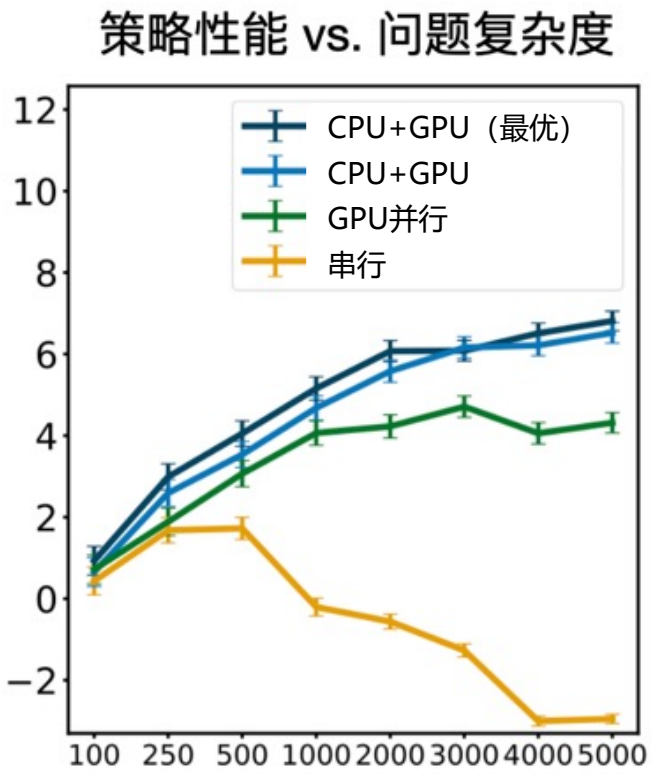


400+倍计算加速

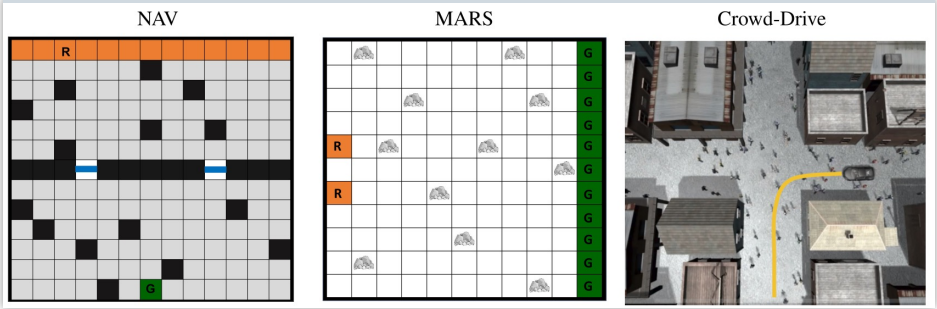
HyP-DESPOT

取得超过400倍算法加速、增强策略性能





仿真中的复杂POMDP问题



人群中的小车自动导航

问题解决？

- 指数复杂度 $O(|A|^{H}K)$ / N:
  - ➔ 只适用于少量动作、短期规划
- 采样不足:
  - ➔ K 个情形对未来可能性的覆盖密度随着 H 指数级下降
- 模型误差累积:
  - ➔ 随着时域 H 的增长，预测变得越来越不准确

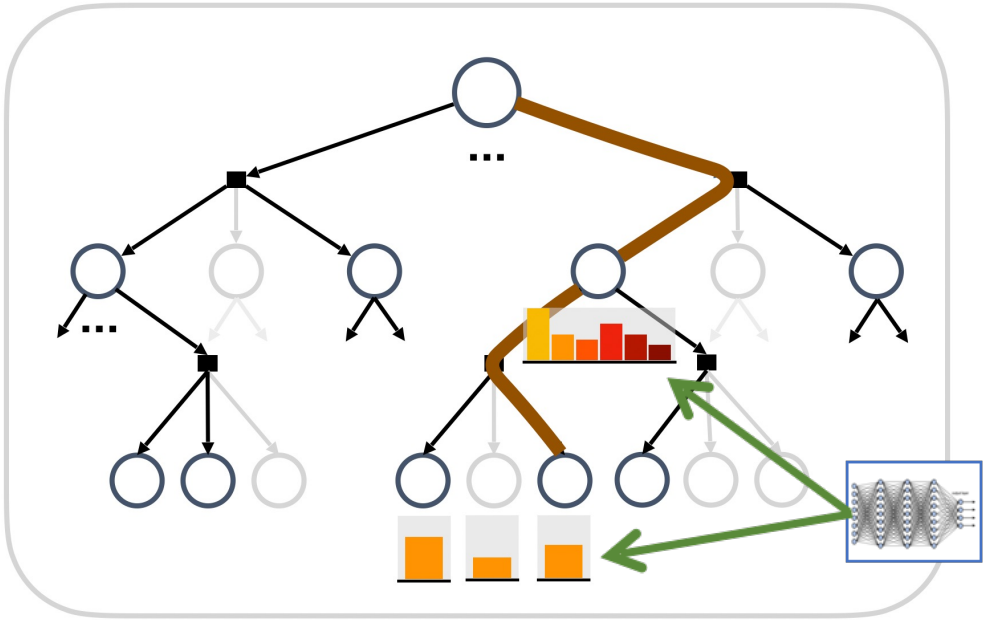
问题关键

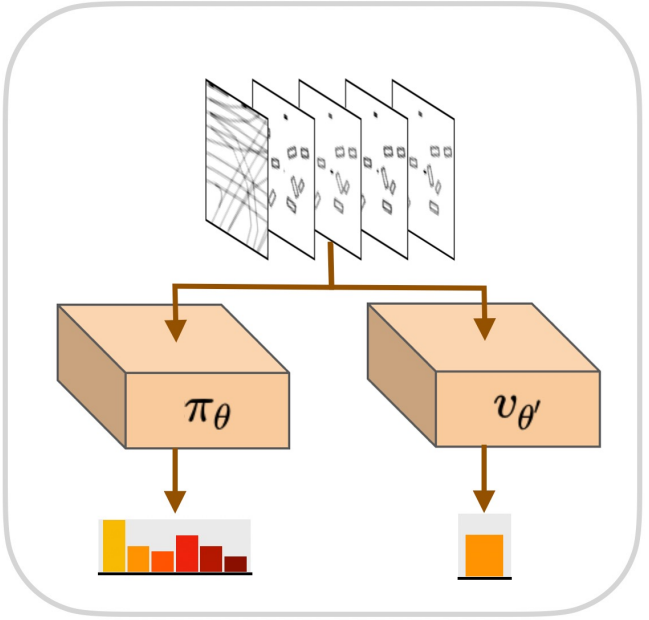
- 指数复杂度 O(|A|^{H}K) / N:
  - ➔ 只适用于少量动作、短期规划
- 采样不足:
  - ➔ K 个情形对未来可能性的覆盖密度随着时域 H 指数级下降
- 模型误差累积:
  - ➔ 随着时域 H 的增长，预测变得越来越不准确

长期规划，但避免深度搜索？

LeTS-Drive: Driving in a Crowd by Learning from Tree Search

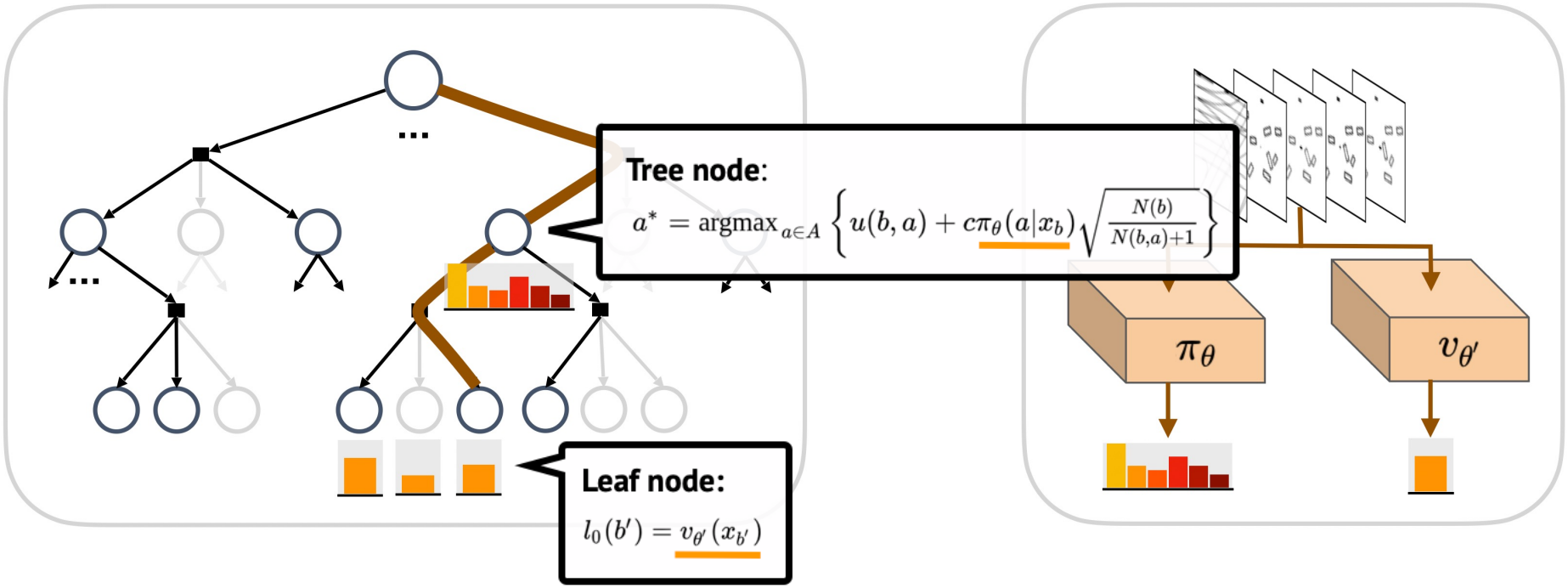
- HyP-DESPOT + 学习搜索启发！





LeTS-Drive: Driving in a Crowd by Learning from Tree Search

- HyP-DESPOT + 学习搜索启发！

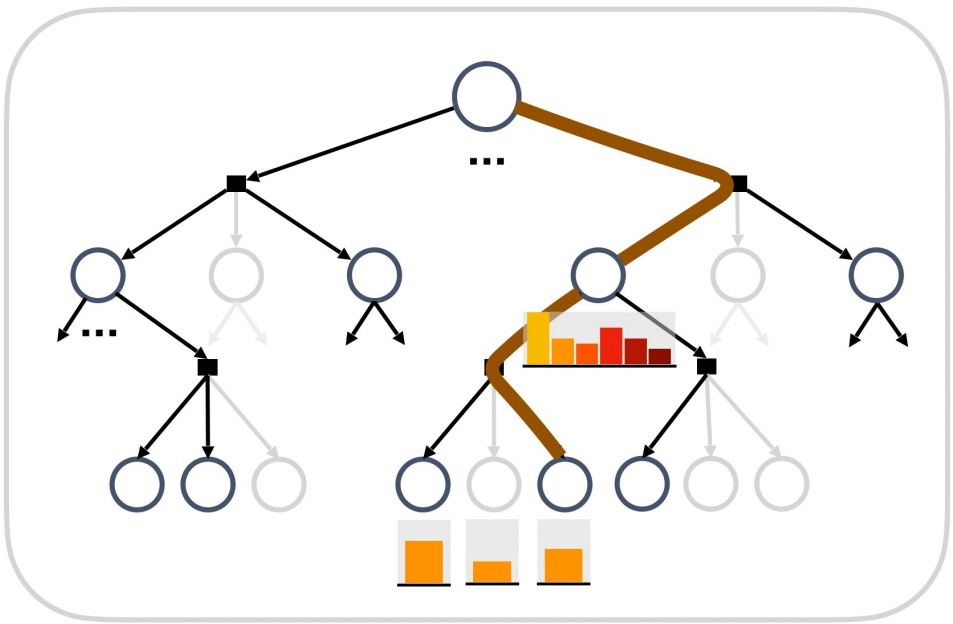


LeTS-Drive: Driving in a Crowd by Learning from Tree Search

- HyP-DESPOT + 学习搜索启发！

规划策略

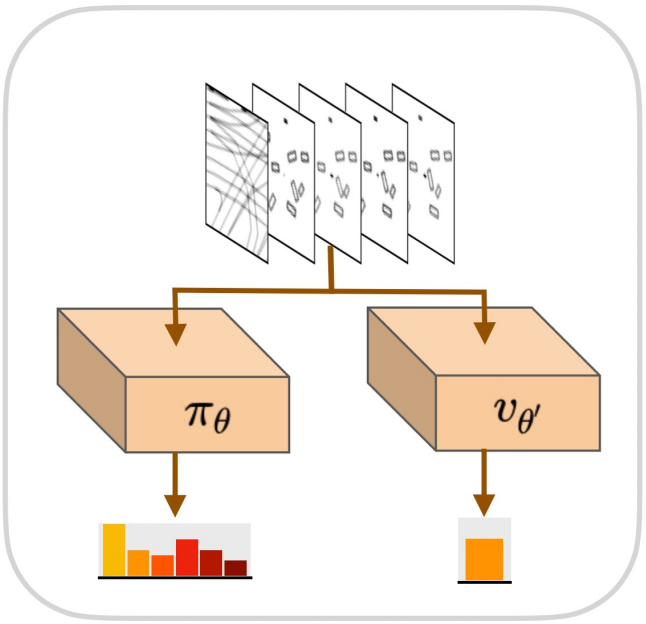
针对实时问题进一步优化！



学习策略

粗略的全局策略

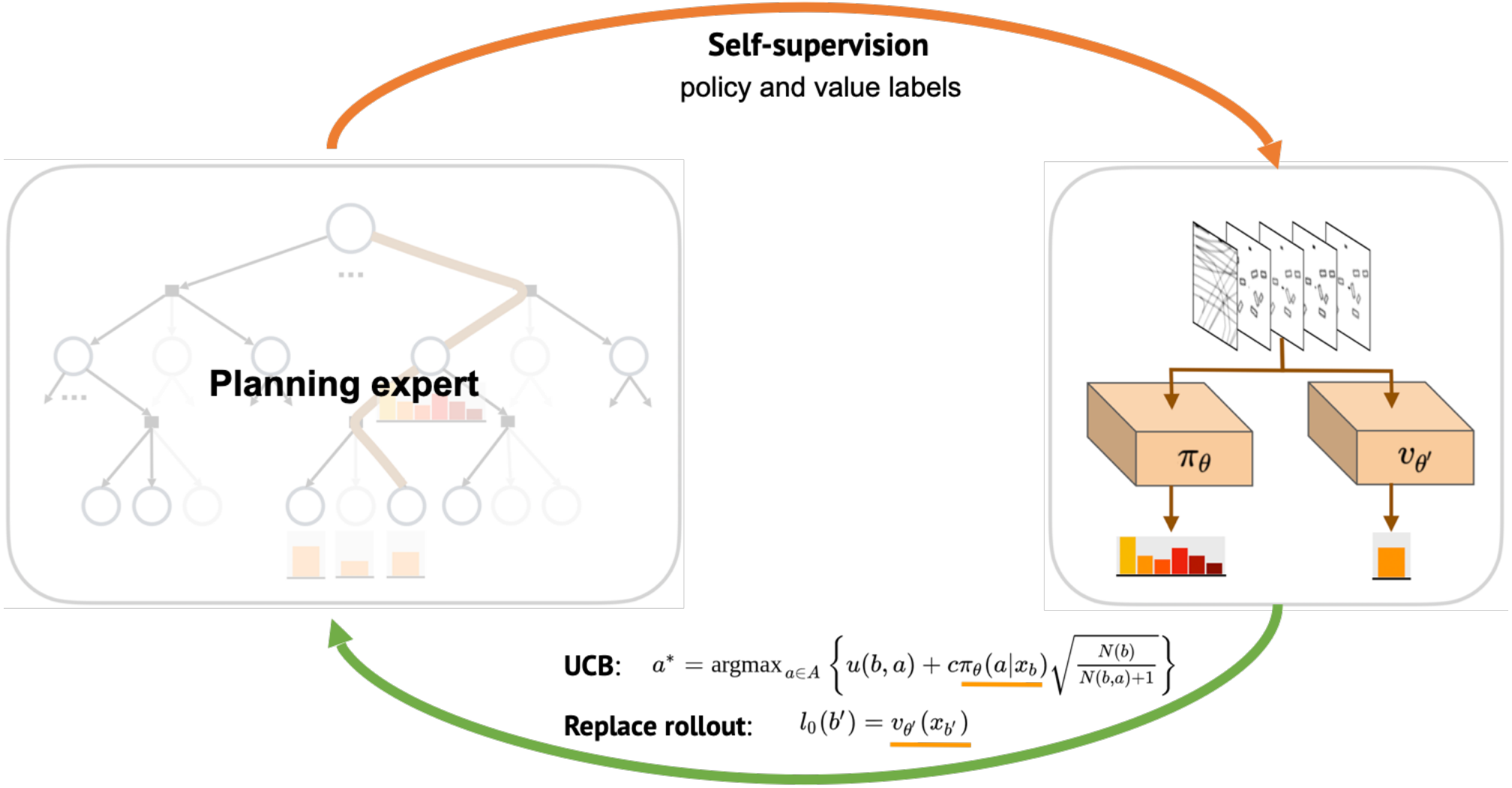
训练数据从哪来？



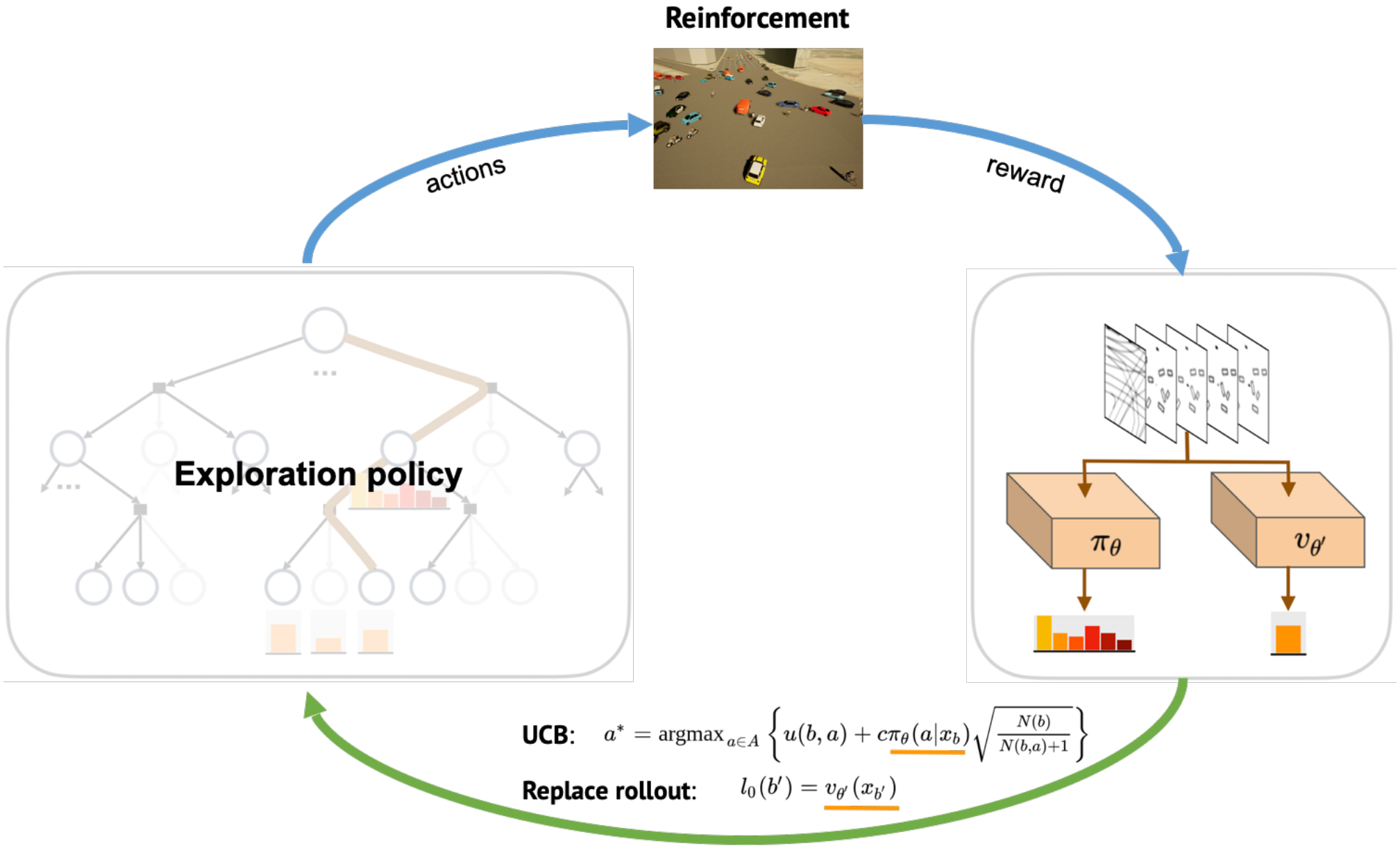
UCB: $a^* = \text{argmax}_{a \in A} \left\{ u(b,a) + c\pi_\theta(a|x_b) \sqrt{\frac{N(b)}{N(b,a)+1}} \right\}$

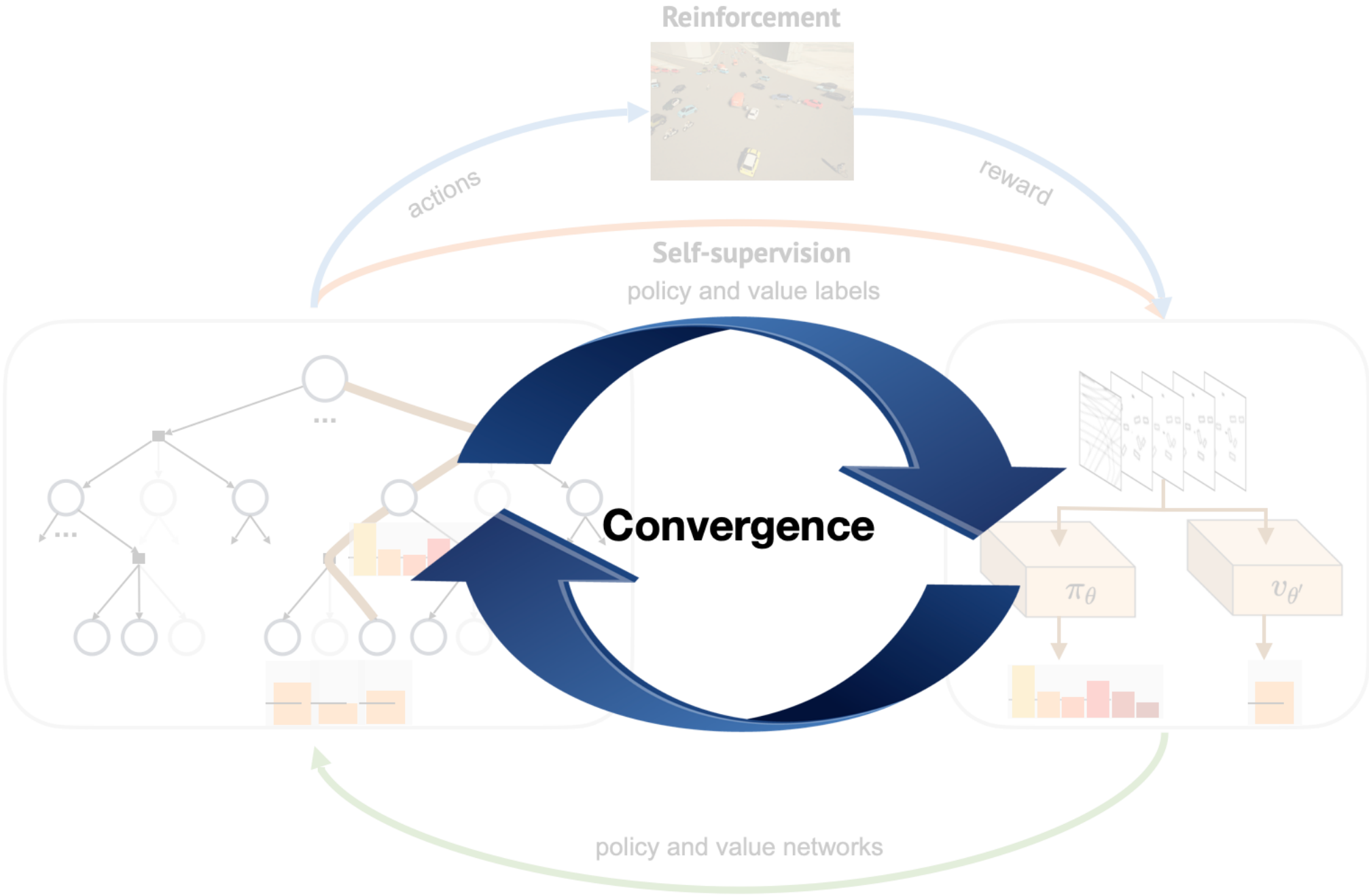
Replace rollout: $l_0(b') = v_{\theta'}(x_{b'})$

LeTS-Drive: Driving in a Crowd by Learning from Tree Search



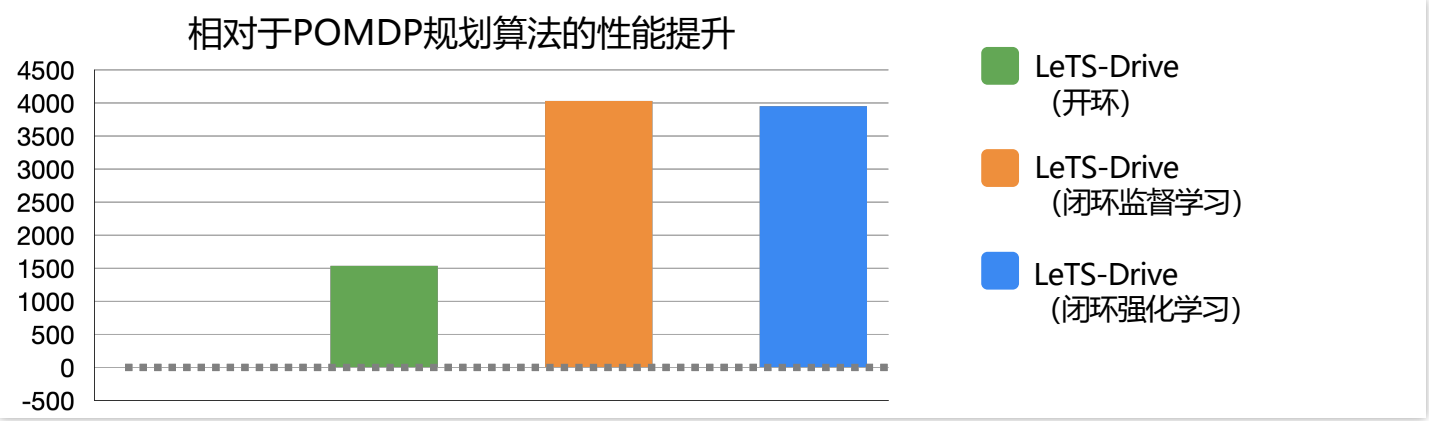
LeTS-Drive: Driving in a Crowd by Learning from Tree Search



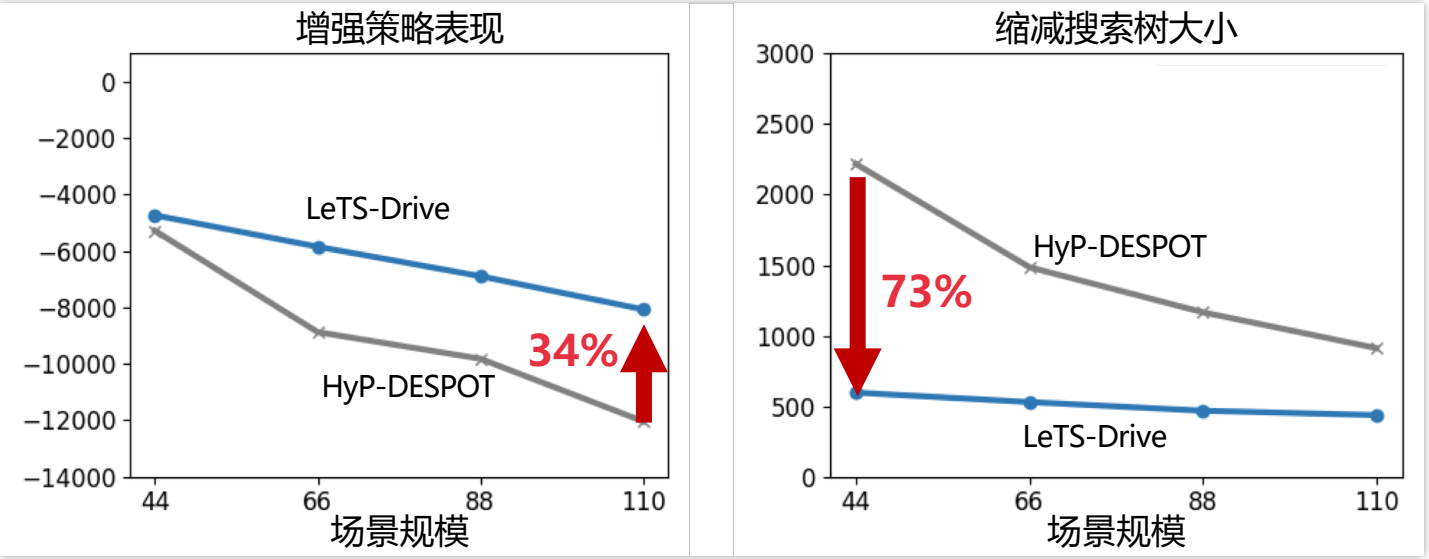

Reinforcement
actions
reward
Self-supervision
policy and value labels
Convergence
π_θ
$v_{\theta'}$
policy and value networks

LeTS-Drive: Driving in a Crowd by Learning from Tree Search

显著提升实时决策质量



提升大规模决策规划的可扩展性



人群中的自动驾驶

无交通规则路口自动驾驶

第十一讲总结

- 部分可观马尔科夫决策过程（POMDP）
  - 数学模型 $< S, A, Z, T, O, R, b_0 >$
  - POMDP Bellman 最优性条件
- POMDP 决策规划算法
  - QMDP
  - POMCP
  - DESPOT
  - HyP-DESPOT
  - 自动驾驶：LeTS-Drive

第十一讲扩展阅读

- **POMDP 基础**： Kochenderfer, Mykel J. Decision making under uncertainty: theory and application. MIT press, 2015.
- **QMDP**： M. L. Littman, A. R. Cassandra, and L. P. Kaelbling. Learning policies for partially observable environments: Scaling up. In International Conference on Machine Learning, pages 362–370, 1995.
- **POMCP**： Silver, David, and Joel Veness. "Monte-Carlo planning in large POMDPs." Advances in neural information processing systems 23 (2010).
- **DESPOT**: Ye, Nan, Adhiraj Somani, David Hsu, and Wee Sun Lee. "DESPOT: Online POMDP planning with regularization." Journal of Artificial Intelligence Research 58 (2017): 231-266.
- **HyP-DESPOT**: Cai, Panpan, Yuanfu Luo, David Hsu, and Wee Sun Lee. "HyP-DESPOT: A hybrid parallel algorithm for online planning under uncertainty." The International Journal of Robotics Research 40, no. 2-3 (2021): 558-573.
- **LeTS-Drive**: Cai, Panpan, and David Hsu. "Closing the planning–learning loop with application to autonomous driving." IEEE Transactions on Robotics 39, no. 2 (2022): 998-1011.

分组论文报告

- 规则：
  - 每三人一组，共同阅读、讨论、讲解一篇学术论文
  - 每组12分钟报告时间（严格计时）+3分钟Q&A
  - 若一人代表报告，成绩按（140%，80%, 80%）分配
  - 若三人合作报告，成绩平均分配
- 时间：
  - 第15周：感知、预测、学习（共11组）
  - 第16周：控制、规划、决策（共11组）
- 请尽快在canvas完成分组与选文！