强化学习

Reinforcement learning

第8节　整合学习与规划

张世周

Outlines

- 1.1 介绍
- 1.2 基于模型的强化学习
- 1.3 集成架构
- 1.4 基于仿真的搜索

基于模型的强化学习

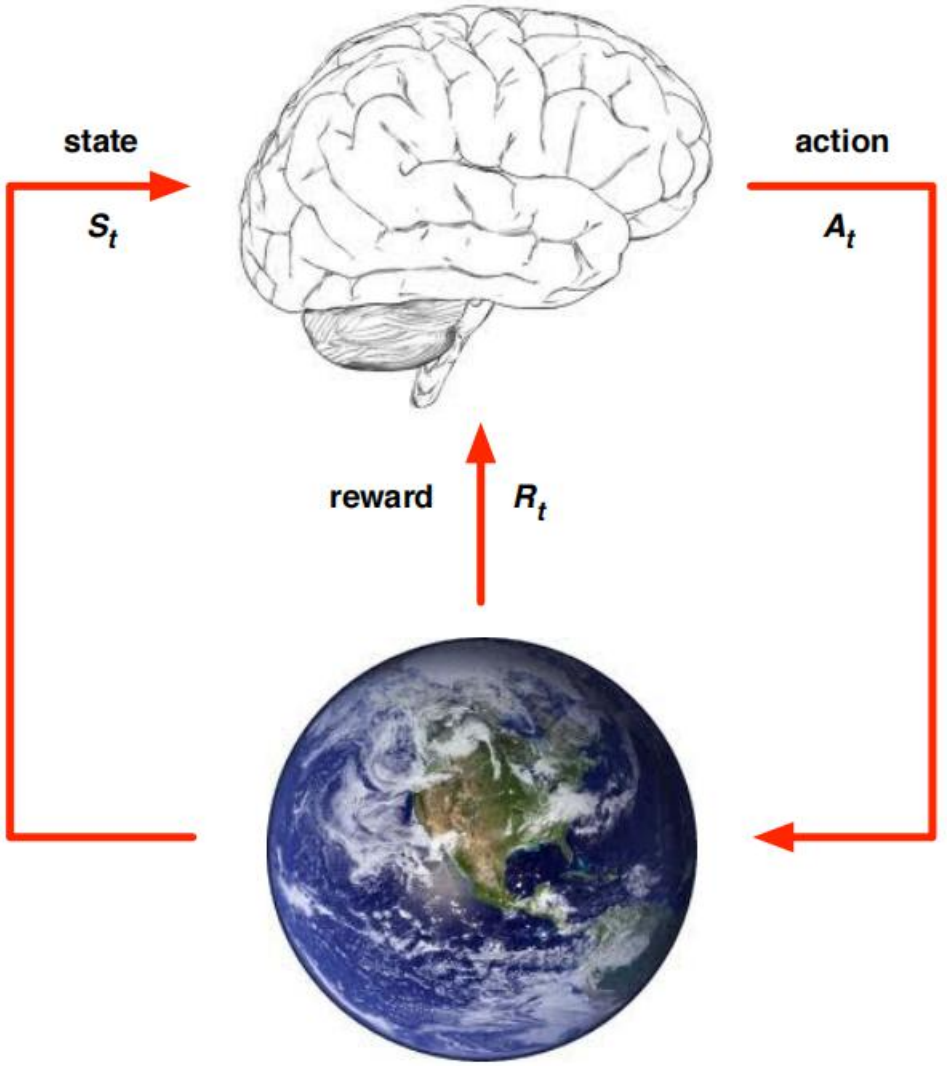
- 上一节课我们讲了如何直接从经验中学习策略，再往前的课我们学习了如何从经验中学习价值函数。

- 这节课我们将学习如何从经验中学习模型并使用规划来构建一个价值函数或策略，将学习和规划集成到一个单一的架构中。

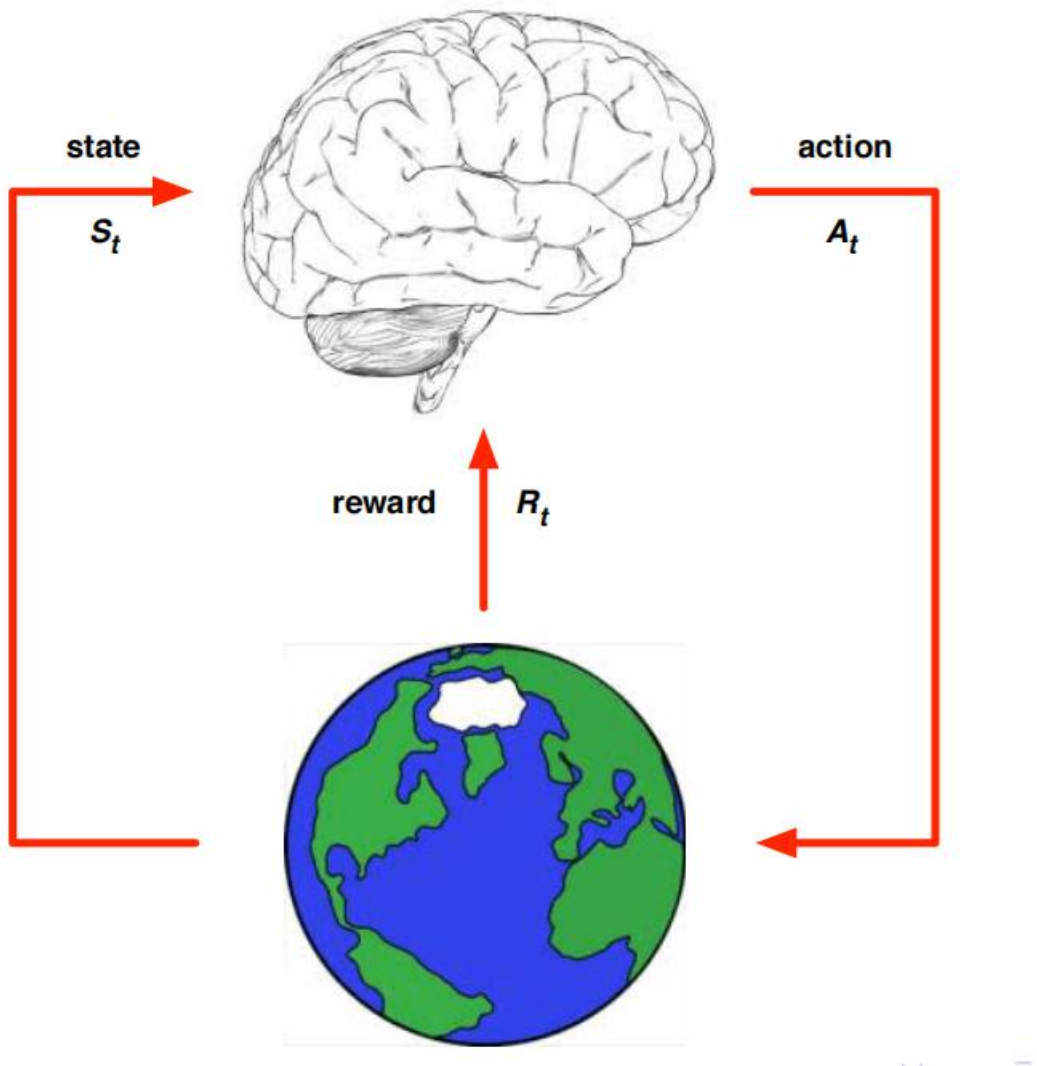
基于模型和无模型的RL的区别

- 无模型的RL：
  - 没有模型
  - 从经验中直接学习价值函数（或策略）
- 基于模型的RL：
  - 从经验中学习模型
  - 从模型中“规划”价值函数（或策略）

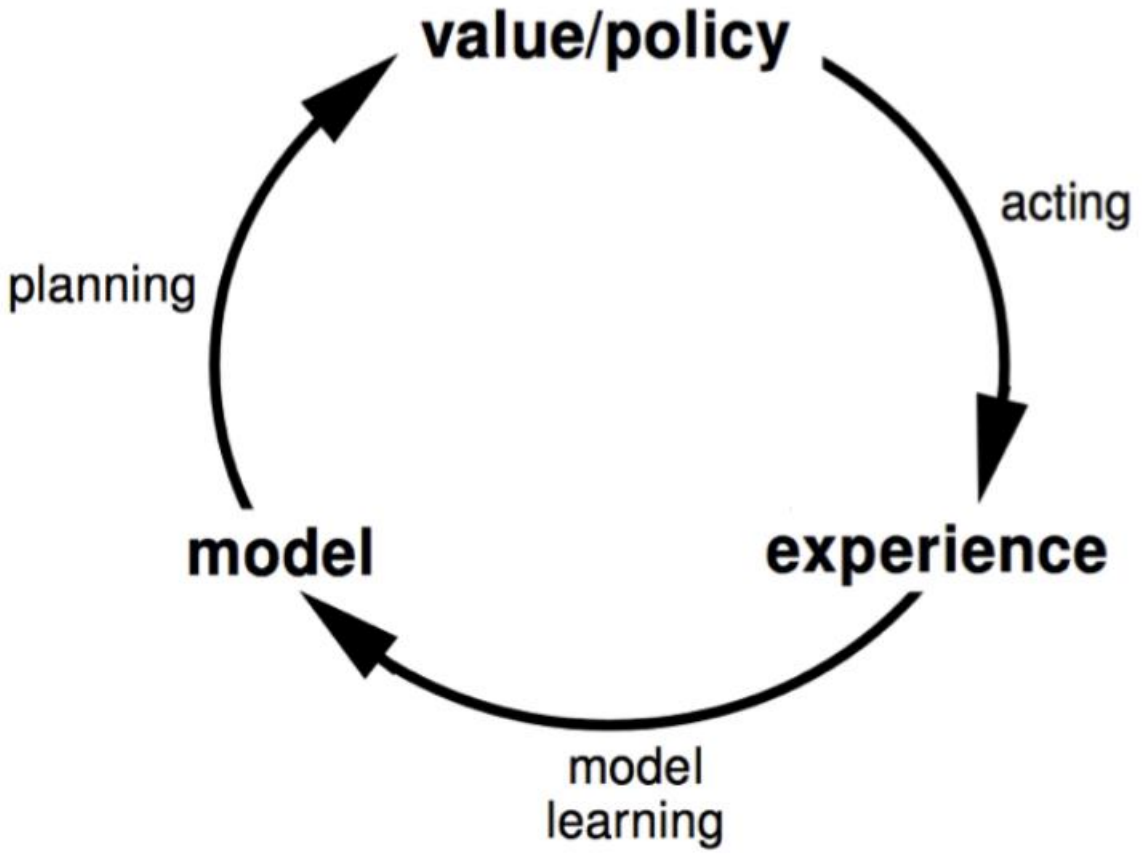
无模型的RL



基于模型的RL



基于模型的RL


value/policy
acting
experience
model learning
model
planning

基于模型的RL的优势

基于模型的RL的优势：

- 当学习价值函数或策略变得很困难的时候，学习模型可能是一条不错的途径，像下棋这类活动，模型是相对直接的，相当于就是游戏规则；
- 可以使用监督式学习方法来学习模型；
- 模型能够对智能体理解环境提供帮助，通过模型，智能体不再局限于如何最大化奖励本身，还能在一定程度上了解采取的动作为什么是好的或者不好，也就是具备一定的推理能力。

基于模型的RL的缺点：

- 模型其实是智能体对环境运行机制的描述，不完全是真实的环境运行机制，因此存在一定程度的近似。另外当使用一个近似的模型去进行价值函数或策略函数的学习时，又会引入一次近似。因此会带来双重的近似误差。

什么是模型

- 模型M是一个 MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}, \mathcal{R} \rangle$ 的参数化 η表现形式。
- 假定：状态空间 S 和行为空间 A 是已知的。
- 可以看出，模型 $\mathcal{M} = \langle \mathcal{P}_\eta, \mathcal{R}_\eta \rangle$ 呈现了状态转移 $\mathcal{P}_\eta \approx \mathcal{P}$ 和奖励 $\mathcal{R}_\eta \approx \mathcal{R}$ ：

$$S_{t+1} \sim \mathcal{P}_\eta(S_{t+1} \mid S_t, A_t)$$
$$R_{t+1} = \mathcal{R}_\eta(R_{t+1} \mid S_t, A_t)$$

- 通常我们需要如下的假设，即状态转移函数和奖励函数是条件独立的：

$$\mathbb{P}\left[S_{t+1}, R_{t+1} \mid S_t, A_t\right] = \mathbb{P}\left[S_{t+1} \mid S_t, A_t\right] \mathbb{P}\left[R_{t+1} \mid S_t, A_t\right]$$

模型学习

- 目标：从经验$\{S_1, A_1, R_2, ..., S_T\}$中估计一个模$\mathcal{M}_\eta$
- 这是一个监督学习问题：

$$S_1, A_1 \rightarrow R_2, S_2$$
$$S_2, A_2 \rightarrow R_3, S_3$$
$$\vdots$$
$$S_{T-1}, A_{T-1} \rightarrow R_T, S_T$$

- 从 s、a 学习 r 的过程是一个回归问题(regression problem)
- 从 s、a 学习 s' 的过程是一个密度估计问题(density estimation problem)
- 选择损失函数：例如MSE，KL divergence
- 通过最小化经验损失寻找参数η

模型示例

根据使用的算法不同，可以有如下多种模型：

- 查表式模型(Table lookup Model)
- 线性期望模型(Linear Expectation Model)
- 线性高斯模型(Linear Gaussian Model)
- 高斯过程模型(Gaussian Process Model)
- 深度置信网络模型(Deep Belief Network Model)

本节课主要以查表模型来解释模型的构建。

查表模型

- 通过经验得到各状态行为转移概率和奖励，把这些数据存入表中，使用时直接检索。状态转移概率和奖励计算方法如下：

$$\hat{\mathcal{P}}^a_{s,s'} = \frac{1}{N(s,a)} \sum_{t=1}^{T} \mathbf{1}(S_t, A_t, S_{t+1} = s, a, s')$$

$$\hat{\mathcal{R}}^a_s = \frac{1}{N(s,a)} \sum_{t=1}^{T} \mathbf{1}(S_t, A_t = s, a) R_t$$

- 在学习的每一步（time-step)，记录如下的状态转换（经验片段）：

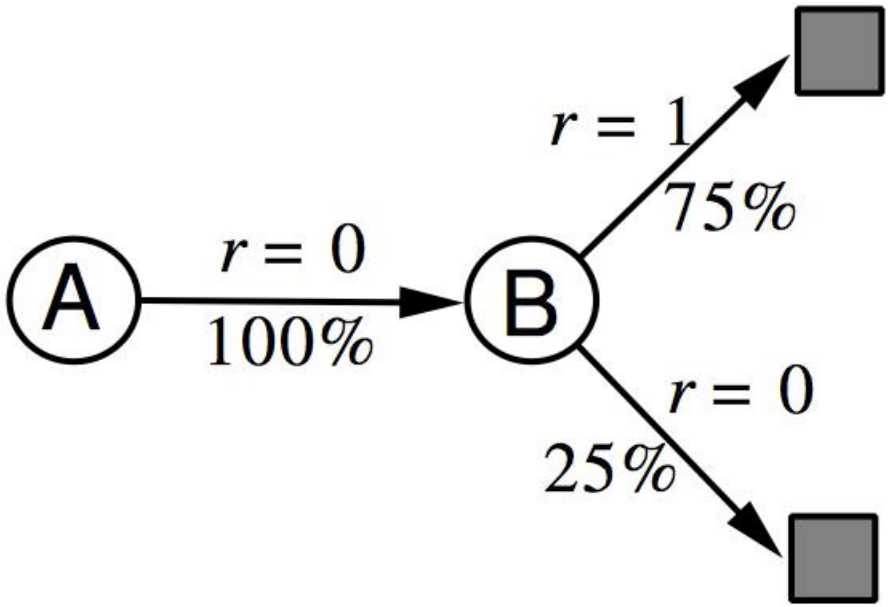
$$\langle S_t, A_t, R_{t+1}, S_{t+1} \rangle$$

- 从模型采样构建虚拟经验时，从tuples中随机选择符合 $\langle s, a, \cdot, \cdot \rangle$ 的一个tuple，提供给智能体进行价值或策略函数的更新。这里的随机选择就体现了状态s的各种后续状态的概率分布。

示例——AB状态

两个状态A，B，无折扣，8个episodes经验

A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0



我们根据经验构建了一个查找表模型

利用模型进行“规划”

理解了模型的学习，“规划”就很简单了，规划的过程相当于求解已知MDP的过程。即给定一个模型 $\mathcal{M}_\eta = \langle \mathcal{P}_\eta, \mathcal{R}_\eta \rangle$ ，求解MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}_\eta, \mathcal{R}_\eta \rangle$ 即找到基于该模型的最优价值函数或最优策略，也就是在给定的状态s下确定最优的行为a。

我们可以使用之前介绍过：

- 价值迭代（Value Iteration）
- 策略迭代方法（Policy Iteration）
- 树搜索方法（Tree Search）
- ...

基于采样的规划（Sample-Based Planning）

- 基于采样的规划是一个简单而强大的方法，在实际规划过程中，模型不是把这些概率值传递给迭代过程，而仅仅是利用模型来产生一个虚拟的状态转换，即一个时间步长的虚拟经验：

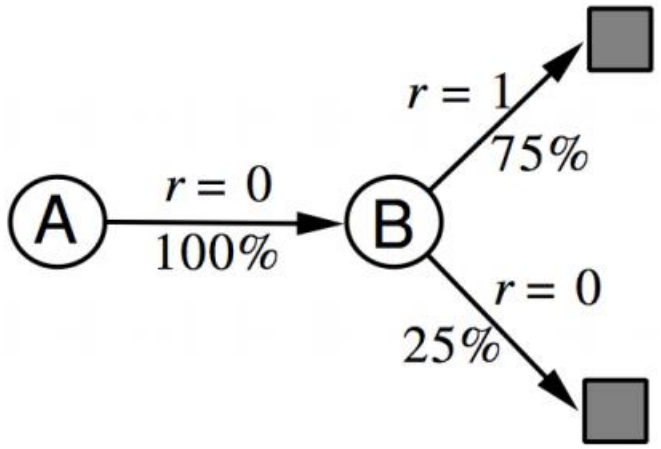
$$S_{t+1} \sim \mathcal{P}_\eta(S_{t+1} \mid S_t, A_t)$$
$$R_{t+1} = \mathcal{R}_\eta(R_{t+1} \mid S_t, A_t)$$

- 有了这些采样所得的虚拟经验，之后可以使用无模型的强化学习方法（MC control、Sarsa、Q-learning）来学习得到价值或策略函数。
- 这种基于采样的规划方法通常更加有效。

示例——AB状态

Real experience

A, 0, B, 0
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 1
B, 0



Sampled experience

B, 1
B, 0
B, 1
A, 0, B, 1
B, 1
A, 0, B, 1
B, 1
B, 0

从真实经历中构建一个查找表模型，将无模型RL应用到采样所得的经验中。使用蒙特卡罗学习方法且衰减系数 $\gamma = 1$ 时，可以得到状态A,B的价值分别为1和0.75。

不精确的模型(Imperfect model)

- 给定一个不完美的模型：$\langle \mathcal{P}_\eta, \mathcal{R}_\eta \rangle \neq \langle \mathcal{P}, \mathcal{R} \rangle$ 。基于模型的强化学习的能力是受到从近似的 MDP $\langle \mathcal{S}, \mathcal{A}, \mathcal{P}_\eta, \mathcal{R}_\eta \rangle$ 中学到的最优策略的限制。
- 基于模型的强化学习最好的结果是受到学习到的模型限制，如果模型本身是不准确的，那么“规划”过程可能会得到一个次优的策略。

- 解决方法1：当模型是错误的，使用无模型的RL
- 解决方法2：显式推理模型的不确定性。

实际经验和模拟经验

有两种经验来源：

- **真实经验：**直接从环境中采样（真实MDP）

$$S' \sim \mathcal{P}^a_{s,s'}$$
$$R = \mathcal{R}^a_s$$

- **模拟经验：**从模型中采样（近似MDP）

$$S' \sim \mathcal{P}_\eta(S' \mid S, A)$$
$$R = \mathcal{R}_\eta(R \mid S, A)$$

整合学习和规划（Integrating Learning and Planning）

无模型的RL：

- 没有模型
- 从真实经验中直接学习价值函数（或策略）

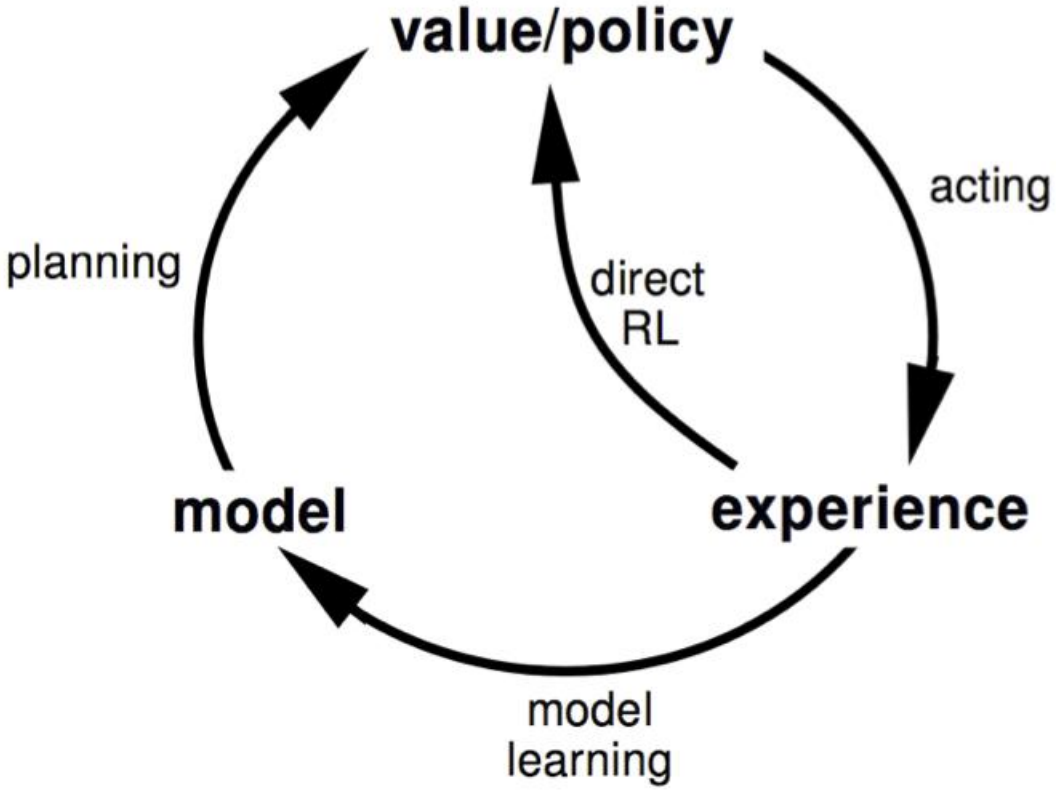
基于模型的RL（使用基于采样的规划）：

- 从真实经验中学习模型
- 从模拟经验中“规划”价值函数（或策略）

Dyna：

- 从真实经验中学习模型
- 从真实和模拟经验中学习和规划价值函数（或策略）

Dyna 结构


value/policy
acting
direct
RL
planning
model
experience
model
learning

Dyna-Q算法

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

Do forever:

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

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

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

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

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

(f) Repeat n times:

$S \leftarrow$ random previously observed state

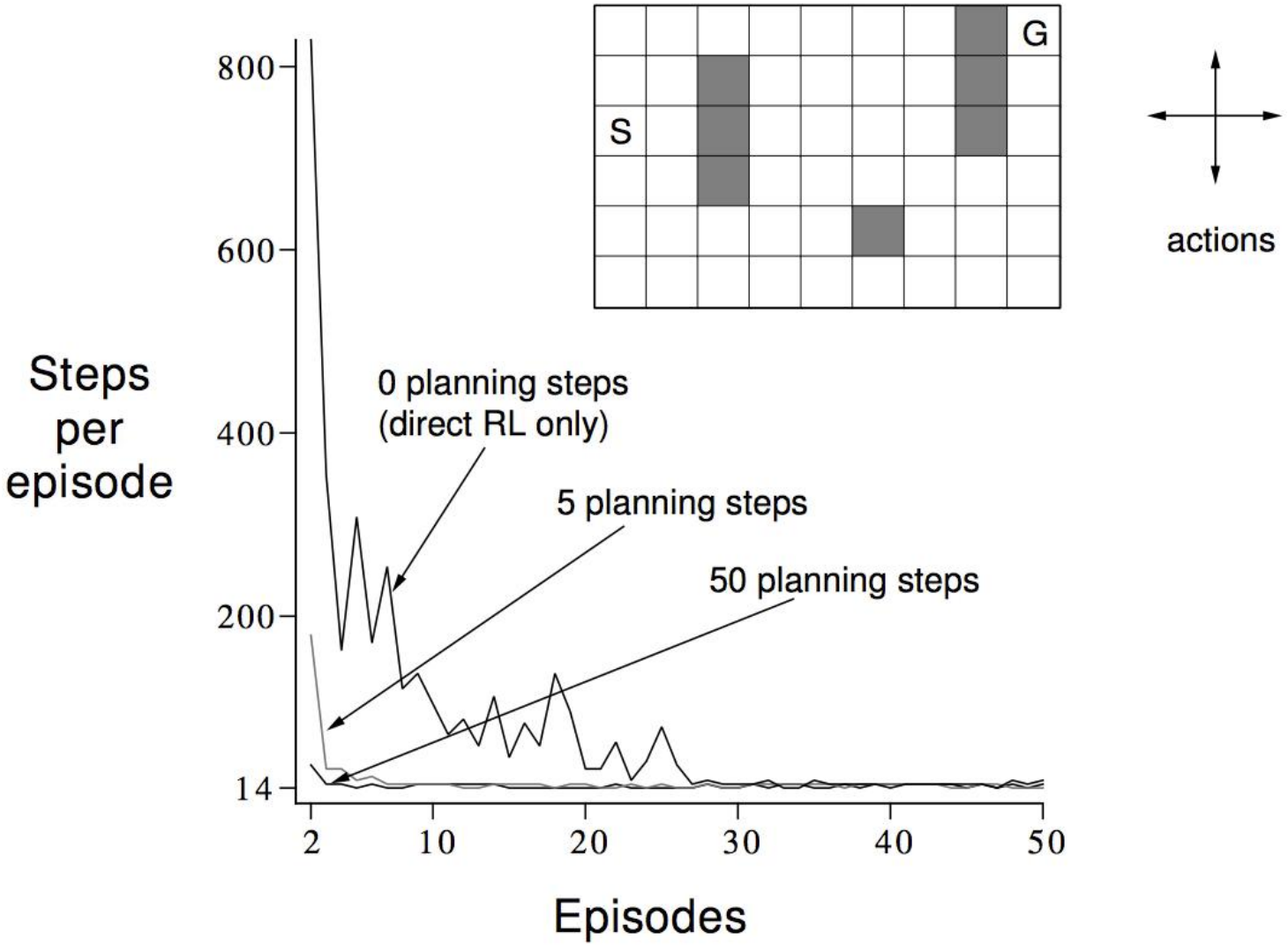
$A \leftarrow$ random action previously taken in S

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

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

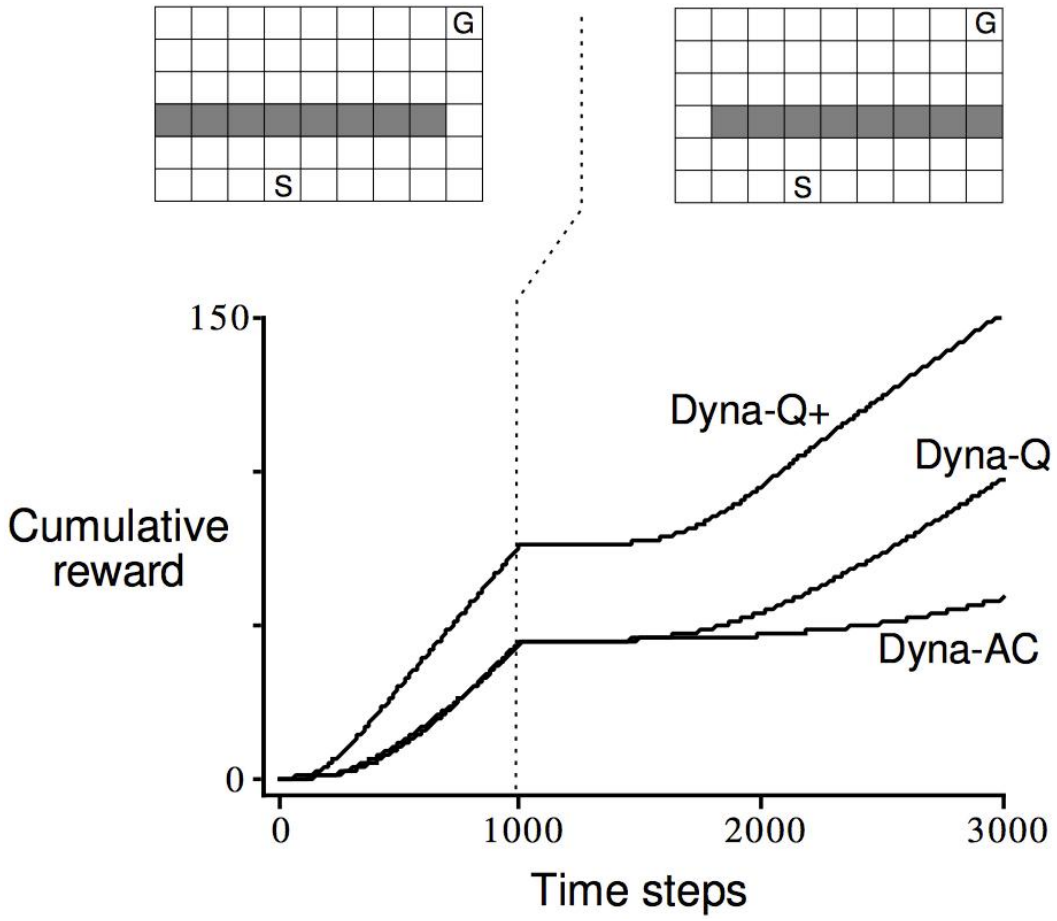
(d)Q-learning; (e) model learning; (f) planning

Dyna-Q 在一个简单迷宫中的应用



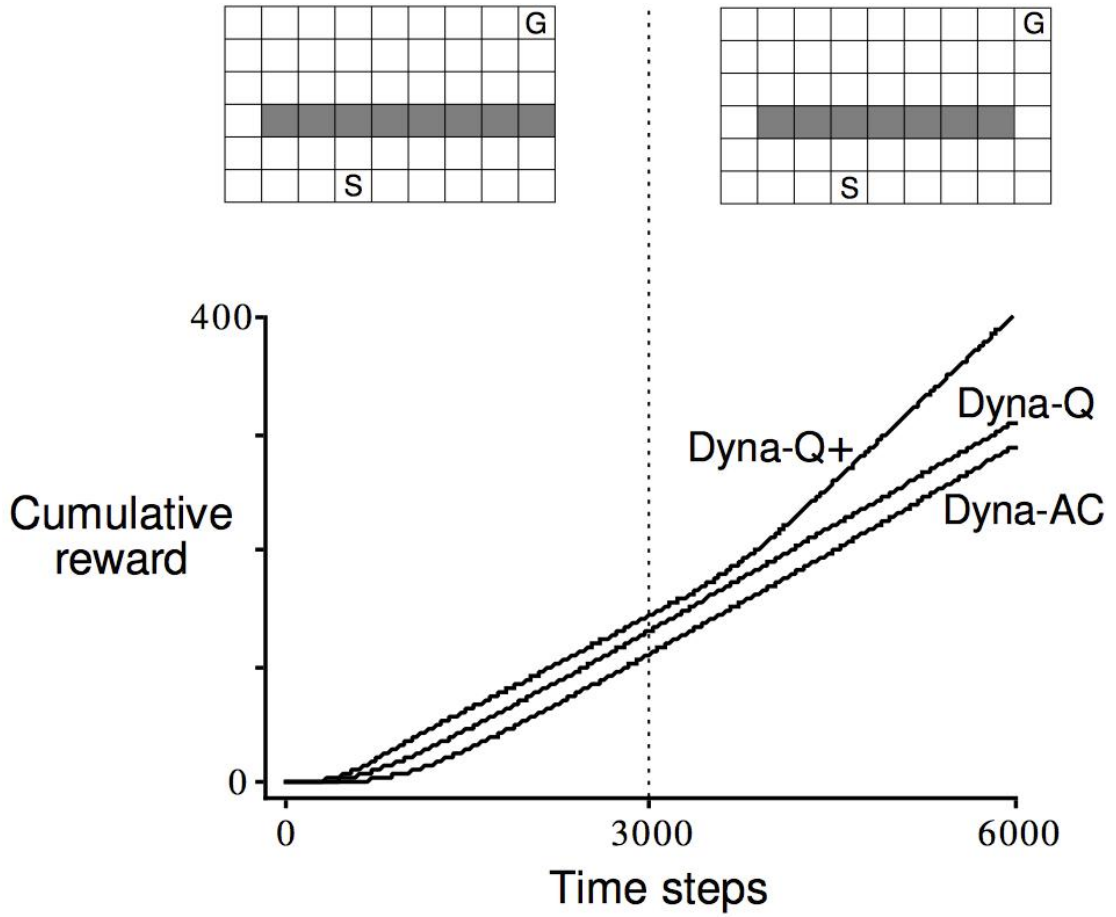
Dyna-Q 和一个不准确的模型

下面的例子模拟了某时刻（虚线所示）环境发生了改变，变得更加困难



Dyna-Q 和一个不准确的模型（2）

下面的例子模拟了某时刻（虚线所示）环境发生了改变，变得更加简单

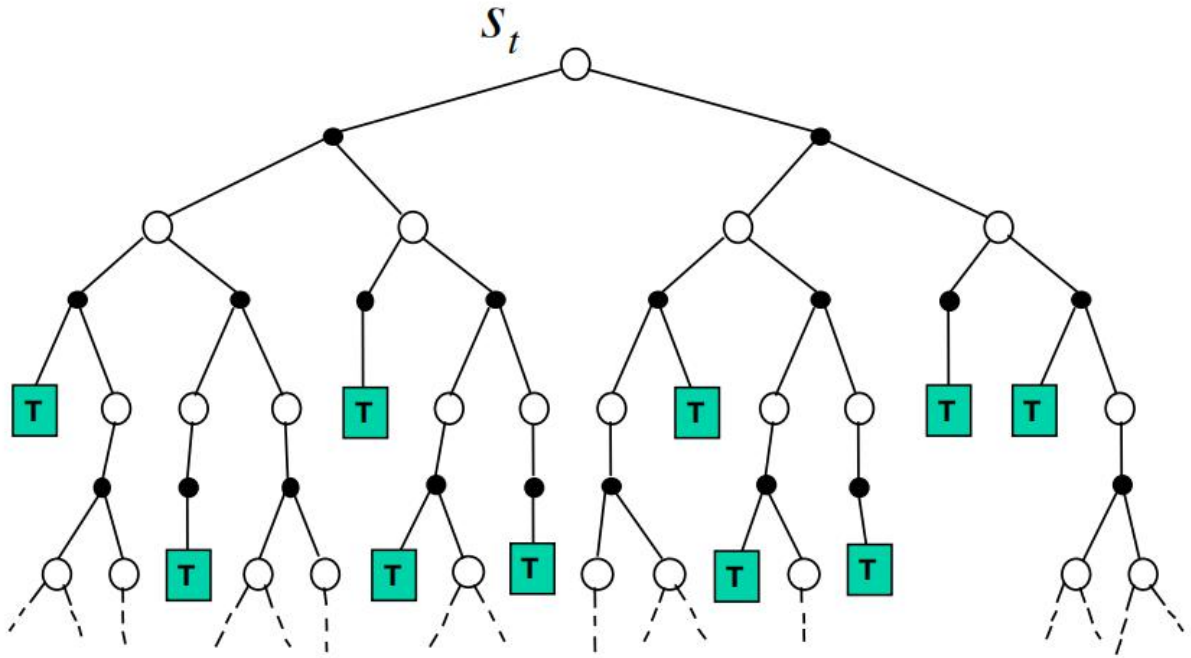


Outlines

- 1.1 介绍
- 1.2 基于模型的强化学习
- 1.3 集成架构
- 1.4 基于模拟的搜索（simulation-based search）

前向搜索

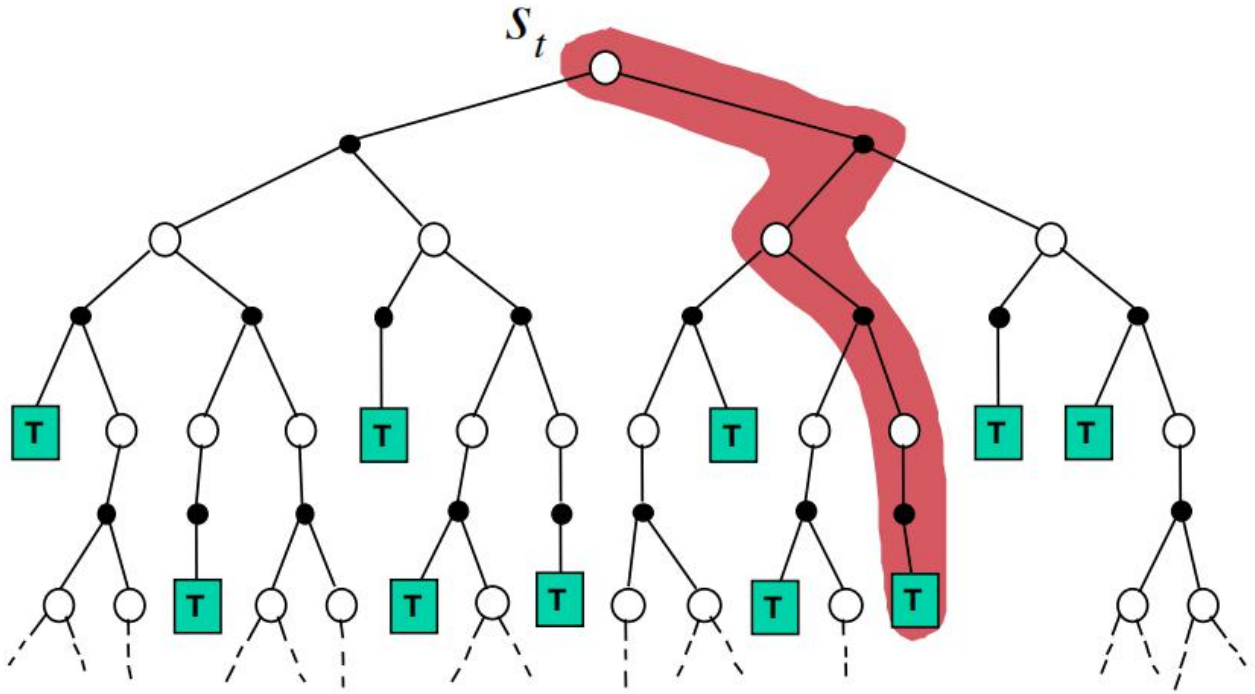
- 前向搜索(Forward Search)算法通过往前看来采取最好的行为。这种算法把当前状态St作为根节点构建了一个搜索树（Search Tree），使用MDP模型进行前向搜索。



- 前向搜索不需要解决整个MDP，而仅仅需要构建一个从当前状态开始与眼前的未来相关的次级（子）MDP。

基于模拟的搜索(Simulation-Based Search)

- 基于模拟的搜索(Simulation-Based Search) 是前向搜索的一种形式，它从当前时刻开始，使用基于模拟采样的规划，构建一个关注于短期未来的前向搜索树，把这个搜索树作为一个学习资源，然后使用Model Free的强化学习来寻找最优策略。



基于模拟的搜索(Simulation-Based Search) （2）

- 从当前时刻t开始，从模型或者从实际经历中进行采样，形成从当前状态开始到终止状态 S_T 的一系列Episode：

$$\{s_t^k, A_t^k, R_{t+1}^k, \cdots, S_T^k\}_{k=1}^K \sim \mathcal{M}_\nu$$

- 有了这些Episode资源，我们可以使用Model-Free的强化学习方法，如果我们使用蒙特卡罗控制，那么这个算法可以称作蒙特卡罗搜索(Monte-Carlo Search)，如果使用Sarsa方法，则称为TD搜索(TD Search)。

简单蒙特卡罗搜索 Simple Monte-Carlo Search

1. 给定一个模型 $\mathcal{M}_\nu$ 和一个模拟的策略 π

2. 针对行为空间里的每一个行为 $a \in \mathcal{A}$

   a) 从这个当前（实际）状态 s_t 开始模拟出K个Episodes:

$$\{s_t, a, R^k_{t+1}, S^k_{t+1}, A^k_{t+1}, ..., S^k_T\}^K_{k=1} \sim \mathcal{M}_\nu, \pi$$

   b) 使用平均收获（蒙特卡罗评估）来评估当前行为a的行为价值 $Q(s_t, a)$

$$Q(s_t, a) = \frac{1}{K} \sum_{k=1}^{K} G_t \xrightarrow{P} q_\pi(s_t, a)$$

3.选择一个有最大$Q(s_t, a)$ 的行为作为实际要采取的行为 a_t ：

$$a_t = \underset{a \in \mathcal{A}}{\operatorname{argmax}}\ Q(s_t, a)$$

蒙特卡罗树搜索 Monte-Carlo Tree Search

每一次模拟结束，策略将得到改进

每次模拟包括两个阶段（in-tree, out-of-tree）

- 树内确定性策略（Tree Policy）：选择动作以最大限度提高Q(s,a)
- 树外默认策略（Default Policy）：随机选择动作

重复 (每次模拟):

- 使用蒙特卡罗评估评估状态Q(s,a)
- 同时基于 $\epsilon - \text{greedy}(Q)$ 的搜索策略使得搜索树将不断的扩展，策略也将不断得到改善。

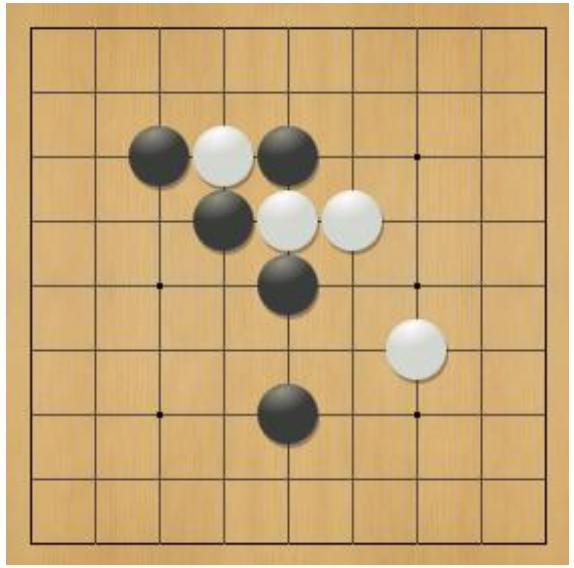
上述方法就相当在某一个状态s_t时，针对模拟的经历使用蒙特卡罗控制来寻找以当前状态 S_t 为根状态的最优策略。这种方法最终将收敛到最优策略。

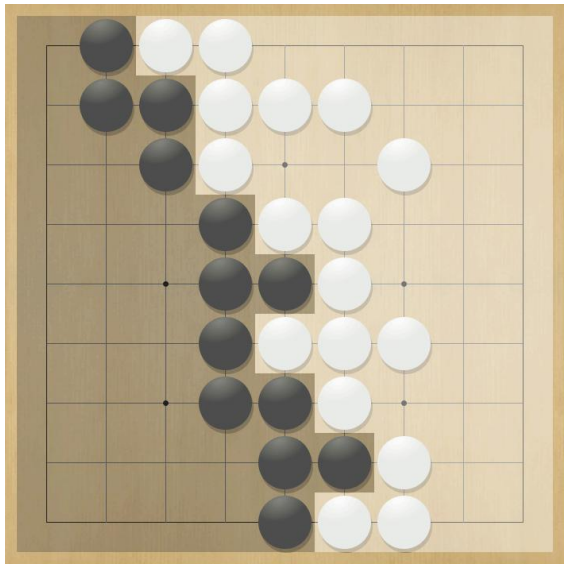
案例研究：围棋

围棋是诞生于2500年前的古代东方的一种益智游戏，它被认为是经典棋盘类游戏中最难的游戏，同时对于计算机领域也是AI领域的一个巨大挑战，传统的暴力搜索算法无法战胜人类高级棋手一度使得人们相信围棋更多地要依靠人的直觉和灵感。

围棋的规则

围棋游戏通常在一个19*19、13*13或9*9的方格棋盘中，有黑白两种颜色的棋子，交手双方交替落棋，棋只能落在棋盘中线线交点上。被一方完全围住的棋需要从棋盘中移除。计算胜负时看哪一方棋子围住的区域大。下图展示的是一个9*9棋盘的早期（左）和最终结局（右，白棋获胜）。





围棋的位置评估

获得收益的规则：

$$R_t = 0 \text{ for all non-terminal steps } t < T$$

$$R_T = \begin{cases} 1 & \text{if Black wins} \\ 0 & \text{if White wins} \end{cases}$$

这是一个双方博弈的过程，策略可以被看成是双方策略的联合：$\pi = \langle \pi_B, \pi_W \rangle$

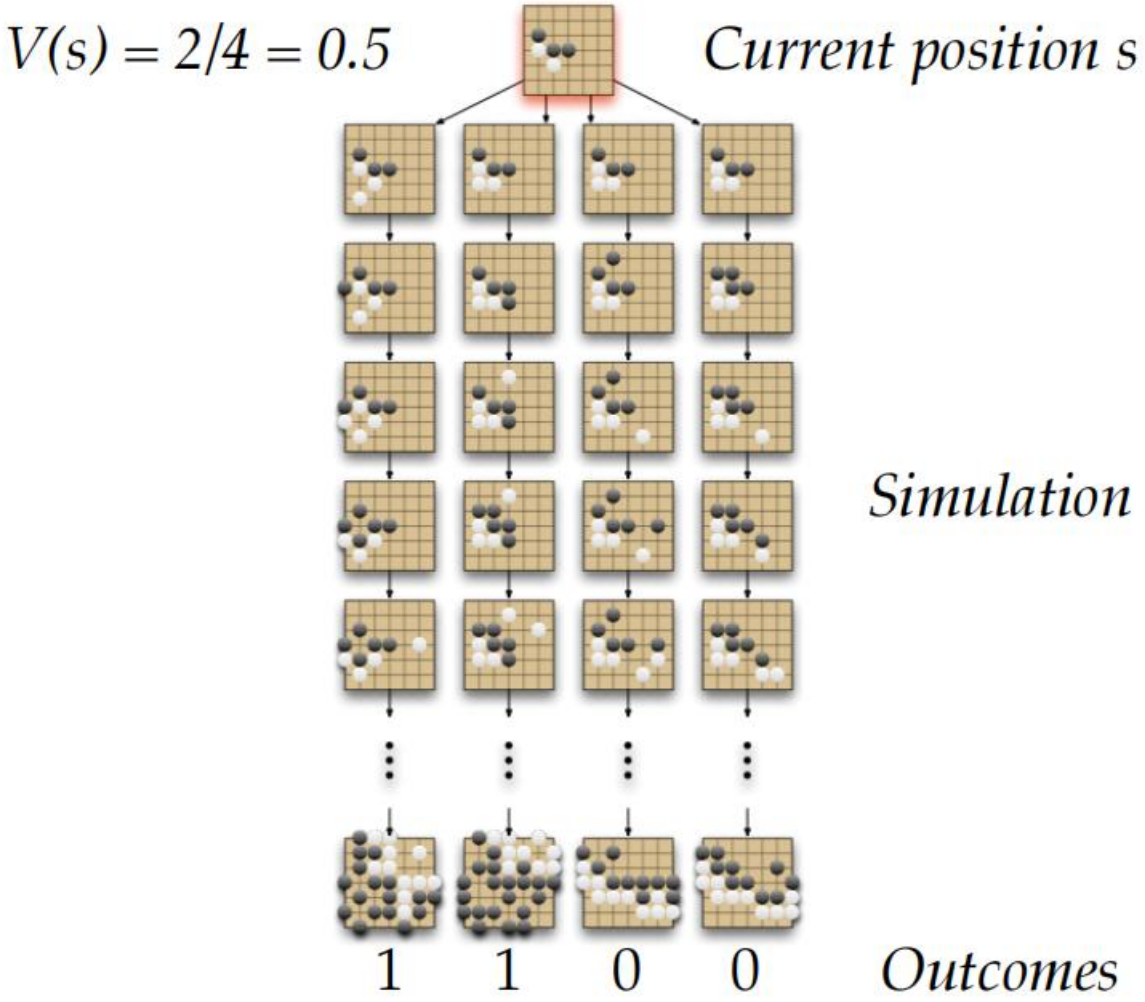
通常在针对双方博弈的游戏中，基于某一方（黑方）来设计某一状态的价值函数：

$$v_\pi(s) = \mathbb{E}_\pi \left[R_T \mid S = s \right] = \mathbb{P} \left[\text{Black wins} \mid S = s \right]$$

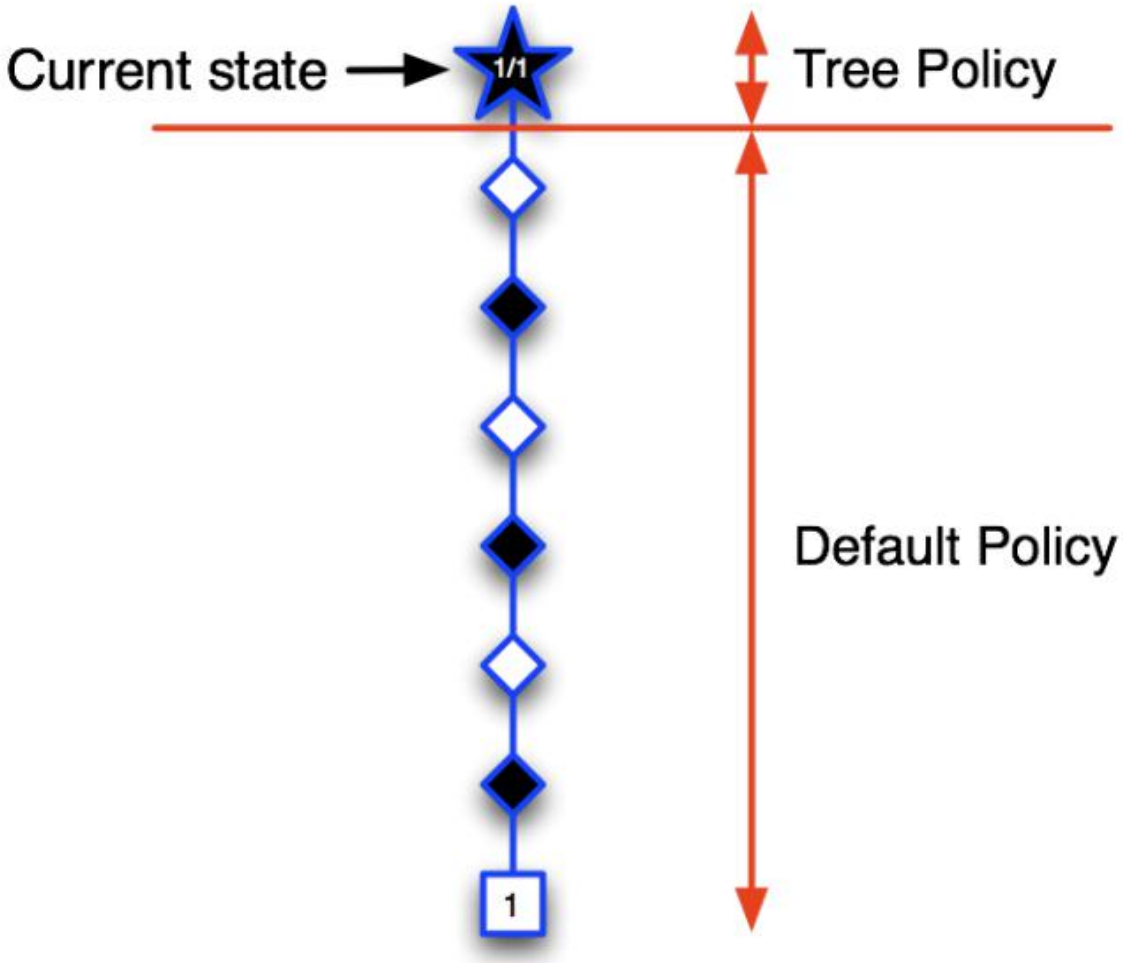
策略π下每一个状态的最优价值是：该策略下从白棋最小化状态价值列表中找到一个相对最大的价值：

$$v_*(s) = \max_{\pi_B} \min_{\pi_W} v_\pi(s)$$

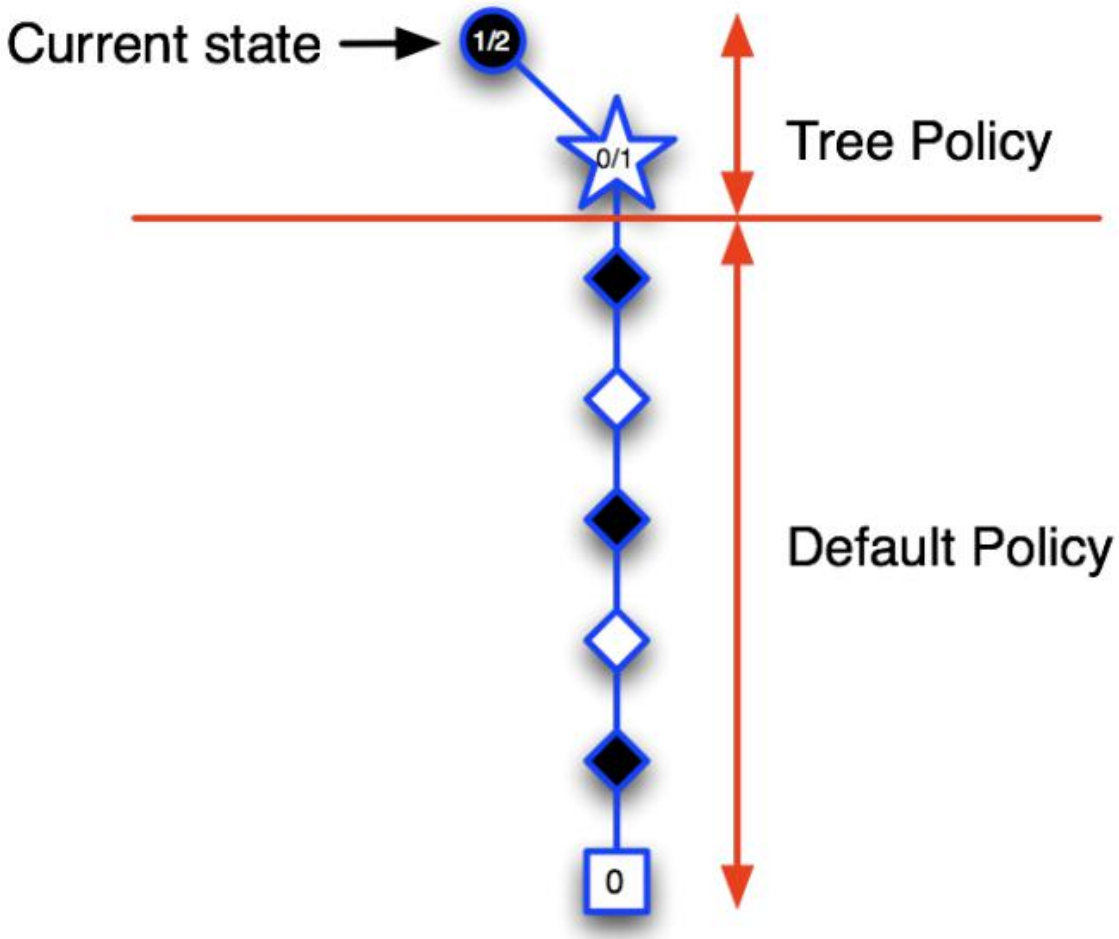
围棋中的蒙特卡罗评估



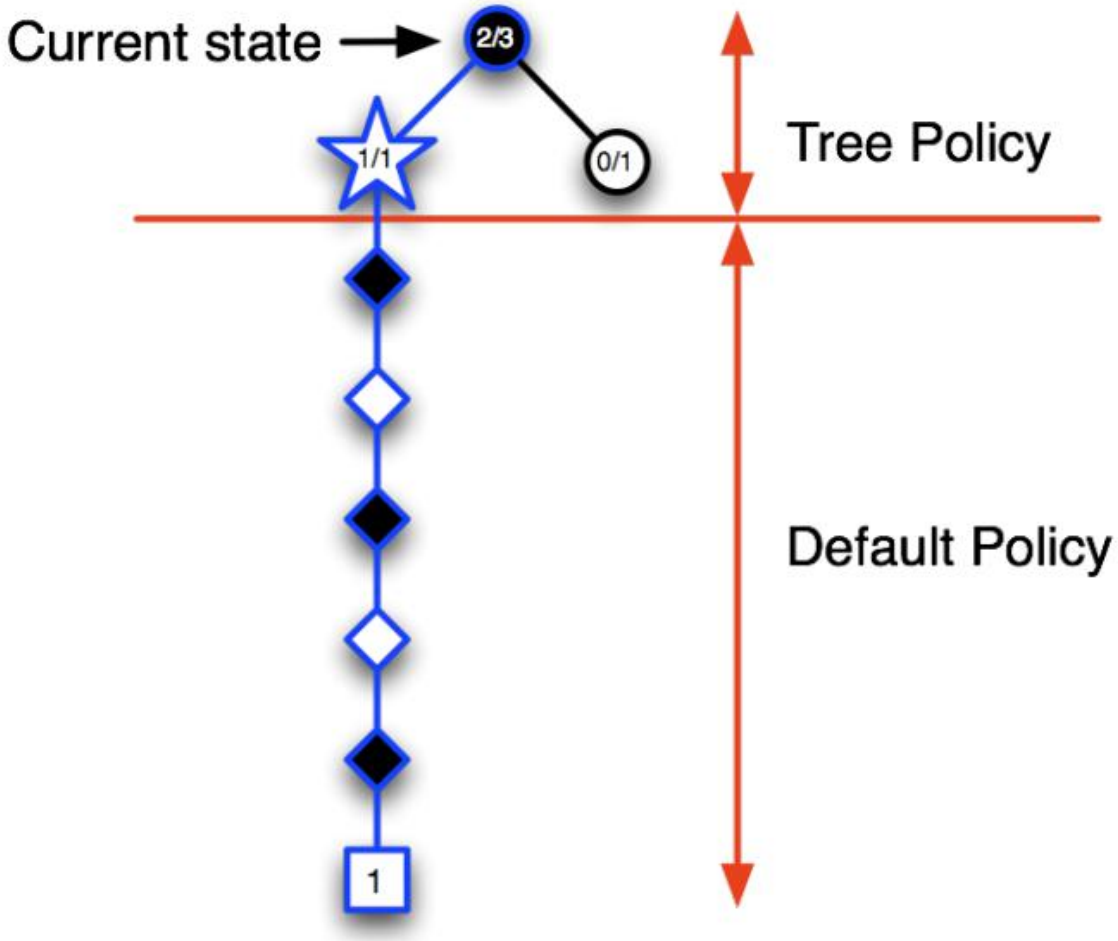
应用蒙特卡罗树搜索（1）



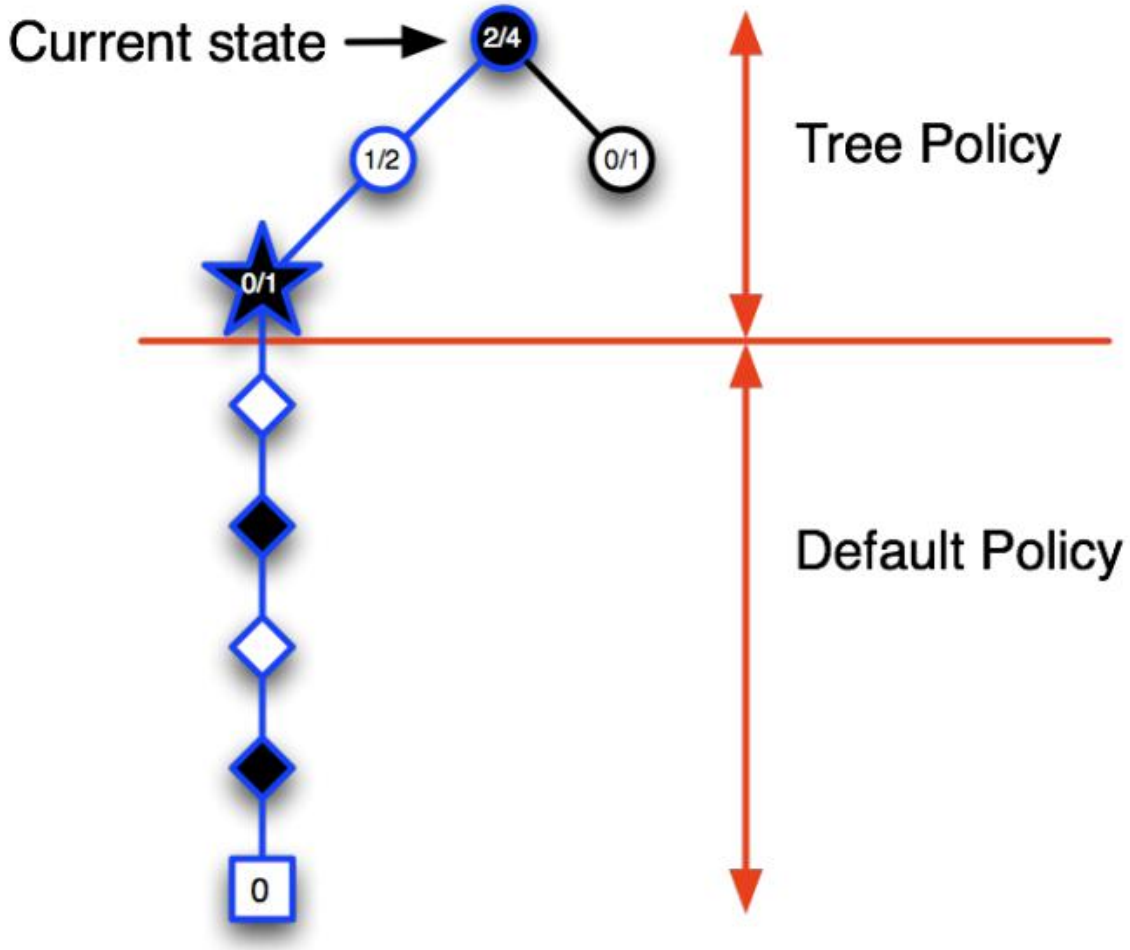
应用蒙特卡罗树搜索（2）



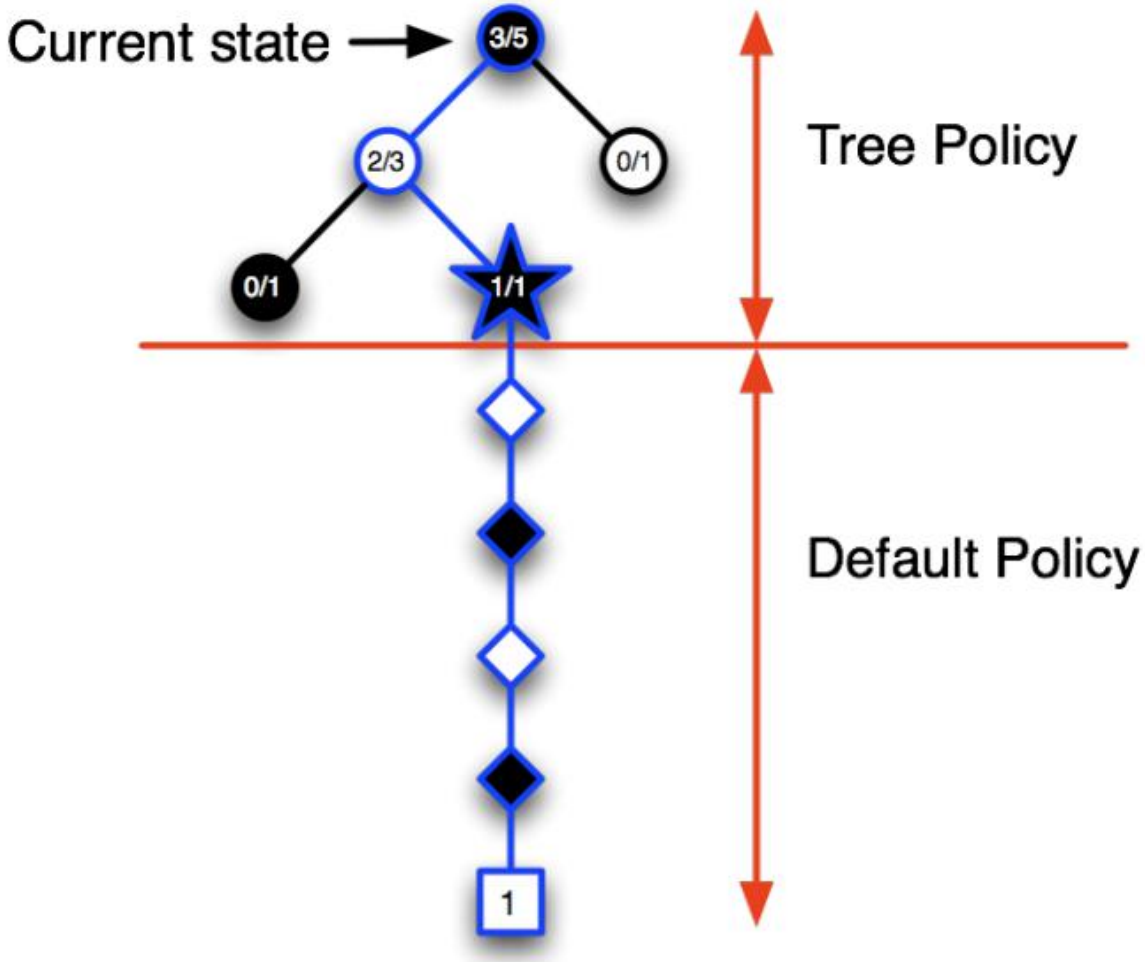
应用蒙特卡罗树搜索（3）



应用蒙特卡罗树搜索（4）



应用蒙特卡罗树搜索（5）



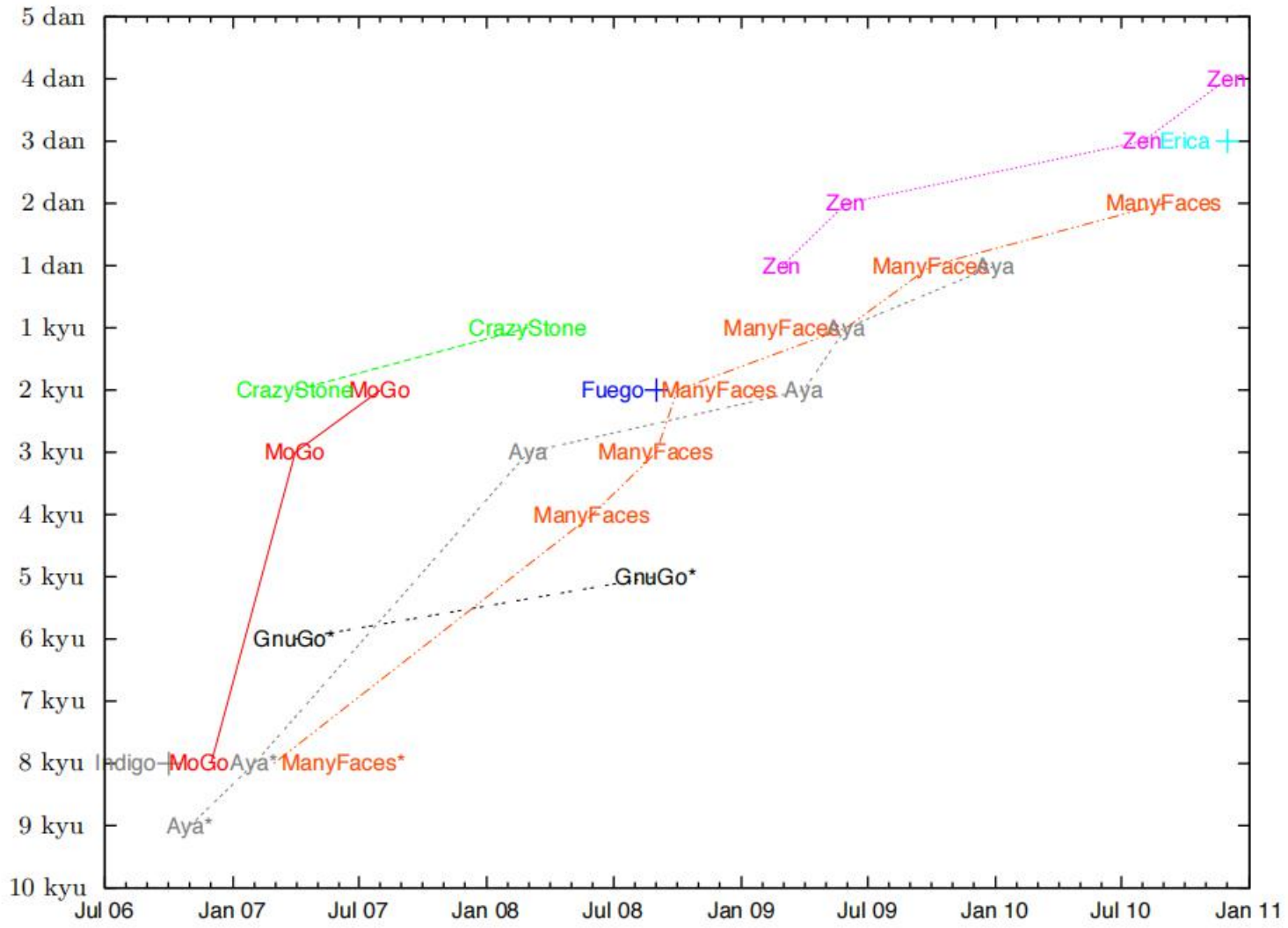
蒙特卡罗树搜索的优点 Advantages of MC Tree Search

1、首先蒙特卡罗树搜索是具有高度选择性的（Highly selective)、基于导致越好结果的行为越被优先选择（best-first）的一种搜索方法；

2、其次它可以动态的评估各状态的价值，这种动态更新价值的方法与动态规划不同，后者聚焦于整个状态空间，而蒙特卡罗树搜索是立足于当前状态，动态更新与该状态相关的状态价值；它使用采样避免了维度灾难；同样由于它仅依靠采样，因此适用于那些“黑盒”模型（black-box models）。

上述这些优点决定了其是可以高效计算的、不受时间限制以及可以并行处理的。

计算机运行中的蒙特卡罗树搜索



时序差分搜索 Temporal-Difference Search

特点：

1、基于仿真的搜索

2、使用TD而不是MC（bootstrapping）

3、蒙特卡罗树搜索是对从当前状态开始的一个子MDP问题应用蒙特卡罗控制，那么TD搜索可以被看成时对从当前状态开始的一个子MDP问题应用SARSA学习。

MC搜索 vs. TD 搜索

对于无模型的强化学习，bootstrapping是有帮助的

- TD学习减少了方差，但增加了偏差
- TD学习通常比MC更有效、
- TD（λ）可以比MC更有效

对于基于模拟的搜索，bootstrapping也很有帮助

- TD搜索减少了方差，但增加了偏差
- TD搜索通常比MC搜索更有效
- TD（λ）搜索比MC搜索要有效得多

TD搜索

1. 从当前实际状态 s_t 开始，模拟一系列Episodes，在这一过程中使用状态行为价值作为节点录入搜索树，

2. 对搜索树内的每一个节点（状态行为对），估计其价值Q(s,a)

3. 对于模拟过程中的每一步，使用Sarsa学习更新行为价值：

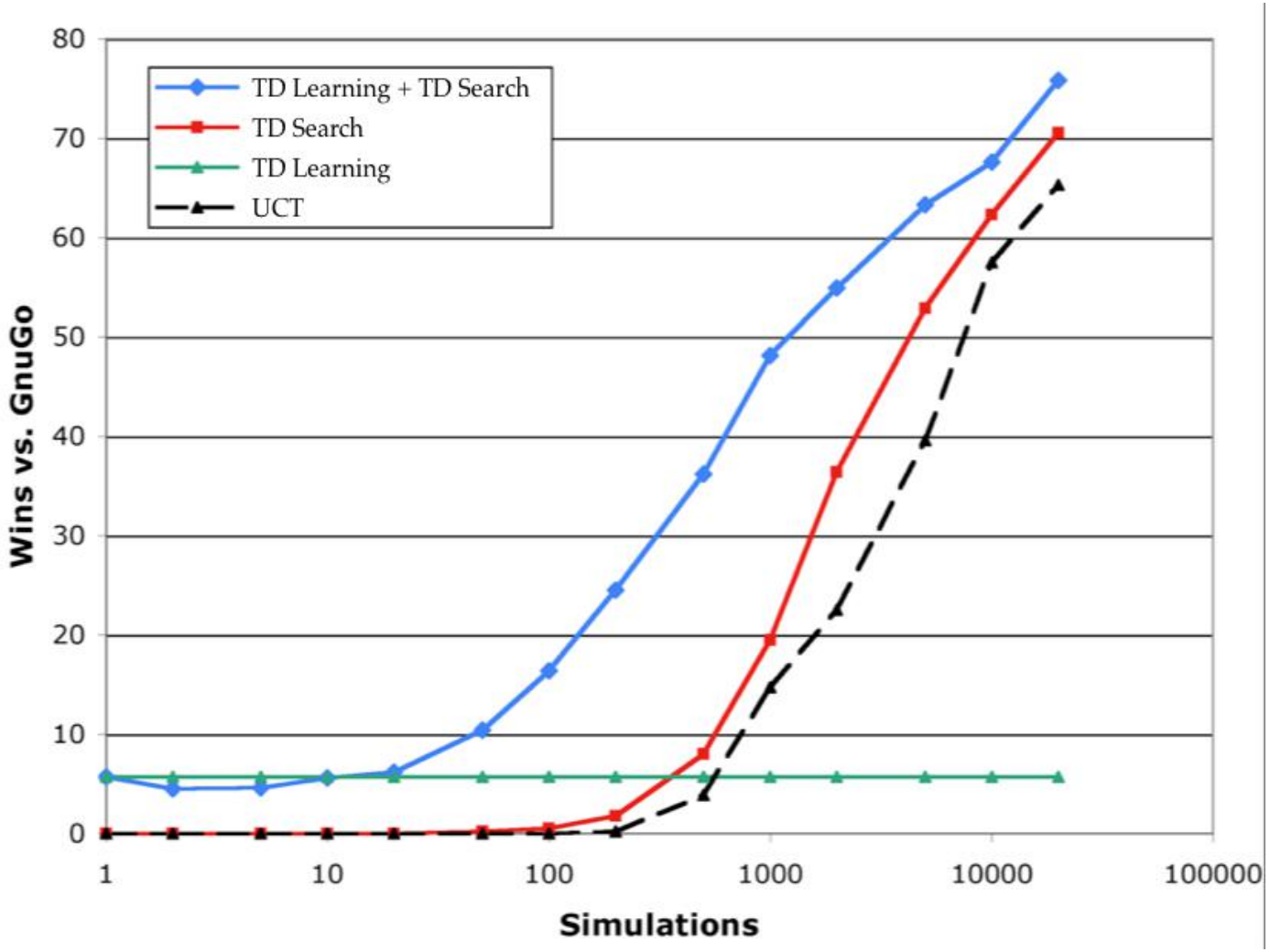
$$\Delta Q(S,A) = \alpha(R + \gamma Q(S',A') - Q(S,A))$$

4. 基于Q值，使用ε-greedy或其他探索方法来生成行为。

Dyna-2

- 在Dyna-2中，智能体存储了两组特征权值：
- 一组反映了智能体的长期记忆，该记忆是从真实经历中使用TD学习得到，它反映了个体对于某一特定强化学习问题的普遍性的知识、经验。
- 另一组反映了个体的短期记忆，该记忆从基于模拟经历中使用TD搜索得到，它反映了智能体对于某一特定强化学习在特定条件（比如某一Episode、某一状态下）下的特定的、局部适用的知识、经验
- Dyna-2算法最终将两套特征权重下产生的价值综合起来进行决策，以期得到更优秀的策略。

在围棋中的TD搜索结果



The End