

Planning and Optimization

G5. Monte-Carlo Tree Search Algorithms (Part I)

Malte Helmert and Gabriele Röger

Universität Basel

December 14, 2020

Planning and Optimization

December 14, 2020 — G5. Monte-Carlo Tree Search Algorithms (Part I)

G5.1 Introduction

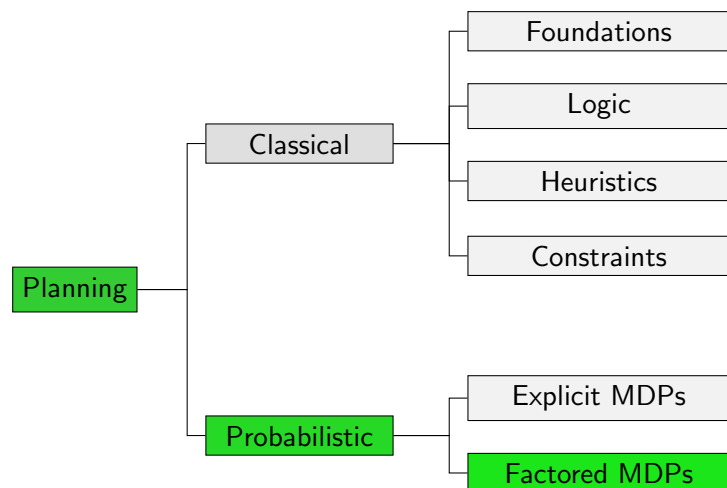
G5.2 Default Policy

G5.3 Asymptotic Optimality

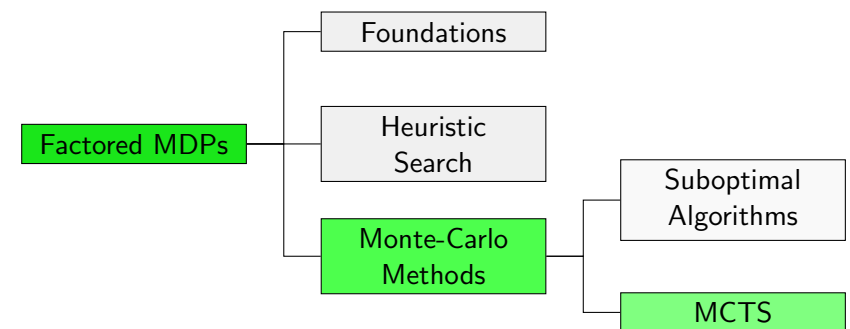
G5.4 Multi-armed Bandit Problem

G5.5 Summary

Content of this Course



Content of this Course: Factored MDPs

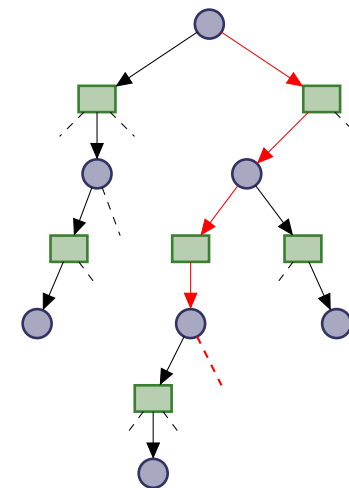


G5.1 Introduction

Monte-Carlo Tree Search: Reminder

Performs iterations with 4 phases:

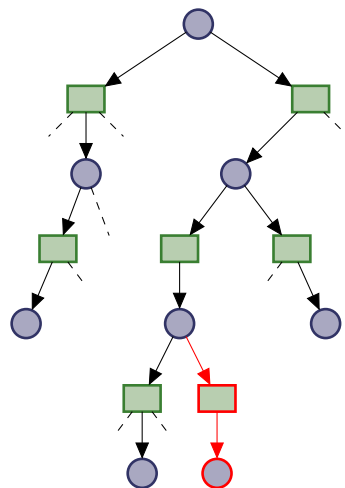
- ▶ **selection**: use **given tree policy** to traverse explicated tree
- ▶ **expansion**: add node(s) to the tree
- ▶ **simulation**: use **given default policy** to simulate run
- ▶ **backpropagation**: update visited nodes with Monte-Carlo backups



Monte-Carlo Tree Search: Reminder

Performs iterations with 4 phases:

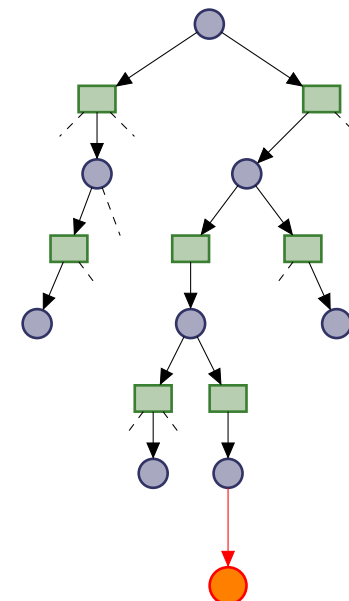
- ▶ **selection**: use **given tree policy** to traverse explicated tree
- ▶ **expansion**: add node(s) to the tree
- ▶ **simulation**: use **given default policy** to simulate run
- ▶ **backpropagation**: update visited nodes with Monte-Carlo backups



Monte-Carlo Tree Search: Reminder

Performs iterations with 4 phases:

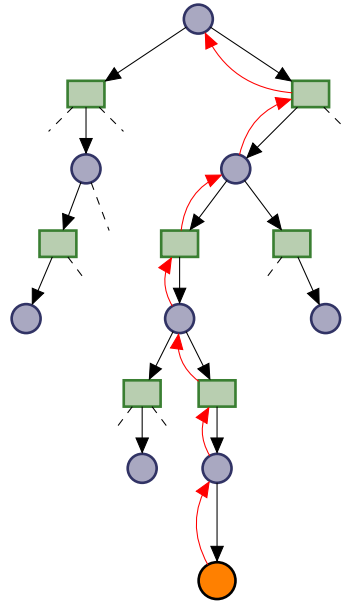
- ▶ **selection**: use **given tree policy** to traverse explicated tree
- ▶ **expansion**: add node(s) to the tree
- ▶ **simulation**: use **given default policy** to simulate run
- ▶ **backpropagation**: update visited nodes with Monte-Carlo backups



Monte-Carlo Tree Search: Reminder

Performs iterations with 4 phases:

- ▶ **selection**: use **given tree policy** to traverse explicated tree
- ▶ **expansion**: add node(s) to the tree
- ▶ **simulation**: use **given default policy** to simulate run
- ▶ **backpropagation**: update visited nodes with Monte-Carlo backups



Motivation

- ▶ Monte-Carlo Tree Search is a **framework** of algorithms
 - ▶ concrete MCTS algorithms are specified in terms of
 - ▶ a tree policy;
 - ▶ and a default policy
 - ▶ for most tasks, a **well-suited** MCTS configuration exists
 - ▶ but for each task, many MCTS configurations **perform poorly**
 - ▶ and every MCTS configuration that **works well** in one problem **performs poorly** in another problem
- ⇒ There is no “Swiss knife” configuration for MCTS

Role of Tree Policy

- ▶ used to **traverse explicated tree** from root node to a leaf
- ▶ maps **decision nodes** to a probability distribution over actions (usually as a function over a decision node and its children)
- ▶ **exploits** information from search tree
 - ▶ able to **learn over time**
 - ▶ requires MCTS tree to **memorize collected information**

Role of Default Policy

- ▶ used to **simulate run** from some state to a goal
- ▶ maps **states** to a probability distribution over actions
- ▶ **independent** from MCTS tree
 - ▶ does not improve over time
 - ▶ can be **computed quickly**
 - ▶ **constant memory** requirements
- ▶ **accumulated cost of simulated run** used to **initialize state-value estimate** of decision node

G5.2 Default Policy

MCTS Simulation

MCTS simulation with default policy π from state s

cost := 0

while $s \notin S_x$:

$a : \sim \pi(s)$

 cost := cost + $c(a)$

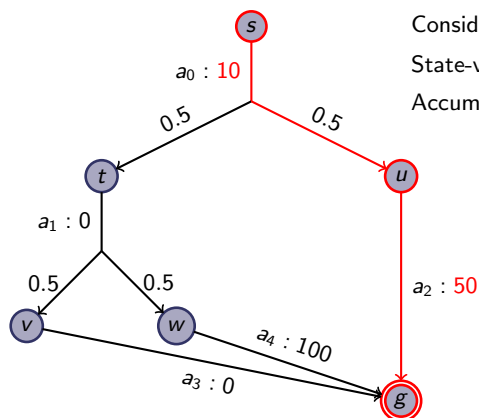
$s : \sim \text{succ}(s, a)$

return cost

Default policy must be **proper**

- ▶ to guarantee **termination** of the procedure
- ▶ and a **finite cost**

Default Policy: Example



Consider deterministic default policy π

State-value of s under π : 60

Accumulated cost of run: 60

Default Policy Realizations

- ▶ Early MCTS implementations used **random default policy**:

$$\pi(a | s) = \begin{cases} \frac{1}{|A(s)|} & \text{if } a \in A(s) \\ 0 & \text{otherwise} \end{cases}$$

- ▶ only **proper** if goal can be reached from each state
- ▶ **poor** guidance, and due to high variance even **misguidance**

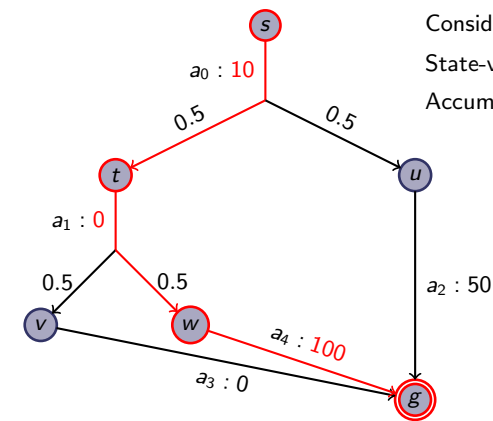
Default Policy Realizations

There are only **few alternatives** to random default policy, e.g.,

- ▶ heuristic-based policy
- ▶ domain-specific policy

Reason: No matter how good the policy, result of simulation can be arbitrarily poor

Default Policy: Example (2)



Consider deterministic default policy π

State-value of s under π : 60

Accumulated cost of run: 110

Default Policy Realizations

Possible **solution** to overcome this weakness:

- ▶ **average** over multiple random walks
- ▶ **converges** to true action-values of policy
- ▶ computationally **often very expensive**

Cheaper and more successful alternative:

- ▶ skip **simulation** step of MCTS
- ▶ use heuristic **directly for initialization** of state-value estimates
- ▶ instead of simulating execution of heuristic-guided policy
- ▶ much more successful (e.g. neural networks of **AlphaGo**)

G5.3 Asymptotic Optimality

Optimal Search

Heuristic search algorithms (like RTDP) achieve optimality by combining

- ▶ greedy search
- ▶ admissible heuristic
- ▶ Bellman backups

In Monte-Carlo Tree Search

- ▶ search behavior defined by a **tree policy**
- ▶ **admissibility** of default policy / heuristic **irrelevant** (and usually **not given**)
- ▶ **Monte-Carlo backups**

MCTS requires a different idea for **optimal behavior in the limit**.

Asymptotic Optimality

Asymptotic Optimality

Let an MCTS algorithm build an MCTS tree $\mathcal{G} = \langle d_0, D, C, E \rangle$. The MCTS algorithm is **asymptotically optimal** if

$$\lim_{k \rightarrow \infty} \hat{Q}^k(c) = Q_*(s(c), a(c)) \text{ for all } c \in C^k,$$

where k is the number of trials.

- ▶ this is just one special form of asymptotic optimality
- ▶ some optimal MCTS algorithms are not asymptotically optimal by this definition (e.g., $\lim_{k \rightarrow \infty} \hat{Q}^k(c) = \ell \cdot Q_*(s(c), a(c))$ for some $\ell \in \mathbb{R}^+$)
- ▶ all **practically relevant** optimal MCTS algorithms are asymptotically optimal by this definition

Asymptotically Optimal Tree Policy

An MCTS algorithm is **asymptotically optimal** if

- 1 its tree policy **explores forever**:
 - ▶ the (infinite) sum of the probabilities that a decision node is visited must diverge
 - ▶ \Rightarrow every search node is **explicated eventually** and **visited infinitely often**
- 2 its tree policy is **greedy in the limit**:
 - ▶ probability that optimal action is selected converges to 1
 - ▶ \Rightarrow in the limit, backups based on iterations where only an **optimal policy** is followed dominate suboptimal backups
- 3 its default policy initializes decision nodes with **finite values**

Example: Random Tree Policy

Example

Consider the **random tree policy** for decision node d where:

$$\pi(a | d) = \begin{cases} \frac{1}{|A(s(d))|} & \text{if } a \in A(s(d)) \\ 0 & \text{otherwise} \end{cases}$$

The random tree policy **explores forever**:

Let $\langle d_0, c_0, \dots, d_n, c_n, d \rangle$ be a sequence of connected nodes in \mathcal{G}^k and let $p := \min_{0 < i < n-1} T(s(d_i), a(c_i), s(d_{i+1}))$.

Let \mathbb{P}^k be the probability that d is visited in trial k . With $\mathbb{P}^k \geq (\frac{1}{|A|} \cdot p)^n$, we have that

$$\lim_{k \rightarrow \infty} \sum_{i=1}^k \mathbb{P}^k \geq k \cdot \left(\frac{1}{|A|} \cdot p\right)^n = \infty$$

Example: Random Tree Policy

Example

Consider the **random tree policy** for decision node d where:

$$\pi(a | d) = \begin{cases} \frac{1}{|A(s(d))|} & \text{if } a \in A(s(d)) \\ 0 & \text{otherwise} \end{cases}$$

The random tree policy is **not greedy in the limit** unless all actions are always optimal:

The probability that an optimal action a is selected in decision node d is

$$\lim_{k \rightarrow \infty} 1 - \sum_{\{a' \notin \pi_{V^*}(s)\}} \frac{1}{|A(s(d))|} < 1.$$

↪ MCTS with random tree policy **not asymptotically optimal**

Example: Greedy Tree Policy

Example

Consider the **greedy tree policy** for decision node d where:

$$\pi(a | d) = \begin{cases} \frac{1}{|A_*^k(d)|} & \text{if } a \in A_*^k(d) \\ 0 & \text{otherwise,} \end{cases}$$

with $A_*^k(d) = \{a(c) \in A(s(d)) \mid c \in \arg \min_{c' \in \text{children}(d)} \hat{Q}^k(c')\}$.

- ▶ Greedy tree policy is **greedy in the limit**
 - ▶ Greedy tree policy does **not explore forever**
- ↪ MCTS with greedy tree policy **not asymptotically optimal**

Tree Policy: Objective

To satisfy **both** requirements, MCTS tree policies have two contradictory objectives:

- ▶ **explore** parts of the search space that have not been investigated thoroughly
- ▶ **exploit** knowledge about good actions to focus search on promising areas of the search space

central challenge: **balance** exploration and exploitation

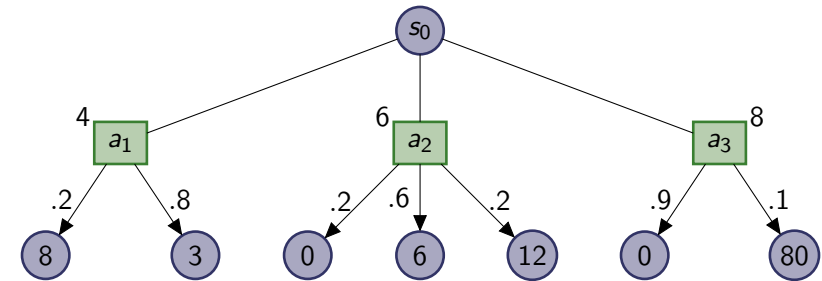
⇒ borrow ideas from related **multi-armed bandit** problem

G5.4 Multi-armed Bandit Problem

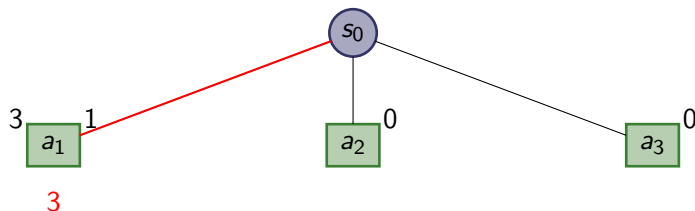
Multi-armed Bandit Problem

- ▶ most commonly used tree policies are **inspired** from research on the multi-armed bandit problem (MAB)
- ▶ MAB is a **learning** scenario (model not revealed to agent)
- ▶ agent repeatedly faces the same decision: to pull one of several arms of a **slot machine**
- ▶ pulling an arm yields **stochastic reward**
⇒ in MABs, we have rewards rather than costs
- ▶ can be modeled as an MDP

Multi-armed Bandit Problem: Planning Scenario

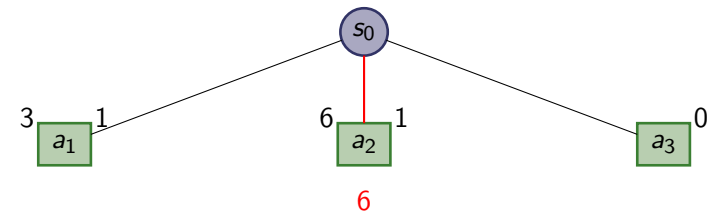


Multi-armed Bandit Problem: Learning Scenario



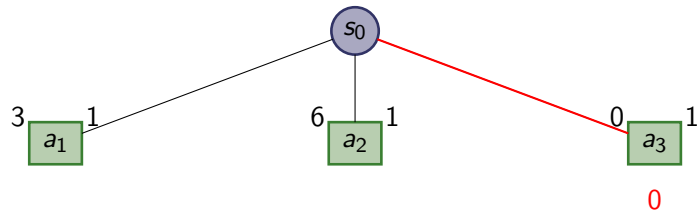
- ▶ Pull arms following **policy** to **explore** or **exploit**
- ▶ Update \hat{Q} and N based on observations
- ▶ Accumulated reward after 1 trial is 3

Multi-armed Bandit Problem: Learning Scenario



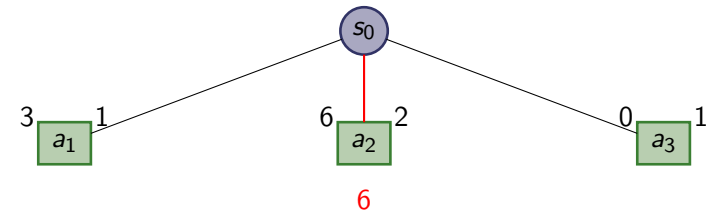
- ▶ Pull arms following **policy** to **explore** or **exploit**
- ▶ Update \hat{Q} and N based on observations
- ▶ Accumulated reward after 2 trials is $3 + 6 = 9$

Multi-armed Bandit Problem: Learning Scenario



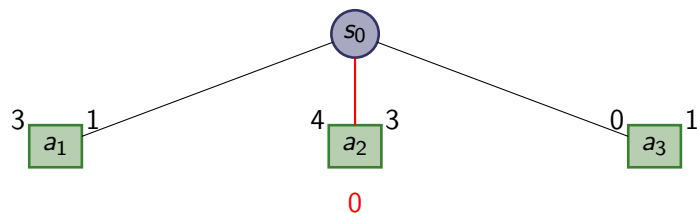
- ▶ Pull arms following **policy** to **explore** or **exploit**
- ▶ Update \hat{Q} and N based on observations
- ▶ Accumulated reward after 3 trials is $3 + 6 + 0 = 9$

Multi-armed Bandit Problem: Learning Scenario



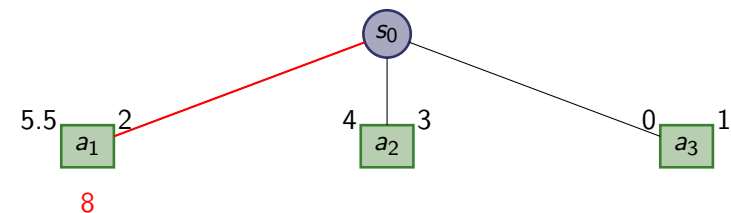
- ▶ Pull arms following **policy** to **explore** or **exploit**
- ▶ Update \hat{Q} and N based on observations
- ▶ Accumulated reward after 4 trials is $3 + 6 + 0 + 6 = 15$

Multi-armed Bandit Problem: Learning Scenario



- ▶ Pull arms following **policy** to **explore** or **exploit**
- ▶ Update \hat{Q} and N based on observations
- ▶ Accumulated reward after 5 trials is $3 + 6 + 0 + 6 + 0 = 15$

Multi-armed Bandit Problem: Learning Scenario



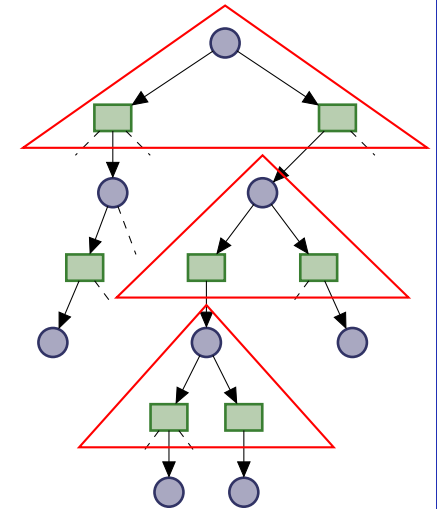
- ▶ Pull arms following **policy** to **explore** or **exploit**
- ▶ Update \hat{Q} and N based on observations
- ▶ Accumulated reward after 6 trials is $3 + 6 + 0 + 6 + 0 + 8 = 23$

Policy Quality

- ▶ Since model unknown to MAB agent, it cannot achieve accumulated reward of $k \cdot V_*$ with $V_* := \max_a Q_*(a)$ in k trials
- ▶ Quality of MAB policy π measured in terms of **regret**, i.e., the difference between $k \cdot V_*$ and expected reward of π in k trials
- ▶ Regret cannot grow slower than **logarithmically** in the number of trials

MABs in MCTS Tree

- ▶ many tree policies treat **each decision node as MAB**
- ▶ where each action yields a **stochastic reward**
- ▶ dependence of reward on future decision is **ignored**
- ▶ MCTS **planner** uses simulations to **learn reasonable** behavior
- ▶ SSP model is **not considered**



G5.5 Summary

Summary

- ▶ The simulation phase **simulates** the **execution** of the default policy
- ▶ MCTS algorithms are **optimal in the limit** if
 - ▶ the tree policy is **greedy in the limit**,
 - ▶ the tree policy **explores forever**, and
 - ▶ the default policy **initializes with finite value**
- ▶ Central challenge of most tree policies: balance **exploration** and **exploitation**
- ▶ each decision of an MCTS tree policy can be viewed as an **multi-armed bandit** problem.