

Planning and Optimization

G6. Monte-Carlo Tree Search: Algorithms Part I

Gabriele Röger and Thomas Keller

Universität Basel

December 12, 2018

Planning and Optimization

December 12, 2018 — G6. Monte-Carlo Tree Search: Algorithms Part I

G6.1 Motivation

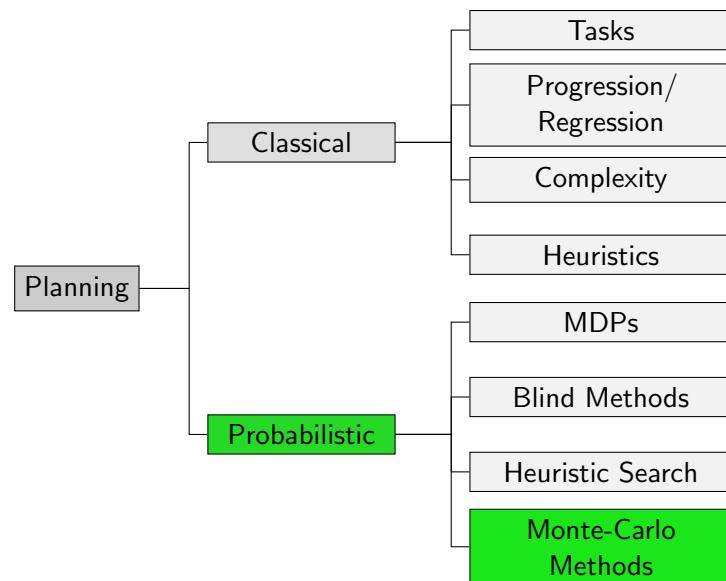
G6.2 Default Policy

G6.3 Asymptotic Optimality

G6.4 Multi-armed Bandit Problem

G6.5 Summary

Content of this Course



G6.1 Motivation

Motivation

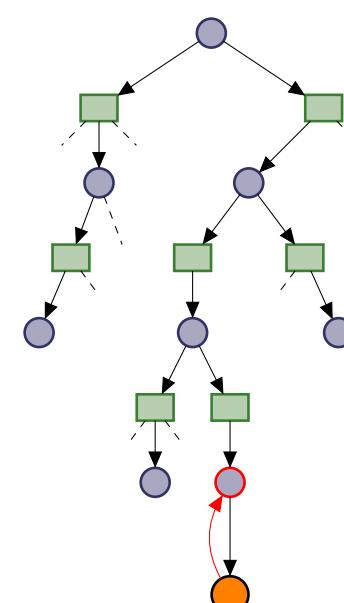
- ▶ Monte-Carlo Tree Search is a **framework** of algorithms
- ▶ Concrete MCTS algorithms are specified in terms of:
 - ▶ tree policy
 - ▶ default policy
- ▶ For most tasks, a **well-suited** MCTS configuration exists
- ▶ **But:** for each task, many MCTS configurations **ill-suited**
- ▶ **And:** every MCTS configuration that **works well** in one problem **performs poorly** in another problem

⇒ no dominating MCTS configuration

⇒ we present and analyze different tree and default policies

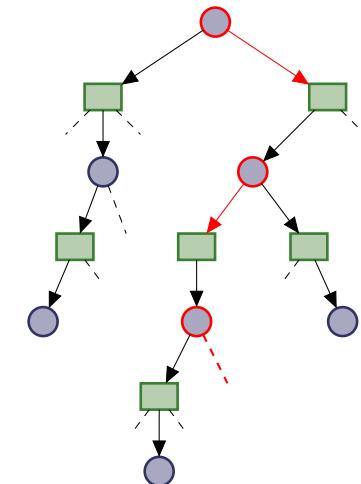
Tree Policy: Recap

- ▶ Tree policy used to **traverse** **exploited tree**, starting at root
- ▶ Assigns probability distribution over actions to each **decision node**
- ▶ May access information from current search tree
- ▶ Comparable to **evaluation function** in best-first search
- ▶ Tree policy **more general**: evaluation function determined upon node generation, while tree policy dynamic in each trial



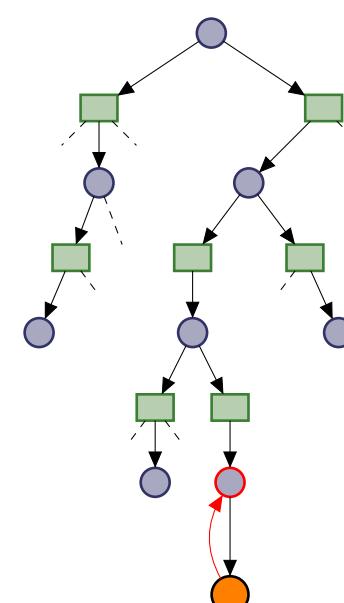
Tree Policy: Recap

- ▶ Tree policy used to **traverse** **exploited tree**, starting at root
- ▶ Assigns probability distribution over actions to each **decision node**
- ▶ May access information from current search tree
- ▶ Comparable to **evaluation function** in best-first search
- ▶ Tree policy **more general**: evaluation function determined upon node generation, while tree policy dynamic in each trial



Default Policy: Recap

- ▶ Default policy used to **simulate** **run**, starting at recently added decision node
- ▶ Assigns probability distribution over actions to each **state**
- ▶ Independent from current search tree
- ▶ Same role in MCTS as **heuristic** in heuristic search
- ▶ Heuristic **more general**: default policy is a specific kind of heuristic



G6.2 Default Policy

Default Policy

Default Policy for state s of SSP \mathcal{T}

```
cost = 0
while  $s \notin S_*$ :
    sample action  $a$  from default policy  $\pi(\cdot | s)$ 
    cost := cost +  $c(a)$ 
     $s \sim \text{succ}(s, a)$ 
return cost
```

- ▶ Default policy used to **simulate run**
- ▶ Role of default policy comparable to role of heuristic
- ▶ Heuristic value is **accumulated cost** of simulated run under default policy

Default Policy as Heuristic: Properties

Is a default policy

- ▶ goal-aware? \Rightarrow **Yes**
- ▶ safe? \Rightarrow Only for **proper** policy and **dead end free** SSP. Otherwise, no guarantee for **termination** of computation
- ▶ admissible? \Rightarrow Possible for few SSPs (e.g., if optimal policy is deterministic), but usually **not**.
- ▶ consistent? \Rightarrow Possible for few SSPs (e.g., if optimal policy is deterministic), but usually **not**.

Default Policy Realizations

- ▶ Early work on MCTS proposed **random walk** default policy:

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

- ▶ Random walks are **proper**
- ▶ **Poor** guidance, and due to high variance even **misguidance**
 \Rightarrow Variant: run multiple random walks and use average
- ▶ **Computation expensive** if probability to reach goal is low
 \Rightarrow Variant: apply **heuristic** after finite number of steps
- ▶ Predominant alternative: **domain-dependent** solutions
e.g., neural networks of **AlphaGo** variants

G6.3 Asymptotic Optimality

Optimal Search

- ▶ Optimal heuristic search algorithms AO* and RTDP use **greedy policy**
- ▶ with **admissible heuristic**
- ▶ and **full Bellman backups** to guarantee optimality
- ▶ MCTS uses **Monte-Carlo backups**
- ▶ and default policy is **not admissible**

⇒ MCTS requires different way to guarantee **optimality**

Asymptotic Optimality

Asymptotic Optimality

An MCTS algorithm is **asymptotically optimal** if $\hat{Q}^k(c)$ converges to the optimal action-value $Q_*(s(c), a(c))$ for all $c \in \text{succ}(d_0)$ when the number of trials k approaches infinity ($k \rightarrow \infty$).

Note: this definition does not catch all MCTS configurations that are asymptotically optimal
(e.g., if all $\hat{Q}^k(c)$ converge to $\ell \cdot Q_*(s(c), a(c))$ for some $\ell \in \mathbb{R}^+$).

Asymptotically Optimal Tree Policy

An MCTS algorithm is **asymptotically optimal** if

- ➊ its tree policy **explores forever**:
 - ▶ the (infinite) sum of the probabilities that a decision node is visited must diverge
 - ▶ ⇒ every search node is **exploited eventually** and **visited infinitely often**
- ➋ its tree policy is **greedy in the limit**:
 - ▶ probability that optimal action is selected converges to 1
 - ▶ ⇒ in the limit, backups based on iterations where only an **optimal policy** is followed dominate suboptimal backups
- ➌ its default policy initializes decision nodes with **finite values**
 - ▶ ⇒ require **proper** default policy and **dead end free** SSP

Example: Uniform Tree Policy

Example

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

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

The random tree policy **explores forever**:

a decision node at depth h is visited with probability at least $(\frac{1}{|L|} \cdot \underline{p})^h$, where $\underline{p} := \min_{\{s, a, s' | T(s, a, s') > 0\}} T(s, a, s')$ and hence

$$\sum_{i=1}^k (\frac{1}{|L|} \cdot \underline{p})^h = k \cdot (\frac{1}{|L|} \cdot \underline{p})^h \rightarrow \infty \text{ when } k \rightarrow \infty$$

Example: Uniform Tree Policy

Example

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

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

The random tree policy is **not greedy in the limit**:

the probability that an optimal action a is selected in node d is

$$1 - \sum_{\{a' \in L(s(d)) \mid a' \text{ suboptimal}\}} \frac{1}{|L(s(d))|} \not\rightarrow 1 \text{ when } k \rightarrow \infty.$$

~~ Random tree policy **not asymptotically optimal**

Example: Greedy Tree Policy

Example

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

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

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

- ▶ Greedy tree policy is **greedy in the limit**
- ▶ Greedy tree policy does **not explore forever**

~~ 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

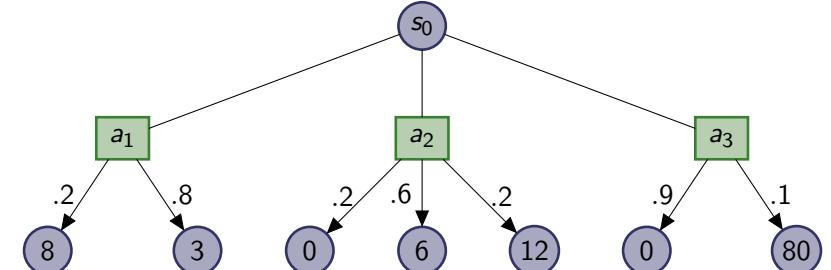
⇒ borrow ideas from related **multi-armed bandit** problem

G6.4 Multi-armed Bandit Problem

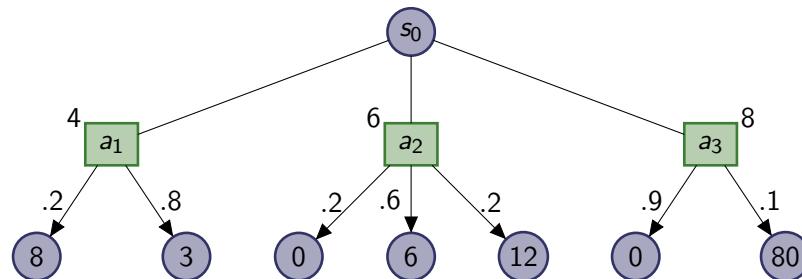
Multi-armed Bandit Problem

- 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**
Note: In this section, we consider rewards rather than costs
- can be modeled as MDP

Multi-armed Bandit Problem

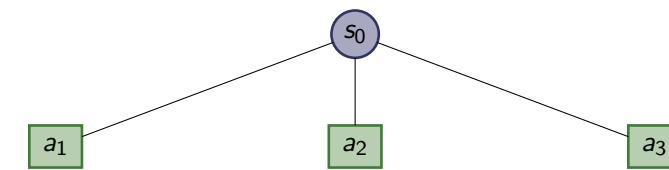


Multi-armed Bandit Problem: Planning Scenario



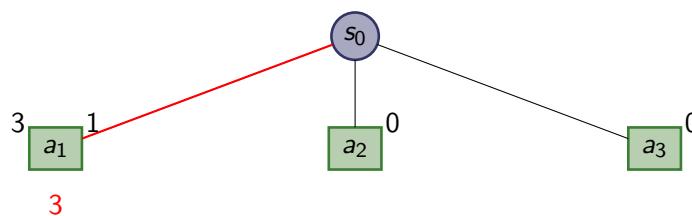
- Compute $Q_*(a)$ for $a \in \{a_1, a_2, a_3\}$
- Pull arm $\arg \max_{a \in \{a_1, a_2, a_3\}} Q_*(a) = a_3$ forever
- Expected accumulated reward after k trials is $8 \cdot k$

Multi-armed Bandit Problem: Learning Scenario



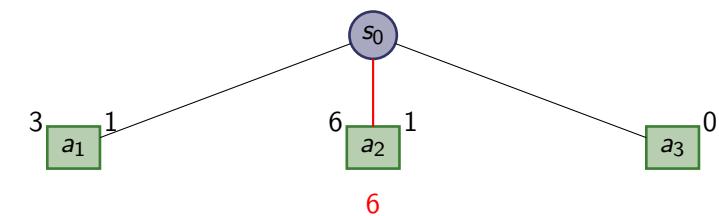
- Pull arms following **policy** to **explore** or **exploit**
- Update \hat{Q} and N based on observations
-

Multi-armed Bandit Problem: Learning Scenario



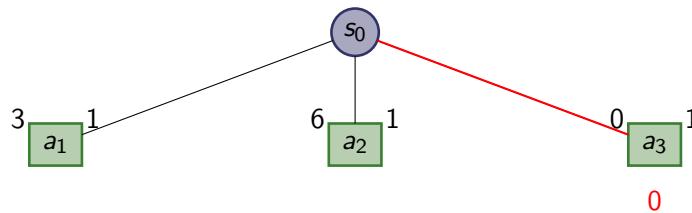
- ▶ Pull arms following **policy** to **explore** or **exploit**
- ▶ Update \hat{Q} and N based on observations
- ▶ Accumulated reward after 1 trial is 3

Multi-armed Bandit Problem: Learning Scenario



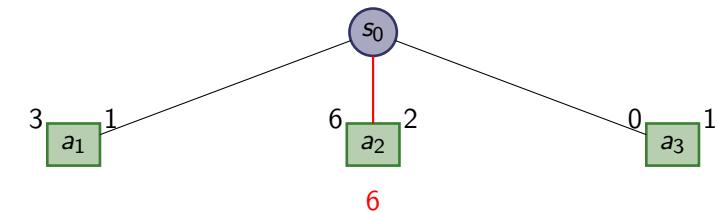
- ▶ Pull arms following **policy** to **explore** or **exploit**
- ▶ Update \hat{Q} and N based on observations
- ▶ Accumulated reward after 2 trials is $3 + 6 = 9$

Multi-armed Bandit Problem: Learning Scenario



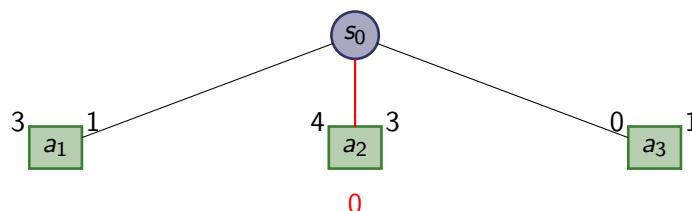
- ▶ Pull arms following **policy** to **explore** or **exploit**
- ▶ Update \hat{Q} and N based on observations
- ▶ Accumulated reward after 3 trials is $3 + 6 + 0 = 9$

Multi-armed Bandit Problem: Learning Scenario



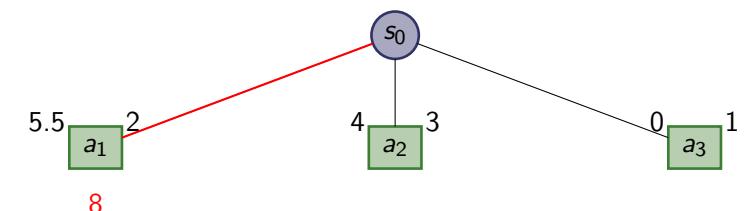
- ▶ Pull arms following **policy** to **explore** or **exploit**
- ▶ Update \hat{Q} and N based on observations
- ▶ Accumulated reward after 4 trials is $3 + 6 + 0 + 6 = 15$

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 **logarithmic** in number of trials

Connection between MCTS Tree Policy and MAB

~~> Blackboard

G6.5 Summary

Summary

- ▶ Default policies **simulate run** under policy
- ▶ Default policy **not admissible**
- ▶ MCTS requires **different idea** to achieve **optimality** than heuristic search:
tree policy must be **greedy in the limit** and **explore forever**
- ▶ Central challenge of tree policies:
balance **exploration** and **exploitation**
- ▶ Each decision of MCTS tree policy can be viewed as **multi-armed bandit** problem