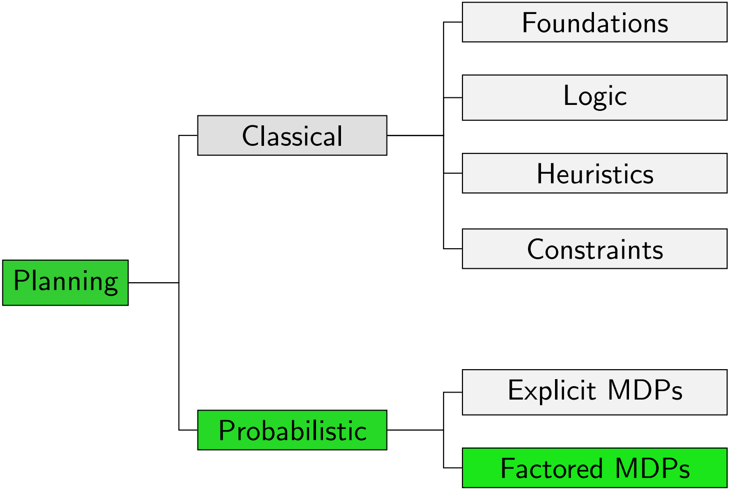
Planning and Optimization

F7. Monte-Carlo Tree Search: Framework

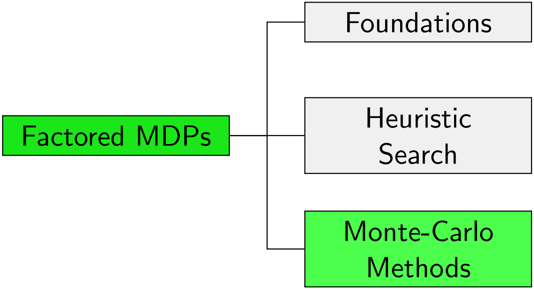
Malte Helmert and Gabriele Röger

Universität Basel

Content of this Course



Content of this Course: Factored MDPs



History

Monte-Carlo Methods: Brief History

- 1930s: first researchers experiment with Monte-Carlo methods
- 1998: Ginsberg's GIB player competes with Bridge experts
- 2002: Kearns et al. propose Sparse Sampling
- 2002: Auer et al. present UCB1 action selection for multi-armed bandits
- 2006: Coulom coins term Monte-Carlo Tree Search (MCTS)
- 2006: Kocsis and Szepesvári combine UCB1 and MCTS to the famous MCTS variant, UCT
- 2007–2016: Constant progress of MCTS in Go culminates in AlphaGo's historical defeat of dan 9 player Lee Sedol

Monte-Carlo Methods

Monte-Carlo Methods: Idea

- Summarize a broad family of algorithms
- Decisions are based on random samples (Monte-Carlo sampling)
- Results of samples are aggregated by computing the average (Monte-Carlo backups)
- Apart from that, algorithms can differ significantly

Careful: Many different definitions of MC methods in the literature

Types of Random Samples

Random samples have in common that something is drawn from a given probability distribution. Some examples:

- a determinization is sampled (Hindsight Optimization)
- runs under a fixed policy are simulated (Policy Simulation)
- considered outcomes are sampled (Sparse Sampling)
- runs under an evolving policy are simulated (Monte-Carlo Tree Search)

Reminder: Bellman Backups

Algorithms like Value Iteration or (L)RTDP use the Bellman equation as an update procedure.

The i-th state-value estimate of state s, $\hat{V}^i(s)$, is computed with Bellman backups as

$$\hat{V}^i(s) := \min_{a \in A(s)} \left(c(a) + \sum_{s' \in S} T(s, a, s') \cdot \hat{V}^{i-1}(s') \right).$$

(Some algorithms use a heuristic if the state-value estimate on the right hand side of the Bellman backup is undefined.)

Monte-Carlo Backups

Monte-Carlo methods instead estimate state-values by averaging over all samples.

Let $N^i(s)$ be the number of samples for state s in the first i algorithm iterations and let $cost^k(s)$ be the cost for s in the k-th sample ($cost^k(s) = 0$ if the k-th sample has no estimate for s).

The i-th state-value estimate of state s, $\hat{V}^i(s)$, is computed with Monte-Carlo backups as

$$\hat{V}^i(s) := \frac{1}{N^i(s)} \cdot \sum_{k=1}^{i} cost^k(s).$$

Monte-Carlo Backups: Properties

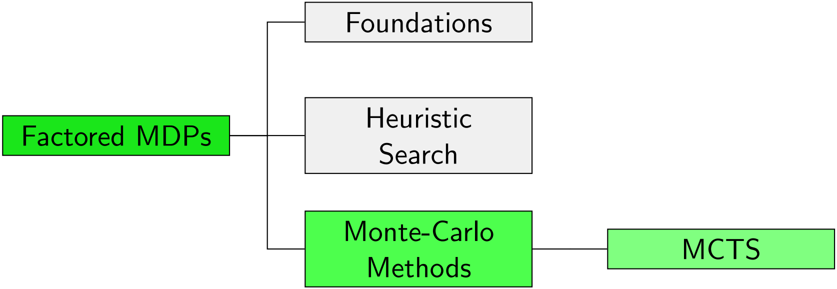
- no need to store $cost^k(s)$ for $k = 1, \ldots, i$:
  it is possible to compute Monte-Carlo backups iteratively as

$$\hat{V}^i(s) := \hat{V}^{i-1}(s) + \frac{1}{N^i(s)}(cost^i(s) - \hat{V}^{i-1}(s))$$

- no need to know SSP model for backups
- if s is a random variable, $\hat{V}^i(s)$ converges to $\mathbb{E}[s]$
  due to the strong law of large numbers
- if s is not a random variable, this is not always the case

MCTS Tree

Content of this Course: Factored MDPs



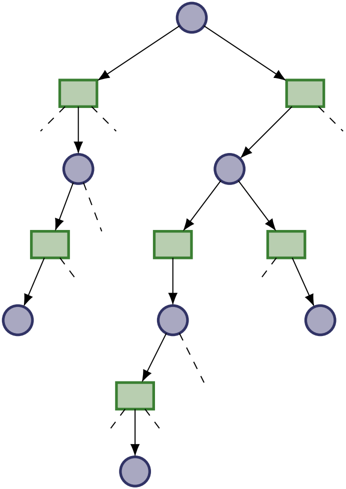
Monte-Carlo Tree Search

- While Monte-Carlo Tree Search (MCTS) has wildely been used for games, we only consider the case for SSPs.
- MCTS successively builds up the most promising parts of the search tree by repeated random sampling of the search space.
- Like (L)RTDP, MCTS performs trials (also called rollouts).
- In each trials, it extends the search tree with potentially interesting nodes.
- It uses Monte-Carlo backups to improve the state-value estimates with the information gathered in the trial.

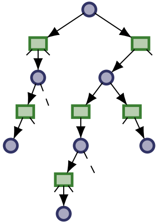
To be more specific, we need to know the details of the MCTS tree.

MCTS Tree

- Unlike previous methods, the SSP is explicated as a tree
- Duplicates (also: transpositions) possible,
  i.e., multiple search nodes with identical associated state
- Search tree can (and often will) have unbounded depth



Tree Structure



- Differentiate between two types of search nodes:
  - Decision nodes
  - Chance nodes
- Search nodes correspond 1:1 to traces from initial state
- Decision and chance nodes alternate
- Decision nodes correspond to states in a trace
- Chance nodes correspond to actions in a trace
- Decision nodes have one child node for each applicable action (if all children are explicated)
- Chance nodes have one child node for each outcome (if all children are explicated)

MCTS Tree

Definition (MCTS Tree)

An MCTS tree is given by a tuple $\mathcal{G} = \langle d_0, D, C, E \rangle$, where

- D and C are disjoint sets of decision and chance nodes (simply search node if the type does not matter)
- $d_0 \in D$ is the root node
- $E \subseteq (D \times C) \cup (C \times D)$ is the set of edges such that the graph $\langle D \cup C, E \rangle$ is a tree

Note: can be regarded as an AND/OR tree

Search Node Annotations

Definition (Search Node Annotations)

Let $\mathcal{G} = \langle d_0, D, C, E \rangle$ be an MCTS Tree.

- Each search node $n \in D \cup C$ is annotated with
  - a visit counter $N(n)$
  - a state $s(n)$
- Each decision node $d \in D$ is annotated with
  - a state-value estimate $\hat{V}(d)$
  - a probability $p(d)$
- Each chance node $c \in C$ is annotated with
  - an action-value estimate (or Q-value estimate) $\hat{Q}(c)$
  - an action $a(c)$

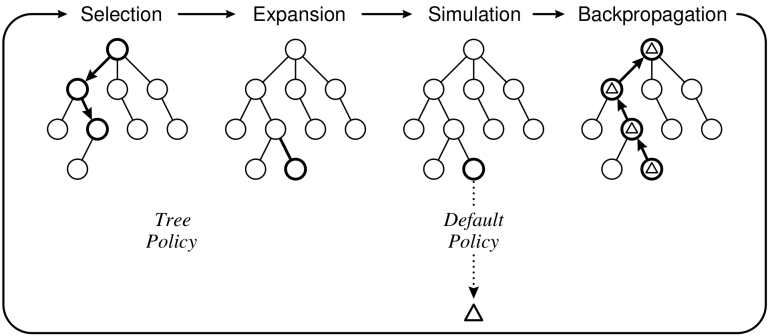
Note: some annotations can be computed on the fly to save memory

Framework

Trials

- The MCTS tree is built in trials
- Trials are performed as long as resources (deliberation time, memory) allow
- Initially, the MCTS tree consists of only the root node for the initial state
- Trials (may) add search nodes to the tree
- MCTS tree at the end of the i-th trial is denoted with $\mathcal{G}^i$
- Use same superscript for annotations of search nodes

Trials



Taken from Browne et al., “A Survey of Monte Carlo Tree Search Methods”, 2012

Phases of Trials

Each trial consists of (up to) four phases:

- Selection: traverse the tree by sampling the execution of the tree policy until
    1. an action is applicable that is not explicated, or
    2. an outcome is sampled that is not explicated, or
    3. a goal state is reached (jump to backpropagation)
- Expansion: create search nodes for the applicable action and a sampled outcome (case 1) or just the outcome (case 2)
- Simulation: simulate default policy until a goal is reached
- Backpropagation: update visited nodes in reverse order by
    - increasing visit counter by 1
    - performing Monte-Carlo backup of state-/action-value estimate

Monte-Carlo Backups in MCTS Tree

- let $d_0, c_0, \ldots, c_{n-1}, d_n$ be the decision and chance nodes that were visited in a trial of MCTS (including explicated ones),
- let h be the cost incurred by the simulation of the default policy until a goal state is reached
- each decision node d_j for $0 \leq j \leq n$ is updated by

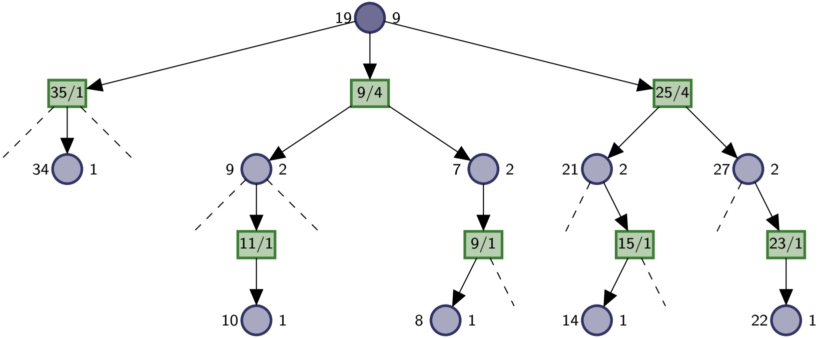
$$\hat{V}^i(d_j) := \hat{V}^{i-1}(d_j) + \frac{1}{N^i(d_j)}\left(\sum_{k=j}^{n-1} cost(a(c_k)) + h - \hat{V}^{i-1}(d_j)\right)$$

- each chance node c_j for $0 \leq j < n$ is updated by

$$\hat{Q}^i(c_j) := \hat{Q}^{i-1}(c_j) + \frac{1}{N^i(c_j)}\left(\sum_{k=j}^{n-1} cost(a(c_k)) + h - \hat{Q}^{i-1}(c_j)\right)$$

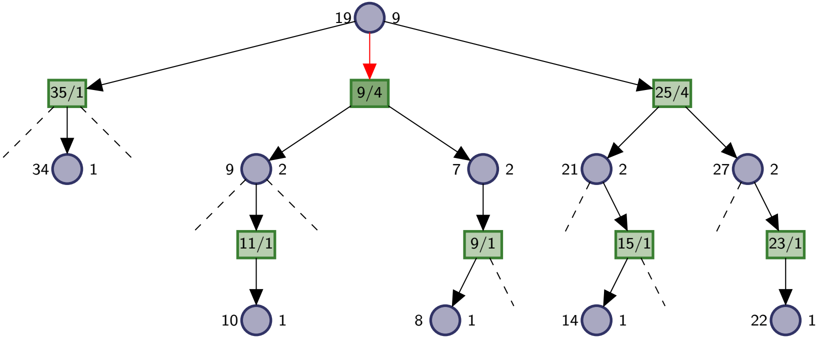
MCTS: (Unit-cost) Example

Selection phase: apply tree policy to traverse tree



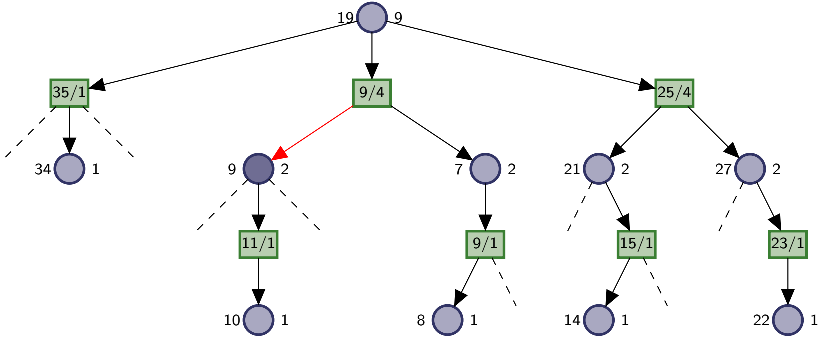
MCTS: (Unit-cost) Example

Selection phase: apply tree policy to traverse tree



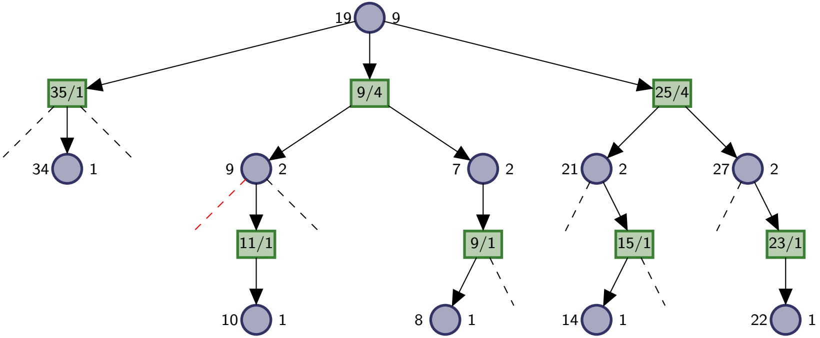
MCTS: (Unit-cost) Example

Selection phase: apply tree policy to traverse tree



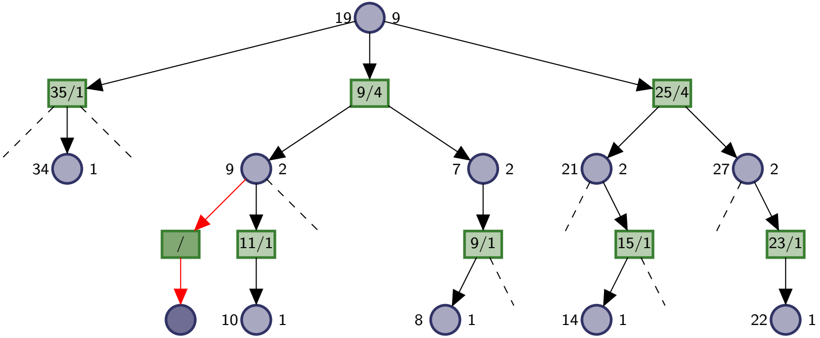
MCTS: (Unit-cost) Example

Selection phase: apply tree policy to traverse tree



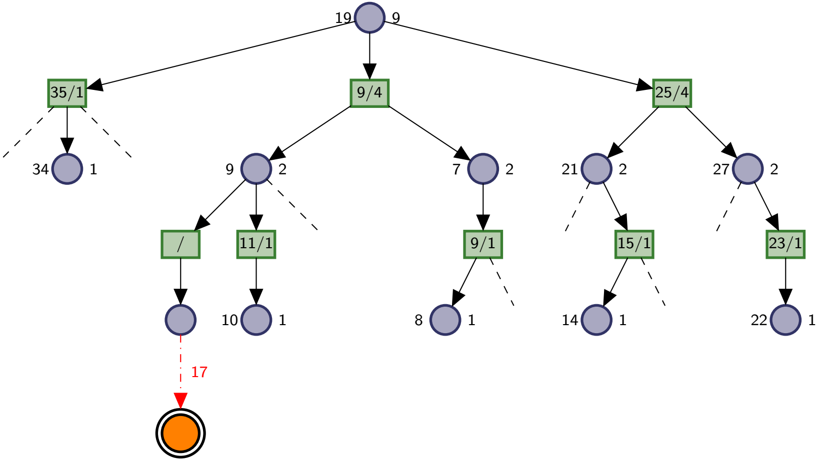
MCTS: (Unit-cost) Example

Expansion phase: create search nodes



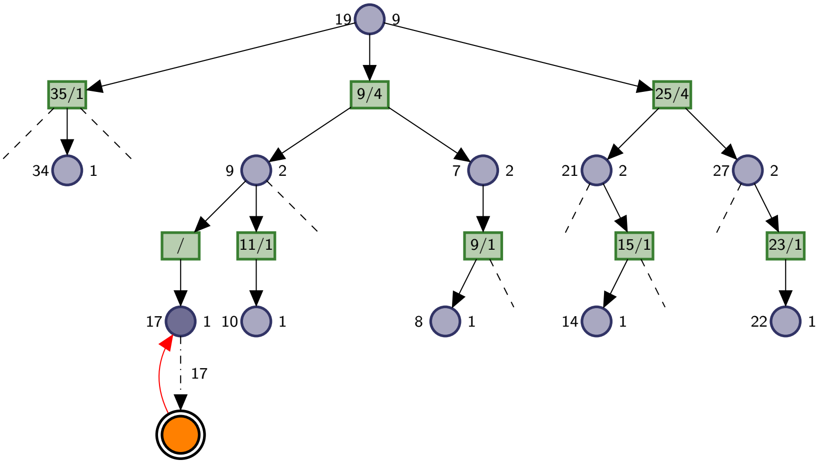
MCTS: (Unit-cost) Example

Simulation phase: apply default policy until goal



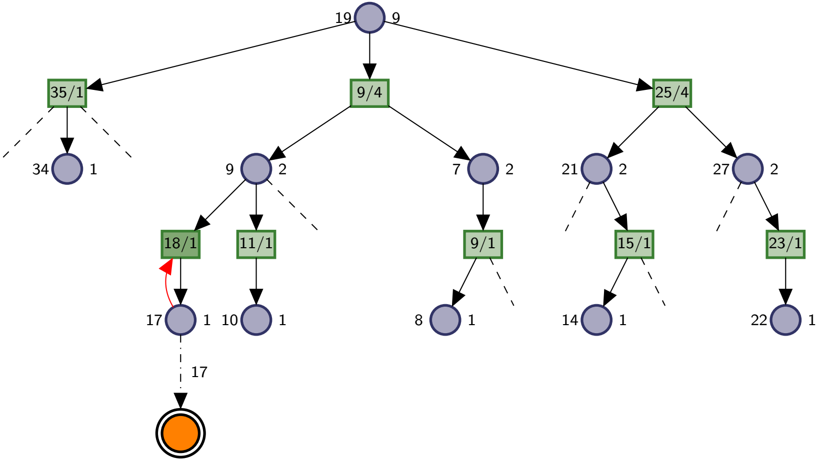
MCTS: (Unit-cost) Example

Backpropagation phase: update visited nodes



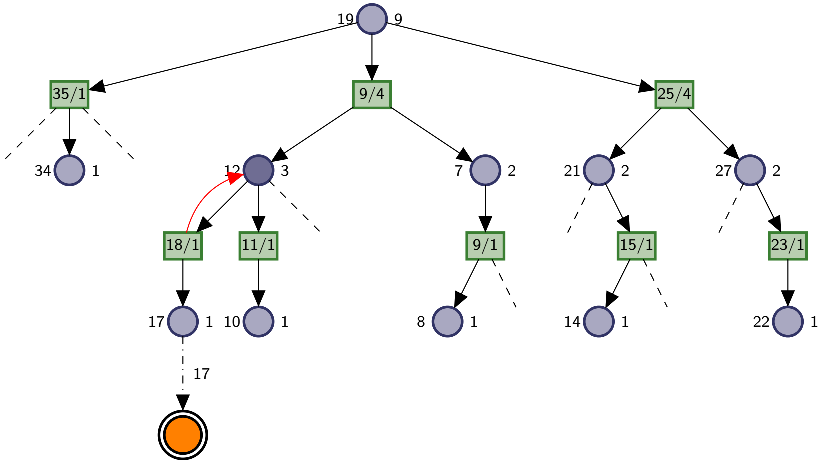
MCTS: (Unit-cost) Example

Backpropagation phase: update visited nodes



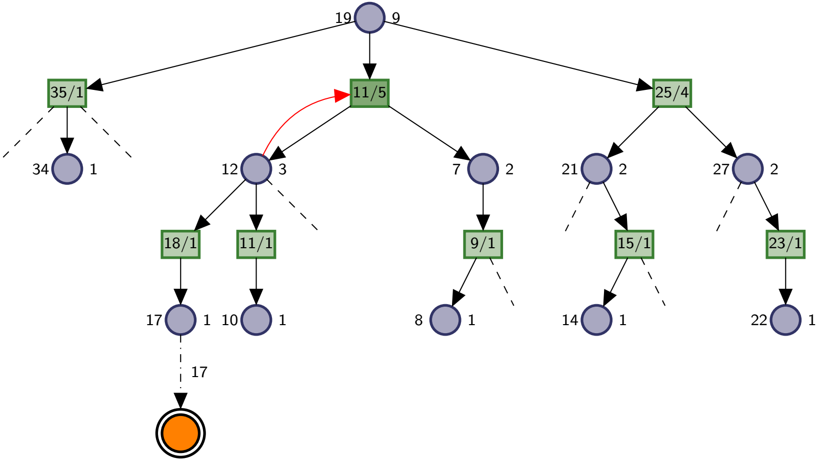
MCTS: (Unit-cost) Example

Backpropagation phase: update visited nodes



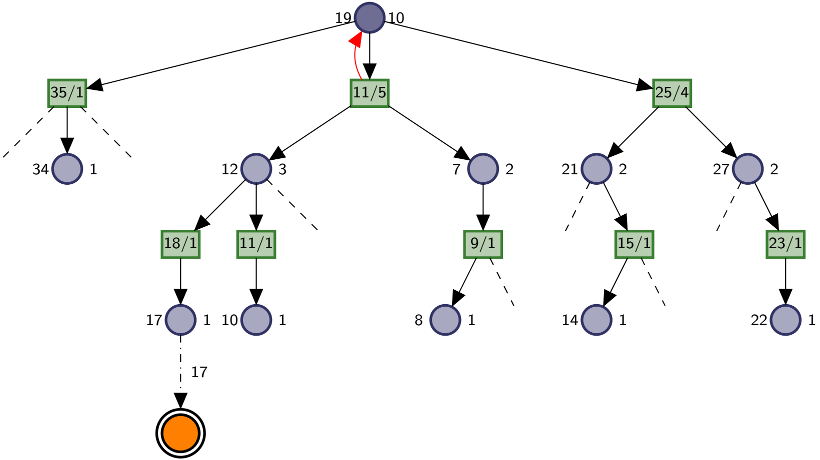
MCTS: (Unit-cost) Example

Backpropagation phase: update visited nodes



MCTS: (Unit-cost) Example

Backpropagation phase: update visited nodes



MCTS Framework

Member of MCTS framework are specified in terms of:

- Tree policy
- Default policy

MCTS Tree Policy

Definition (Tree Policy)

Let $\mathcal{T}$ be an SSP. An MCTS tree policy is a probability distribution $\pi(a \mid d)$ over all $a \in A(s(d))$ for each decision node d.

Note: The tree policy may take information annotated in the current tree into account.

MCTS Default Policy

Definition (Default Policy)

Let $\mathcal{T}$ be an SSP. An MCTS default policy is a probability distribution $\pi(a \mid s)$ over actions $a \in A(s)$ for each state s.

Note: The default policy is independent of the MCTS tree.

Monte-Carlo Tree Search

MCTS for SSP $\mathcal{T} = \langle S, A, c, T, s_0, S_\star \rangle$

d_0 = create root node associated with s_0
while time allows:
    visit_decision_node($d_0, \mathcal{T}$)
return $a(\arg\min_{c \in \text{children}(d_0)} \hat{Q}(c))$

MCTS: Visit a Decision Node

visit_decision_node for decision node d, SSP $\mathcal{T} = \langle S, A, c, T, s_0, S_\star \rangle$

if $s(d) \in S_\star$ **then return** 0
if there is $a \in A(s(d))$ s.t. $a(c) \neq a$ for all $c \in$ children(d):
    select such an a and add node c with $a(c) = a$ to children(d)
else:
    c = tree_policy(d)
cost = visit_chance_node($c, \mathcal{T}$)
$N(d) := N(d) + 1$
$\hat{V}(d) := \hat{V}(d) + \frac{1}{N(d)} \cdot (\text{cost} - \hat{V}(d))$
return cost

MCTS: Visit a Chance Node

visit_chance_node for chance node c, SSP $\mathcal{T} = \langle S, L, c, T, s_0, S_\star \rangle$

$s' \sim \text{succ}(s(c), a(c))$
let d be the node in children(c) with $s(d) = s'$
if there is no such node:
  add node d with $s(d) = s'$ to children(c)
  cost $=$ sample_default_policy(s')
  $N(d) := 1$, $\hat{V}(d) :=$ cost
else:
  cost $=$ visit_decision_node($d, \mathcal{T}$)
cost $=$ cost $+$ $cost(s(c), a(c))$
$N(c) := N(c) + 1$
$\hat{Q}(c) := \hat{Q}(c) + \frac{1}{N(c)} \cdot (\text{cost} - \hat{Q}(c))$
return cost

Summary

Summary

- Monte-Carlo Tree Search is a framework for algorithms
- MCTS algorithms perform trials
- Each trial consists of (up to) 4 phases
- MCTS algorithms are specified by two policies:
  - a tree policy that describes behavior "in" tree
  - and a default policy that describes behavior "outside" of tree