Foundations of Artificial Intelligence

44. Board Games: Advanced Topics in Monte-Carlo Tree Search

Malte Helmert

University of Basel

May 23, 2022

Board Games: Overview

chapter overview:

- 40. Introduction and State of the Art
- 41. Minimax Search and Evaluation Functions
- 42. Alpha-Beta Search
- 43. Introduction to Monte-Carlo Tree Search
- 44. Advanced Topics in Monte-Carlo Tree Search
- 45. AlphaGo and Outlook

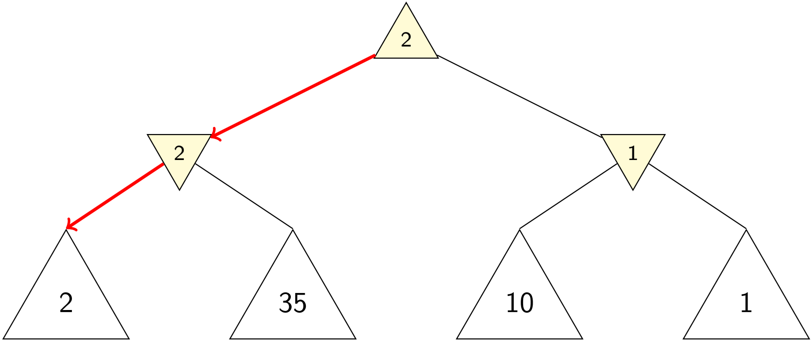
Optimality of MCTS

Reminder: Monte-Carlo Tree Search

- as long as time allows, perform iterations
  - selection: traverse tree
  - expansion: grow tree
  - simulation: play game to final position
  - backpropagation: update utility estimates
- execute move with highest utility estimate

Optimality

complete “minimax tree” computes optimal utility values u^*



Asymptotic Optimality

Asymptotic Optimality

An MCTS algorithm is asymptotically optimal if $\hat{u}^k(n)$ converges to optimal utility $u^*(n)$ for all $n \in \text{succ}(n_0)$ with $k \to \infty$.

Asymptotic Optimality

A tree policy is asymptotically optimal if

- it explores forever:
  - every position is expanded eventually and visited infinitely often (given that the game tree is finite)
  - after a finite number of iterations, only true utility values are used in backups
- and it is greedy in the limit:
  - the probability that an optimal move is selected converges to 1

  $\rightsquigarrow$ in the limit, backups based on iterations where only an optimal policy is followed dominate suboptimal backups

Tree Policy

Objective

tree policies have two contradictory objectives:

- explore parts of the game tree that have not been investigated thoroughly
- exploit knowledge about good moves to focus search on promising areas

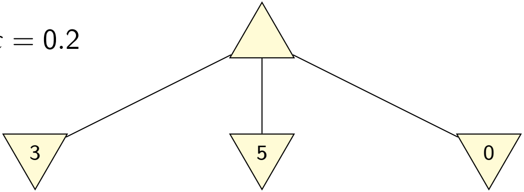
central challenge: balance exploration and exploitation

ε-greedy: Idea

- tree policy with constant parameter ε
- with probability $1 - \varepsilon$, pick a greedy move (i.e., one that leads to a successor node with the best utility estimate)
- otherwise, pick a non-greedy successor uniformly at random

ε-greedy: Example

$\varepsilon = 0.2$



$\mathbb{P}(n_1) = 0.1$ $\mathbb{P}(n_2) = 0.8$ $\mathbb{P}(n_3) = 0.1$

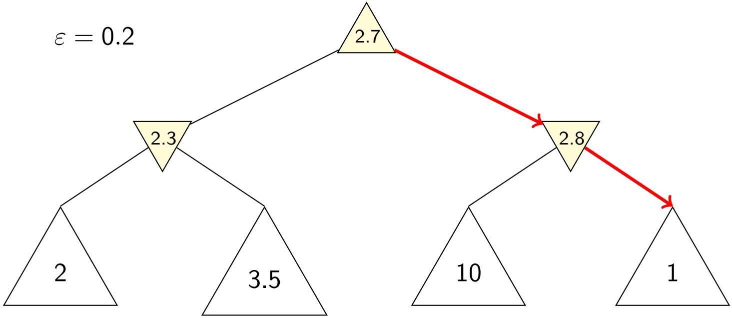
ε-greedy: Asymptotic Optimality

Asymptotic Optimality of ε-greedy

- explores forever
- not greedy in the limit

$\rightsquigarrow$ not asymptotically optimal

$\varepsilon = 0.2$



ε-greedy: Asymptotic Optimality

Asymptotic Optimality of ε-greedy

- explores forever
- not greedy in the limit

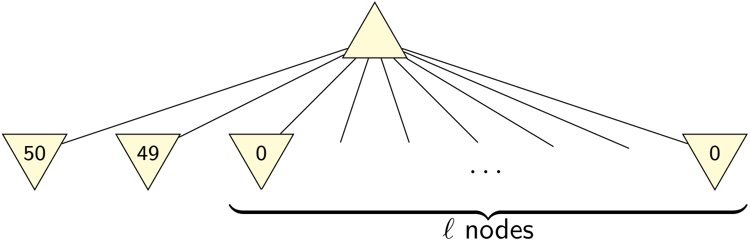
$\rightsquigarrow$ not asymptotically optimal

asymptotically optimal variants:

- use decaying ε, e.g. $\varepsilon = \frac{1}{k}$
- use minimax backups

ε-greedy: Weakness

Problem:
when ε-greedy explores, all non-greedy moves are treated equally

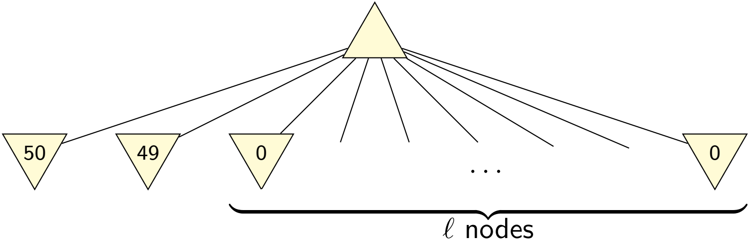


e.g., $\varepsilon = 0.2, \ell = 9$: $\mathbb{P}(n_1) = 0.8$, $\mathbb{P}(n_2) = 0.02$

Softmax: Idea

- tree policy with constant parameter $\tau > 0$
- select moves with a frequency that directly relates to their utility estimate
- Boltzmann exploration selects moves proportionally to $\mathbb{P}(n) \propto e^{\frac{\hat{u}(n)}{\tau}}$ for MAX nodes ($\mathbb{P}(n) \propto e^{\frac{-\hat{u}(n)}{\tau}}$ for MIN nodes)

Softmax: Example



e.g., $\tau = 10, \ell = 9$: $\mathbb{P}(n_1) \approx 0.51$, $\mathbb{P}(n_2) \approx 0.46$

Boltzmann Exploration: Asymptotic Optimality

Asymptotic Optimality of Boltzmann Exploration

- explores forever
- not greedy in the limit
  (probabilities converge to positive constant)

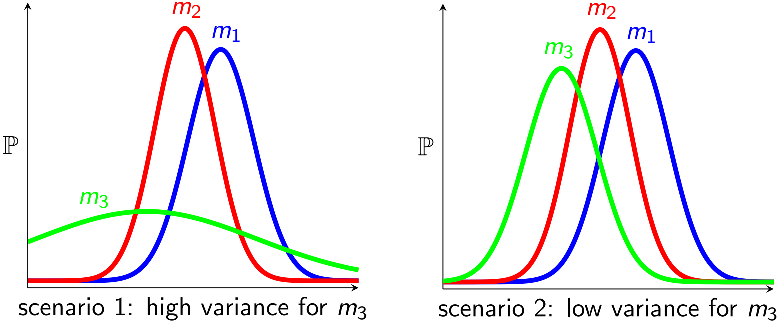
$\rightsquigarrow$ not asymptotically optimal

asymptotically optimal variants:

- use decaying τ
- use minimax backups

careful: τ must not decay faster than logarithmically
(i.e., must have $\tau \geq \frac{\text{const}}{\log k}$) to explore infinitely

Boltzmann Exploration: Weakness



- Boltzmann exploration only considers mean of sampled utilities for the given moves
- as we sample the same node many times, we can also gather information about variance (how reliable the information is)
- Boltzmann exploration ignores the variance, treating the two scenarios equally

Upper Confidence Bounds: Idea

balance exploration and exploitation by preferring moves that

- have been successful in earlier iterations (exploit)
- have been selected rarely (explore)

Upper Confidence Bounds: Idea

Upper Confidence Bounds

for MAX nodes:

- select successor n' of n that maximizes $\hat{u}(n') + B(n')$
- based on utility estimate $\hat{u}(n')$
- and a bonus term $B(n')$
- select $B(n')$ such that $u^*(n') \leq \hat{u}(n') + B(n')$ with high probability
- idea: $\hat{u}(n') + B(n')$ is an upper confidence bound on $u^*(n')$ under the collected information

(analogous for MIN nodes)

Upper Confidence Bounds: UCB1

- use $B(n') = \sqrt{\frac{2 \cdot \ln N(n)}{N(n')}}$ as bonus term
- bonus term is derived from Chernoff-Hoeffding bound, which
  - gives the probability that a sampled value (here: $\hat{u}(n')$)
  - is far from its true expected value (here: $u^*(n')$)
  - in dependence of the number of samples (here: $(N(n')$)
- picks the optimal move exponentially more often

Upper Confidence Bounds: Asymptotic Optimality

Asymptotic Optimality of UCB1

- explores forever
- greedy in the limit

$\rightsquigarrow$ asymptotically optimal

Upper Confidence Bounds: Asymptotic Optimality

Asymptotic Optimality of UCB1

- explores forever
- greedy in the limit

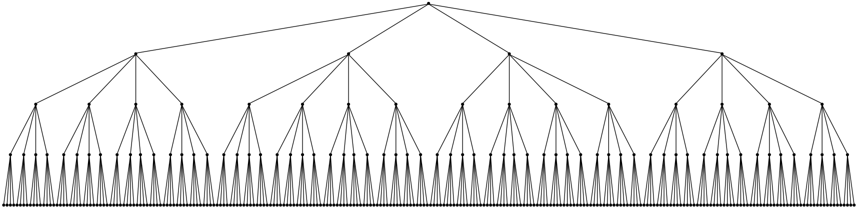
$\rightsquigarrow$ asymptotically optimal

However:

- no theoretical justification to use UCB1 in trees or planning scenarios
- development of tree policies active research topic

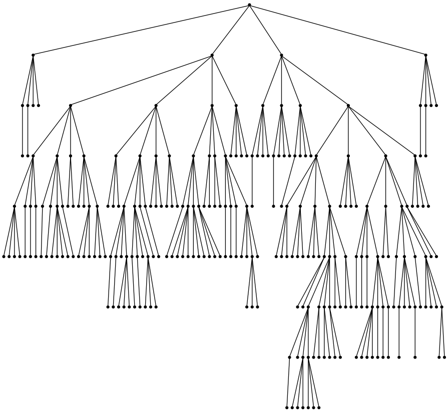
Tree Policy: Asymmetric Game Tree

full tree up to depth 4



Tree Policy: Asymmetric Game Tree

UCT tree (equal number of search nodes)



Other Techniques

Default Policy: Instantiations

default: Monte-Carlo Random Walk

- in each state, select a legal move uniformly at random
- very cheap to compute
- uninformed
- usually not sufficient for good results

alternative: domain-dependent default policy

- hand-crafted or
- function learned offline

Default Policy: Alternative

- default policy simulates a game to obtain utility estimate

$\rightsquigarrow$ default policy must be evaluated in many positions

- if default policy is expensive to compute, simulations are expensive
- solution: replace default policy with heuristic that computes a utility estimate directly

Expansion

- to proceed deeper into the tree, each node must be visited at least once for each legal move

$\rightsquigarrow$ deep lookaheads not possible when branching factor is high and resources are limited

- rather than add a single node, expand encountered leaf node and add all successors
  - allows deep lookaheads
  - needs more memory
  - needs initial utility estimate for all children

Summary

Summary

- tree policy is crucial for MCTS
  - ϵ-greedy favors greedy moves and treats all others equally
  - Boltzmann exploration selects moves proportionally to an exponential function of their utility estimates
  - UCB1 favors moves that were successful in the past or have been explored rarely
- for each, there are applications where they perform best
- good default policies are domain-dependent and hand-crafted or learned offline
- using heuristics instead of a default policy often pays off