CSCE 631 — Algorithmic Game Theory

Lecture 14: MCTS and Sampling-based CFR

Alan Kuhnle

Summer 2026

Texas A&M University

Today's Plan

Monte Carlo Tree Search (MCTS)

Challenges in Imperfect Information

Monte Carlo CFR (MCCFR)

Sampling Variants in Detail

Variance Reduction

Convergence Guarantees

Pruning in CFR

Connection to Deep CFR

Summary

Monte Carlo Tree Search (MCTS)

MCTS: Overview

Setting

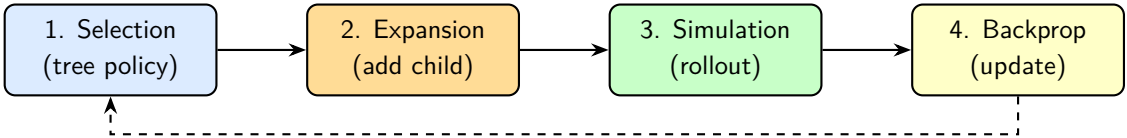
MCTS is a best-first search algorithm for decision-making in large game trees where exhaustive search (minimax / backward induction) is infeasible.

Core idea: build a partial search tree incrementally by sampling trajectories (simulations) from the current state.

1. Focus computation on the most promising parts of the tree
2. Use random playouts to estimate node values
3. Gradually improve estimates as more simulations run

Landmark result: MCTS (with UCT) was the key breakthrough enabling AlphaGo and superhuman Go play.

The Four Phases of MCTS



1. **Selection:** from root, traverse existing tree using a tree policy (e.g., UCB1) until a leaf or unexpanded node is reached
2. **Expansion:** add one (or more) child nodes to the tree
3. **Simulation:** from the new node, run a random playout (rollout) to a terminal state
4. **Backpropagation:** propagate the result back up the tree, updating visit counts and value estimates

UCB1 and UCT

UCB1 (Upper Confidence Bound)

At a node with children $\{c_1, \ldots, c_k\}$, select the child maximizing:

$$\text{UCB1}(c_j) = \bar{X}_j + C\sqrt{\frac{\ln N}{n_j}}$$

where $\bar{X}_j$ = average reward at child j, n_j = visits to j, N = visits to parent, C = exploration constant.

UCT (UCB applied to Trees, Kocsis and Szepesvári, 2006)

Apply UCB1 as the tree policy in MCTS. At each internal node, select the child with highest UCB1 score.

- UCT balances **exploitation** ($\bar{X}_j$) and **exploration** ($\sqrt{\ln N / n_j}$)

MCTS Pseudocode

Algorithm: MCTS with UCT

Input: Root state s_0, computational budget B

Output: Best action from s_0

while budget remaining **do**

$v \leftarrow s_0$

// *Selection*

while v is fully expanded and non-terminal **do**

$v \leftarrow \arg\max_{c \in \text{children}(v)} \text{UCB1}(c)$

end while

// *Expansion:* if v non-terminal, add an unexpanded child v'

// *Simulation:* run random playout from v', get reward r

// *Backpropagation:* update $\bar{X}$ and n from v' to root

end while

MCTS in Perfect-Information Games

- MCTS has been enormously successful for **perfect-information** games:
  - Go (AlphaGo, AlphaZero)
  - Chess, shogi (AlphaZero)
  - General game playing
- **Why it works:** in perfect-information games, the tree structure is "clean" — each node corresponds to a unique game state
- UCT converges to the minimax value with enough simulations
- With neural network value/policy estimates replacing random rollouts: dramatic improvements (AlphaGo Zero, 2017)

Key Question

Can MCTS handle **imperfect-information** games?

Challenges in Imperfect Information

Why MCTS Struggles with Imperfect Information

- In imperfect-information games, nodes within the same information set are **strategically equivalent** — the player cannot distinguish them
- Naive MCTS treats each node independently $\Rightarrow$ fails to respect information constraints
- **Strategy fusion problem:** MCTS may learn different strategies at nodes in the same information set, violating the constraint that the player must play identically at all of them

Determinization (a tempting but flawed approach)

"Sample a possible world (e.g., opponents' cards), solve the perfect-information game, aggregate." This ignores the *informational* aspects: a strategy in one world may reveal information usable in another. Results can be arbitrarily bad.

Information Set MCTS (IS-MCTS)

Idea: modify MCTS to operate on *information sets* rather than individual nodes.

- Maintain statistics (visit counts, values) at the *information set* level
- Selection policy uses information-set statistics
- Requires careful handling of opponent and chance moves

Approaches:

1. **ISMCTS** (Cowling et al., 2012): sample determinizations but aggregate at information-set level
2. **Smooth UCT:** average over all plausible game states in an information set
3. **Better approach:** combine MCTS ideas with CFR $\Rightarrow$ MCCFR

Monte Carlo CFR (MCCFR)

From Vanilla CFR to Monte Carlo CFR

Vanilla CFR: traverses the *entire* game tree each iteration.

- Cost per iteration: $O(|V|)$
- For Texas Hold'em: $|V| \approx 10^{17}$ — a single full traversal is impossible

Monte Carlo CFR (MCCFR): sample only a subset of the tree each iteration.

- Compute *sampled* counterfactual values
- Update regrets only at visited information sets
- Unbiased estimates $\Rightarrow$ same convergence guarantee (in expectation)
- Cost per iteration: much less than $|V|$

Key Trade-off

MCCFR: cheaper iterations but noisier updates. Converges in more iterations but less wall-clock time.

MCCFR Sampling Schemes

Three main variants, differing in *what gets sampled*:

Variant	**What is sampled?**	**Cost/iter**	**Variance**
Chance sampling	Only chance outcomes	$O(\|V\|/\|\text{chance}\|)$	Low
External sampling	Chance + opponent actions	$O(\|V_i\|)$	Medium
Outcome sampling	Chance + opponent + own actions (single trajectory)	$O(\text{depth})$	High

All three produce **unbiased** estimates of counterfactual values, so all converge to NE in two-player zero-sum games.

Sampling Variants in Detail

Chance Sampling

- Sample a single outcome at each **chance node** (e.g., sample one card deal)
- Traverse the full subtree for that chance outcome
- All player actions are explored exhaustively

Cost per iteration: $O(|V|/|\text{chance outcomes}|)$

- For poker: if there are C possible deals, each iteration costs $|V|/C$
- Kuhn poker: 6 deals, saves $6\times$
- Texas Hold'em: $\sim 10^9$ deals, massive savings

Variance: relatively low because all player actions are explored. Payoff variance comes only from the chance sampling.

External Sampling

- Sample chance outcomes *and* the opponent's actions (according to current opponent strategy)
- The traversing player explores **all** their own actions at each of their information sets
- Only one action is sampled for the opponent at each of their info sets

Cost per iteration: proportional to the number of nodes where the traversing player acts — roughly $|V_i|$

External Sampling in Practice

External sampling is the most commonly used MCCFR variant. It offers a good balance of low cost per iteration and manageable variance. Used in Libratus and Pluribus.

Outcome Sampling

- Sample a **single trajectory** from root to leaf:
  - Sample chance outcomes
  - Sample opponent actions (from current strategy)
  - Sample own actions (from a *sampling distribution* q, which may differ from the current strategy)
- Only the information sets along this trajectory are updated

Cost per iteration: $O(d)$ where d = depth of the tree.

Importance weighting: since we sample our own actions from q instead of σ_i, we apply importance weights:

$$\hat{v}_i(I, a) = \frac{\pi^{\sigma}_{-i}(h) \cdot u_i(z)}{q(z)} \cdot \mathbf{1}[\text{action } a \text{ was played at } I]$$

Variance: high due to single-trajectory sampling and importance weights.

Variance Reduction

Variance Reduction Techniques

High variance in sampling-based CFR slows convergence. Key techniques:

1. **Baseline subtraction:** subtract a control variate (baseline) from the sampled value:
$$\hat{v}_i(I) - b(I)$$
Choose $b(I)$ to correlate with $\hat{v}_i(I)$ while having known expectation.
2. **Stochastic-weight averaging:** instead of updating with full importance weight, clip or dampen extreme weights.
3. **Exploration policy q:** mix the current strategy with uniform exploration:
$$q(a \mid I) = \epsilon \cdot \frac{1}{|A(I)|} + (1 - \epsilon) \cdot \sigma_i(a \mid I)$$
Ensures all actions are sampled with minimum probability $\epsilon/|A(I)|$.
4. **Robust sampling** (Schmid et al., 2019): optimizes the sampling distribution to

Probing and Partial Tree Traversal

Probing: at some information sets, instead of sampling one action, evaluate a few "probe" actions and compute a partial expectation.

- Interpolates between outcome sampling (one action) and external sampling (all actions)
- Adjustable cost/variance trade-off

Depth-limited solving:

- Only traverse the tree to a fixed depth d
- At depth d, estimate leaf values using a *value function* (learned offline or from coarser abstraction)
- Used in real-time subgame solving (Libratus, Pluribus)

Convergence Guarantees

Convergence of MCCFR Variants

Theorem (Lanctot et al., 2009)

All MCCFR variants (chance, external, outcome sampling) converge to a Nash equilibrium in two-player zero-sum games. Specifically, the expected overall regret satisfies:

$$\mathbb{E}[R_i^T] \leq \sum_{I \in \mathcal{I}_i} \frac{\Delta_i(I)\sqrt{|A(I)|}}{\sqrt{T}} \cdot C_{\text{var}}$$

where C_{var} depends on the sampling scheme.

Variant	Rate	Variance factor
Vanilla CFR	$O(1/\sqrt{T})$	None (deterministic)
Chance sampling	$O(1/\sqrt{T})$	Small
External sampling	$O(1/\sqrt{T})$	Moderate
Outcome sampling	$O(1/\sqrt{T})$	Large

Convergence: Theory vs. Practice

- All variants have the same *asymptotic* rate $O(1/\sqrt{T})$
- In practice, the constant factor matters enormously:
  - External sampling typically converges fastest per unit of computation
  - Outcome sampling needs many more iterations but each is very cheap
  - Chance sampling is a good default for moderate-sized games
- **Practical guideline:** choose the variant based on the game tree size and available compute:
  - Small tree ($< 10^6$ nodes): vanilla CFR
  - Medium tree (10^6–10^{10}): external sampling
  - Large tree ($> 10^{10}$): outcome sampling + variance reduction

Pruning in CFR

Pruning Techniques for CFR

Motivation: even with sampling, CFR can waste time on clearly suboptimal parts of the tree.

1. **Regret-based pruning (RBP)** (Brown and Sandholm, 2015):
   - Skip traversal of action a at info set I if $R_i^t(I, a) < -C$ for a large threshold C
   - Provably does not affect convergence (pruned actions would have zero probability under regret matching anyway)
   - Speedup: up to $10\times$ in later iterations
2. **Dynamic thresholding:** increase the pruning threshold as iterations progress (more aggressive pruning over time)
3. **Best-response pruning:** skip branches where the best response is already known to avoid that subtree

Regret-Based Pruning: Why It's Safe

Observation

If $R_i^t(I, a) \ll 0$, then regret matching assigns $\sigma_i^t(a \mid I) = 0$. The regret of action a can only increase by at most Δ per iteration. So if $R_i^t(I, a) < -D$, it will take at least D/Δ iterations before a can have positive regret.

- During those D/Δ iterations, we can safely skip the subtree under action a
- Periodically (every D/Δ iterations), unprune and recheck
- In practice: saves 70–90% of computation in later phases when the strategy is nearly converged

Connection to Deep CFR

Preview: Deep CFR

For extremely large games, even MCCFR with external sampling cannot store regrets for all information sets (too many).

Deep CFR (Brown et al., 2019) replaces the regret tables with *neural networks*:

- Train a network $V_\theta(I, a)$ to predict cumulative regret $R_i^T(I, a)$ from information-set features
- Train a network $\Pi_\phi(I, a)$ to predict the average strategy $\bar{\sigma}(a \mid I)$
- Each CFR iteration: sample trajectories (MCCFR), generate training data, update networks

Key Advantage

Generalizes across similar information sets via function approximation. No need to store separate regrets for each of 10^{14} info sets. We will cover this in detail in **Week**

The Landscape: Algorithms for Imperfect-Information Games

Method	Scale	Key Property
Sequence-form LP	Small–medium	Exact solution
Vanilla CFR	Medium	Full traversal, $O(1/\sqrt{T})$
CFR+	Medium–large	Faster convergence
MCCFR (external)	Large	Sampled traversal
MCCFR (outcome)	Very large	Single trajectory
Deep CFR	Massive	Neural function approx.

- Each method extends the tractable game size by orders of magnitude
- The progression mirrors the historical development (1996–2019)
- All rely on the sequence/realization-plan structure from Lecture 12

Summary

Summary

- **MCTS:** selection, expansion, simulation, backpropagation; UCB1/UCT balances exploration and exploitation; powerful for perfect-information games
- **Imperfect information challenge:** strategy fusion, information-set constraints; naive MCTS fails
- **Monte Carlo CFR:** samples portions of the tree instead of full traversal; unbiased regret estimates
- **Three variants:** chance sampling (sample deals), external sampling (sample opponent), outcome sampling (single trajectory)
- **Variance reduction:** baselines, exploration mixing, robust sampling; critical for outcome sampling
- **All variants converge** at $O(1/\sqrt{T})$ to NE in two-player zero-sum games
- **Pruning:** regret-based pruning safely skips subtrees with large negative regret, 70–90% speedup