Planning and Optimization

G5. Asymptotically Suboptimal Monte-Carlo Methods

Malte Helmert and Thomas Keller

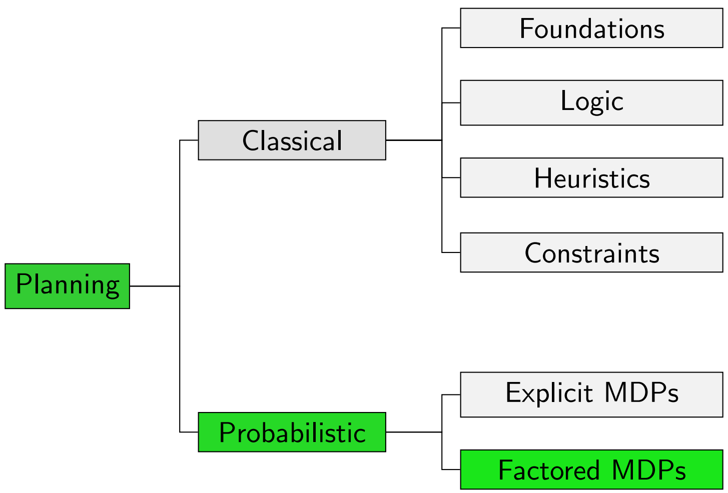
Universität Basel

December 11, 2019

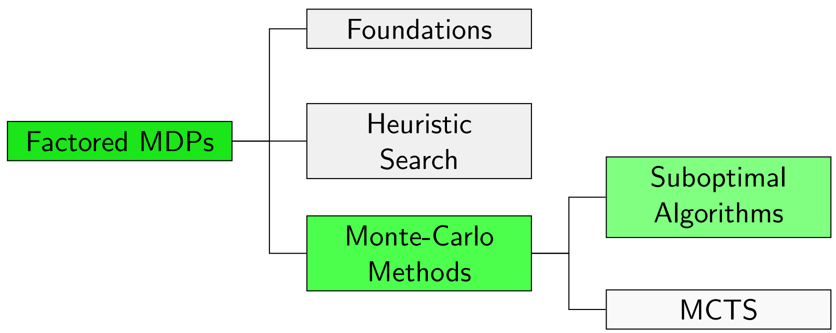
Planning and Optimization

December 11, 2019 — G5. Asymptotically Suboptimal Monte-Carlo Methods

Content of this Course



Content of this Course: Factored MDPs



G5.1 Motivation

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

G5.2 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 only 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, (L)AO* 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

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

(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 estimate state-values by averaging over all samples instead.

Let $N^i(s)$ be the number of samples for state s in the i first algorithm iterations and let $cost^k(s)$ be the cost for s in the k-th sample ($cost^k(s) = 0$ if 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

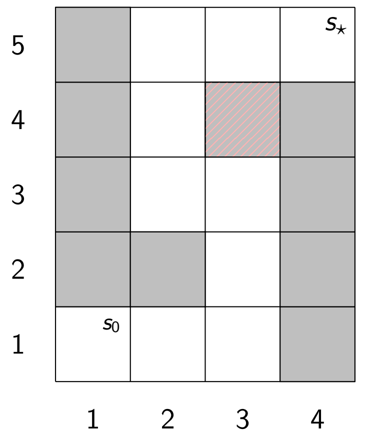
G5.3 Hindsight Optimization

Hindsight Optimization: Idea

Repeat as long as resources (deliberation time, memory) allow:

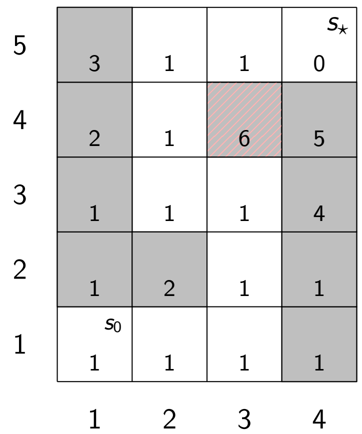
- Sample outcomes of all actions
  $\Rightarrow$ deterministic (classical) planning problem
- For each applicable action $\ell \in L(s_0)$,
  compute plan in the sample that starts with ℓ
- Execute the action with the lowest average plan cost

Hindsight Optimization: Example



- cost of 1 for all actions except for moving away from (3,4) where cost is 3
- get stuck when moving away from gray cells with prob. 0.6

Hindsight Optimization: Example



1st sample

- Samples can be described by number of times agent is stuck
- Multiplication with cost to move away from cell gives cost of leaving cell in sample

Hindsight Optimization: Example

	1	2	3	4
5	5	2	1	$s_\star$ 0
4	5	3	7	5
3	5	4	5	9
2	6	6	6	7
1	s_0 7	7	7	8

$C_1(s)$

- Samples can be described by number of times agent is stuck
- Multiplication with cost to move away from cell gives cost of leaving cell in sample

Hindsight Optimization: Example

	1	2	3	4
5	5	⇒ 2	⇒ 1	$s_\star$ 0
4	5	⇑ 3	7	5
3	⇒ 5	⇑ 4	5	9
2	⇑ 6	6	6	7
1	⇑ s_0 7	7	7	8

$\hat{V}^1(s)$

- Samples can be described by number of times agent is stuck
- Multiplication with cost to move away from cell gives cost of leaving cell in sample

Hindsight Optimization: Example

	1	2	3	4
5	1	1	1	$s_\star$ 0
4	6	1	6	1
3	5	1	1	5
2	3	4	1	1
1	s_0 1	1	1	1

2nd sample

- Samples can be described by number of times agent is stuck
- Multiplication with cost to move away from cell gives cost of leaving cell in sample

Hindsight Optimization: Example

	1	2	3	4
5	3	2	1	$s_\star$ 0
4	9	3	7	1
3	9	4	5	6
2	11	8	6	7
1	s_0 9	8	7	8

$C_2(s)$

- Samples can be described by number of times agent is stuck
- Multiplication with cost to move away from cell gives cost of leaving cell in sample

Hindsight Optimization: Example

5	4	$\Rightarrow$ 2	$\Rightarrow$ 1	$s_\star$ 0
4	7	$\Uparrow$ 3	7	3
3	7	$\Uparrow$ 4	5	7.5
2	8.5	$\Uparrow$ 7	6	7
1	$\Rightarrow^{s_0}$ 8	$\Uparrow$ 7.5	7	8
	1	2	3	4

$\hat{V}^2(s)$

- Samples can be described by number of times agent is stuck
- Multiplication with cost to move away from cell gives cost of leaving cell in sample

Hindsight Optimization: Example

5	4.0	$\Rightarrow$ 2.0	$\Rightarrow$ 1.0	$s_\star$ 0
4	6.3	$\Uparrow$ 3.0	8.8	1.8
3	6.5	$\Uparrow$ 4.0	4.3	4.7
2	7.0	$\Uparrow$ 5.6	5.3	7.2
1	$\Rightarrow^{s_0}$ 7.2	$\Uparrow$ 6.3	6.3	8.3
	1	2	3	4

$\hat{V}^{10}(s)$

- Samples can be described by number of times agent is stuck
- Multiplication with cost to move away from cell gives cost of leaving cell in sample

Hindsight Optimization: Example

5	4.55	$\Rightarrow$ 2.0	$\Rightarrow$ 1.0	$s_\star$ 0
4	5.43	$\Uparrow$ 3.0	8.50	2.40
3	6.57	$\Uparrow$ 4.0	$\Leftarrow$ 4.51	4.99
2	8.22	6.69	$\Uparrow$ 5.51	7.16
1	$\Rightarrow^{s_0}$ 7.69	$\Rightarrow$ 6.89	$\Uparrow$ 6.51	8.48
	1	2	3	4

$\hat{V}^{100}(s)$

- Samples can be described by number of times agent is stuck
- Multiplication with cost to move away from cell gives cost of leaving cell in sample

Hindsight Optimization: Example

5	4.58	$\Rightarrow$ 2.0	$\Rightarrow$ 1.0	$s_\star$ 0
4	5.56	$\Uparrow$ 3.0	8.33	2.44
3	6.54	$\Uparrow$ 4.0	4.49	4.84
2	7.88	$\Uparrow$ 6.48	5.49	6.80
1	$\Rightarrow^{s_0}$ 7.60	$\Uparrow$ 6.75	6.49	8.44
	1	2	3	4

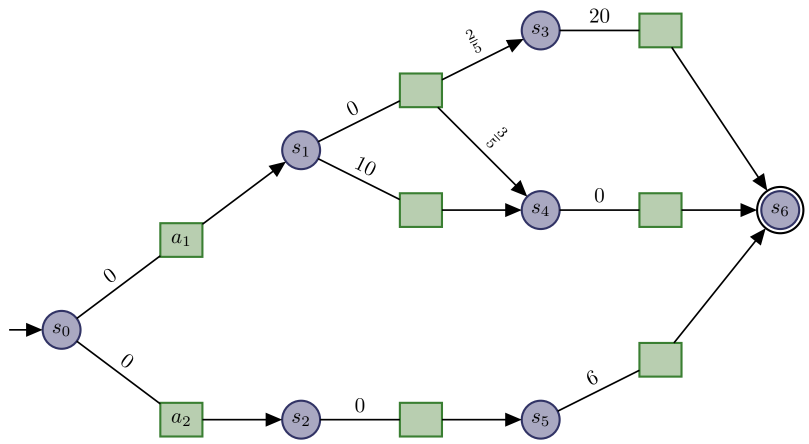
$\hat{V}^{1000}(s)$

- Samples can be described by number of times agent is stuck
- Multiplication with cost to move away from cell gives cost of leaving cell in sample

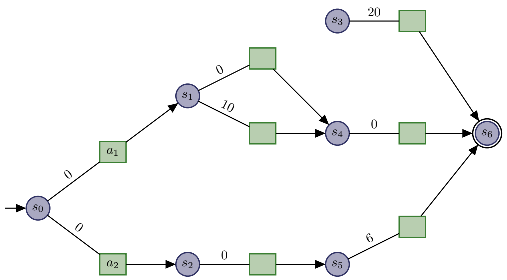
Hindsight Optimization: Evaluation

- HOP well-suited for some problems
- must be possible to solve sampled SSP efficiently:
  - domain-dependent knowledge (e.g., games like Bridge, Skat)
  - classical planner (FF-Hindsight, Yoon et. al, 2008)
- What about optimality in the limit?

Hindsight Optimization: Optimality in the Limit



Hindsight Optimization: Optimality in the Limit

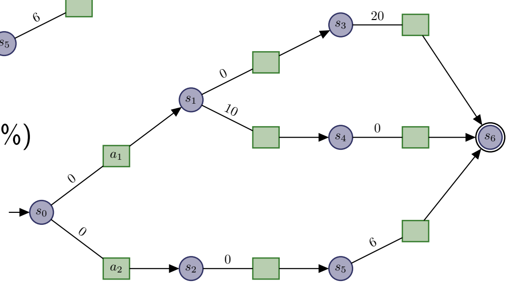


(sample probability: 60%)

with $k \to \infty$:

$\hat{Q}^k(s_0, a_1) \to 4$

$\hat{Q}^k(s_0, a_2) \to 6$



(sample probability: 40%)

Hindsight Optimization: Evaluation

- HOP well-suited for some problems
- must be possible to solve sampled MDP efficiently:
  - domain-dependent knowledge (e.g., games like Bridge, Skat)
  - classical planner (FF-Hindsight, Yoon et. al, 2008)
- What about optimality in the limit?
  $\Rightarrow$ in general not optimal due to assumption of clairvoyance

G5.4 Policy Simulation

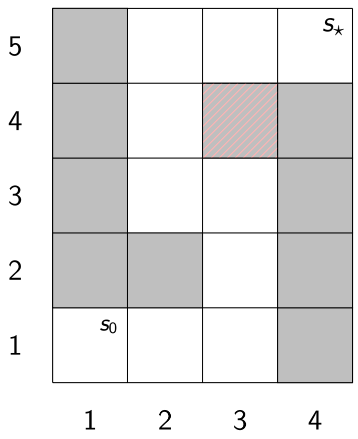
Policy Simulation: Idea

Repeat as long as resources (deliberation time, memory) allow:

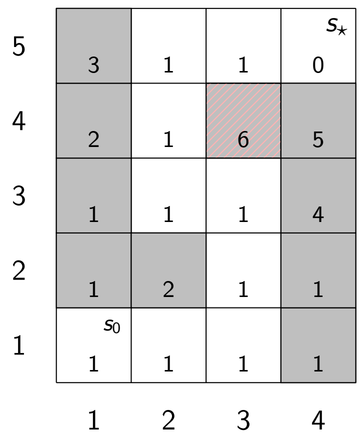
- For each applicable action $\ell \in L(s_0)$,
  start a run from s_0 with ℓ and then follow a given policy π
- Execute the action with the lowest average simulation cost

Avoids clairvoyance by evaluation of policy
through simulation of its execution.

Policy Simulation: Example (following Optimistic Policy)



Policy Simulation: Example (following Optimistic Policy)



1st sample

Policy Simulation: Example (following Optimistic Policy)

$C_1(s)$

	1	2	3	4
5	3	2	1	$s_\star$ 0
4	6	3	13	3
3	5	4	5	8
2	7	7	6	9
1	s_0 9	6	7	11

Policy Simulation: Example (following Optimistic Policy)

$\hat{V}^1(s)$

	1	2	3	4
5	3	$\Rightarrow$ 2	$\Rightarrow$ 1	$s_\star$ 0
4	6	$\Uparrow$ 3	13	3
3	5	$\Uparrow$ 4	5	8
2	7	$\Uparrow$ 7	6	9
1	s_0 $\Rightarrow$ 9	$\Uparrow$ 6	7	11

Policy Simulation: Example (following Optimistic Policy)

$\hat{V}^{10}(s)$

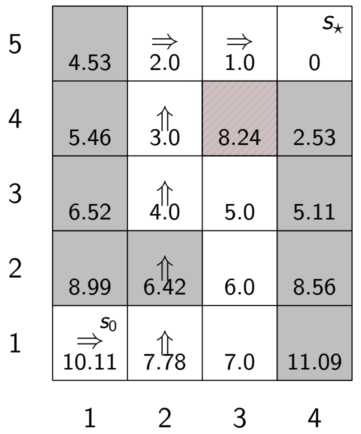
	1	2	3	4
5	4.6	$\Rightarrow$ 2.0	$\Rightarrow$ 1.0	$s_\star$ 0
4	5.5	$\Uparrow$ 3.0	8.2	2.2
3	7.6	$\Uparrow$ 4.0	5.0	5.4
2	9.0	$\Uparrow$ 6.8	6.0	8.8
1	s_0 $\Rightarrow$ 9.3	$\Uparrow$ 6.9	7.0	11.4

Policy Simulation: Example (following Optimistic Policy)

$\hat{V}^{100}(s)$

	1	2	3	4
5	4.55	$\Rightarrow$ 2.0	$\Rightarrow$ 1.0	$s_\star$ 0
4	5.54	$\Uparrow$ 3.0	8.42	2.37
3	6.52	$\Uparrow$ 4.0	5.0	5.13
2	9.2	$\Uparrow$ 6.69	6.0	8.43
1	s_0 $\Rightarrow$ 10.06	$\Uparrow$ 7.63	7.0	10.66

Policy Simulation: Example (following Optimistic Policy)



Policy Simulation: Evaluation

- Base policy is static
- No mechansim to overcome weaknesses of base policy (if there are no weaknesses, we don't need policy simulation)
- Suboptimal decisions in simulation affect policy quality
- What about optimality in the limit?
  $\Rightarrow$ in general not optimal due to inability of policy to improve

G5.5 Sparse Sampling

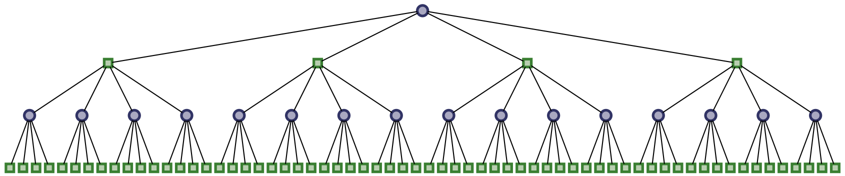
Sparse Sampling: Idea

Sparse Sampling (Kearns et al., 2002) approaches problem that number of reachable states under a policy can be too large

- Creates search tree up to a given lookahead horizon
- A constant number of outcomes is sampled for each state-action pair
- Outcomes that were not sampled are ignored
- Near-optimal: expected cost of resulting policy close to expected cost of optimal policy
- Runtime independent from the number of states

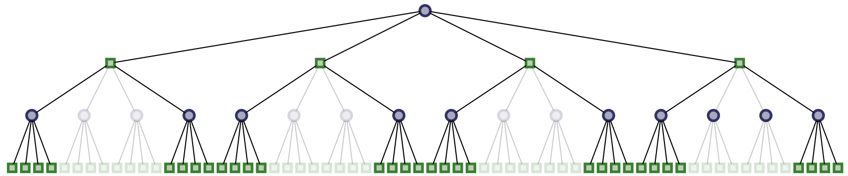
Sparse Sampling: Search Tree

Without Sparse Sampling



Sparse Sampling: Search Tree

With Sparse Sampling



Sparse Sampling: Problems

- Independent from number of states, but still exponential in lookahead horizon
- Constants that give number of outcomes and lookahead horizon large for good bounds on near-optimality
- Search time difficult to predict
- Search tree is symmetric
  $\Rightarrow$ resources are wasted in non-promising parts of the tree

G5.6 Summary

Summary

- Monte-Carlo methods have a long history but no successful applications until 1990s
- Monte-Carlo methods use sampling and backups that average over sample results
- Hindsight optimization averages over plan cost in sampled determinization
- Policy simulation simulates the exection of a policy
- Sparse sampling considers only a fixed amount of outcomes
- All three methods are not optimal in the limit