Planning and Optimization

G4. Asymptotically Suboptimal Monte-Carlo Methods

Gabriele Röger and Thomas Keller

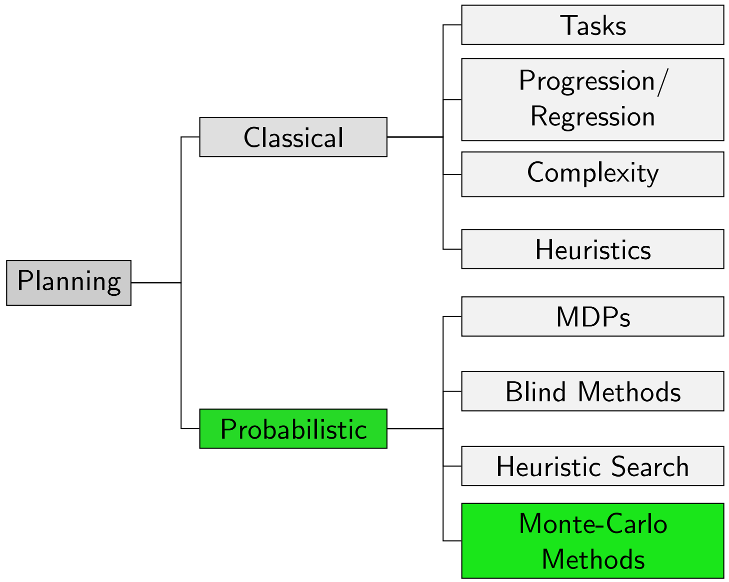
Universität Basel

December 5, 2018

Planning and Optimization

December 5, 2018 — G4. Asymptotically Suboptimal Monte-Carlo Methods

Content of this Course



G4.1 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

G4.2 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

Monte-Carlo Backups

- Algorithms presented so far used full Bellman backups to update state-value estimates:

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

- Monte-Carlo methods use Monte-Carlo backups instead:

$$\hat{V}^i(s) := \frac{1}{N(s)} \cdot \sum_{k=1}^{i} C_k(s), \text{where}$$

  - $N(s) \leq k$ is a counter for the number of state-value estimates for state s in first k algorithm iterations and
  - $C_k(s)$ is cost of k-th iteration for state s (assume $C_i(s) = 0$ for iterations without estimate for s)

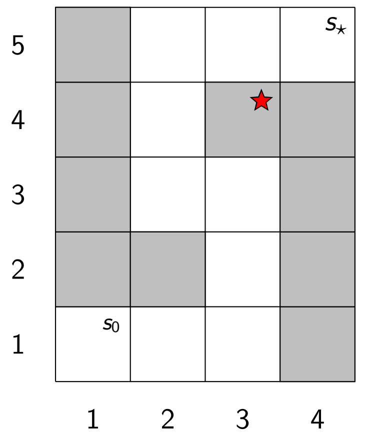
Advantage: no need to know SSP model, a simulator that samples successor states and reward is sufficient

G4.3 Hindsight Optimization

Hindsight Optimization: Idea

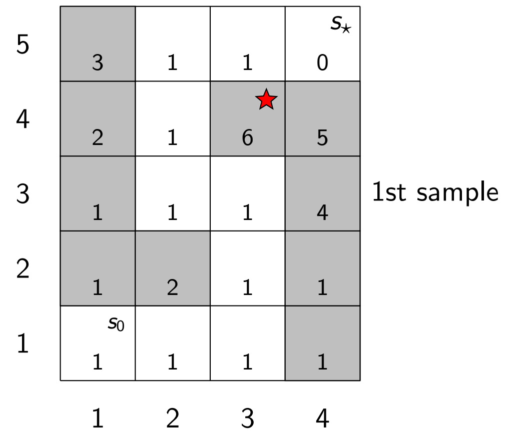
- Perform samples as long as resources (deliberation time, memory) allow
- Sample outcomes of all actions $\Rightarrow$ deterministic (classical) planning problem
- For each applicable action $\ell \in L(s_0)$, compute plan in the sample that starts with ℓ
- Execute the action with the lowest average 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



- 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

$C_1(s)$

	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

- 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

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

	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

- 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

2nd sample

	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

- 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

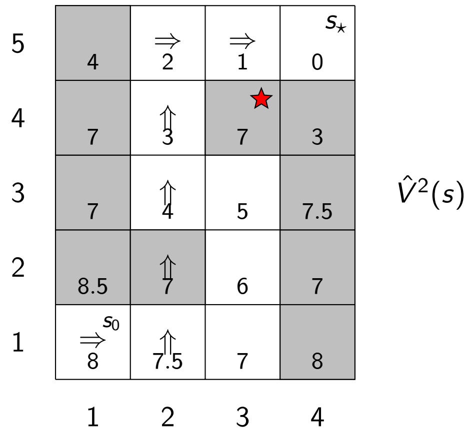
Hindsight Optimization: Example

$C_2(s)$

	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

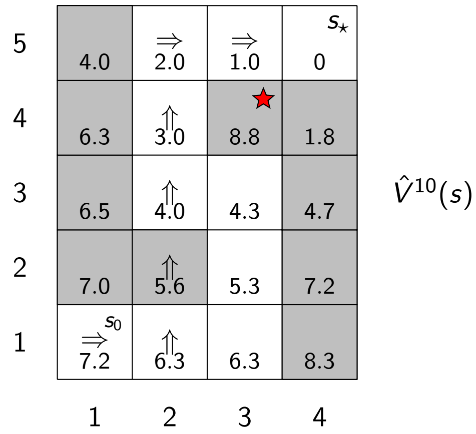
- Samples can be described by number of times agent is stuck
- Multiplication with cost to move away from cell gives cost of leaving cell in sample

Hindsight Optimization: Example



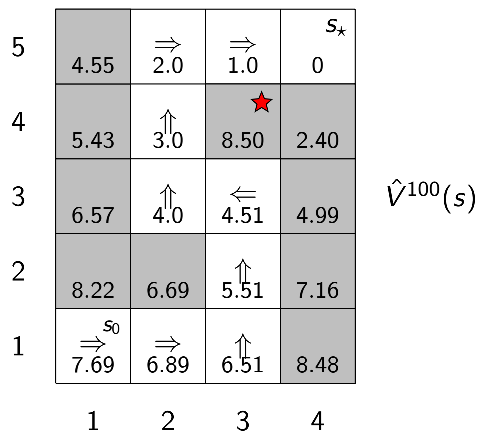
- Samples can be described by number of times agent is stuck
- Multiplication with cost to move away from cell gives cost of leaving cell in sample

Hindsight Optimization: Example



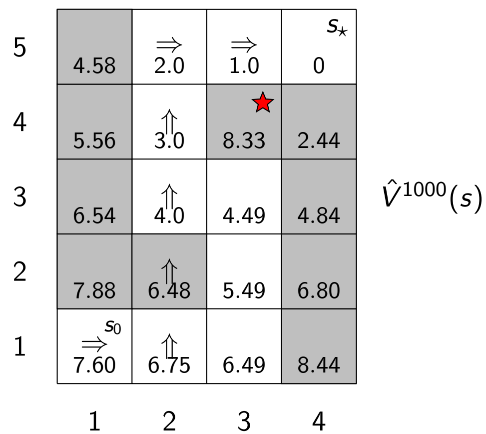
- Samples can be described by number of times agent is stuck
- Multiplication with cost to move away from cell gives cost of leaving cell in sample

Hindsight Optimization: Example



- 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

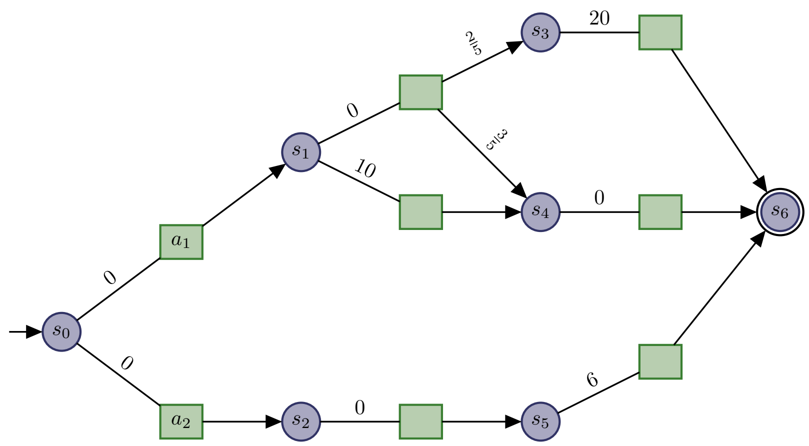


- Samples can be described by number of times agent is stuck
- Multiplication with cost to move away from cell gives cost of leaving cell in sample

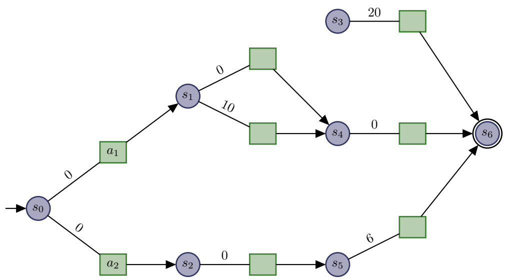
Hindsight Optimization: Evaluation

- HOP well-suited for some problems
- must be possible to solve sampled 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?

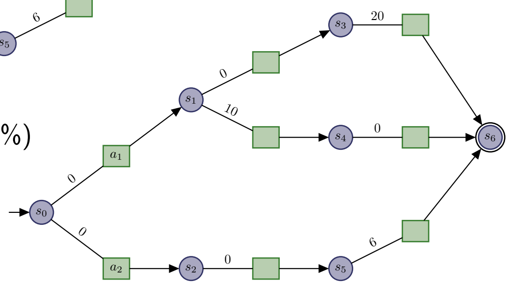
Hindsight Optimization: Optimality in the Limit



Hindsight Optimization: Optimality in the Limit

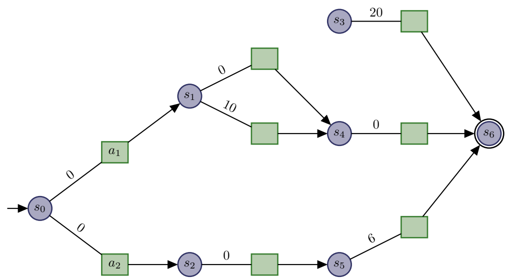


(sample probability: 60%)



(sample probability: 40%)

Hindsight Optimization: Optimality in the Limit

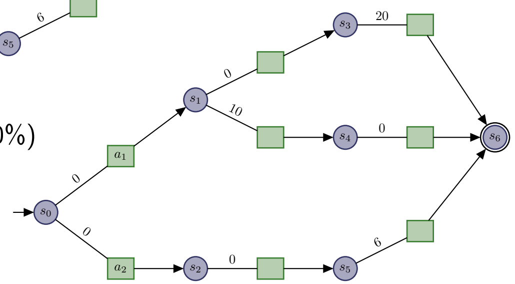


(sample probability: 60%)

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

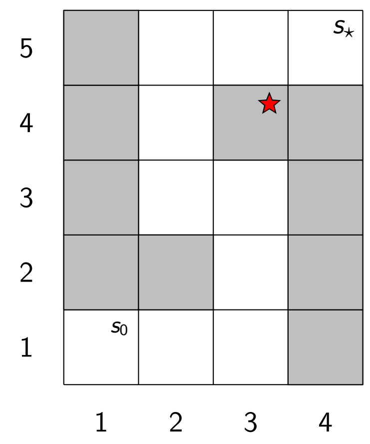
- HOP well-suited for some problems
- must be possible to solve sampled MDP efficiently:
  - domain-dependent knowledge (e.g., games like Bridge, Skat)
  - classical planner (FF-Hindsight, Yoon et. al, 2008)
- What about optimality in the limit?
  ⇒ in general not optimal due to assumption of clairvoyance

G4.4 Policy Simulation

Policy Simulation: Idea

- Avoid clairvoyance by separation of computation of policy and its evaluation:
- Perform samples as long as resources (deliberation time, memory) allow:
  - Sample outcomes of all actions
    ⇒ deterministic (classical) planning problem
  - Compute policy by solving the sample
  - Simulate the policy
- Execute the action with the lowest average simulation cost

Policy Simulation: Example



Policy Simulation: Example

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

1st sample

Policy Simulation: Example

	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

$C_1(s)$

Policy Simulation: Example

	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

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

Policy Simulation: Example

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

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

Policy Simulation: Example

	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

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

Policy Simulation: Example

	1	2	3	4
5	4.53	$\Rightarrow$ 2.0	$\Rightarrow$ 1.0	$s_\star$ 0
4	5.46	$\Uparrow$ 3.0	★ 8.24	2.53
3	6.52	$\Uparrow$ 4.0	5.0	5.11
2	8.99	$\Uparrow$ 6.42	6.0	8.56
1	s_0 $\Rightarrow$ 10.11	$\Uparrow$ 7.78	7.0	11.09

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

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

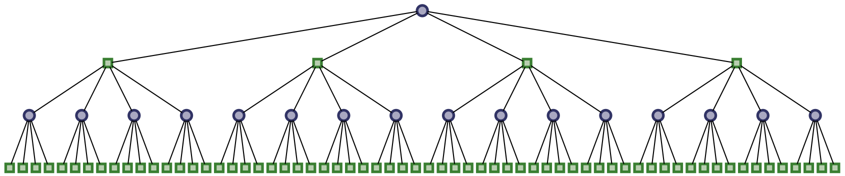
G4.5 Sparse Sampling

Sparse Sampling: Idea

- Proposed by Kearns et al. (2002)
- 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

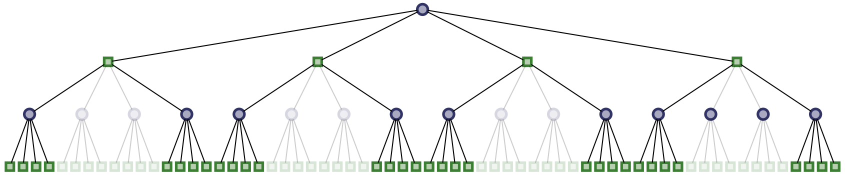
Sparse Sampling: Search Tree

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

G4.6 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 uses plan cost in (deterministic) samples
- 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