Sample-based Planning ~~and Learning~~ for Continuous Markov Decision Processes

Ari Weinstein
Rutgers University

Michael L. Littman
Rutgers University, Brown University

Motivation

- Many real-world planning problems are fundamentally continuous
- Planning literature is mostly interested in discrete domains
  - Common approach is to coarsely discretize continuous dimensions; can be effective but is wasteful
- We are interested in efficient planning in continuous Markov Decision Processes (MDPs)

Overview

- Bandits
  - Discrete, UCB
  - Continuous, HOO
- Markov Decision Processes
  - MDP Planning with HOO: HOLOP
- Empirical Results: HOLOP vs. UCT

sample code made available at:

`http://code.google.com/p/holop/`

k-armed Bandits

- Agent selects an arm from a set *A*, where |*A*|=*k*
- Each arm *a* has a distribution over rewards *R*(*a*)

$$a^* = \arg\max_{a \in A} E[R(a)]$$

- Call the arm pulled at time *t* a_t , reward $r_t \sim R(a_t)$
- The *regret* is the sum of differences in reward between arms pulled and optimal arm; want regret to increase sub-linearly in *t*

$$\text{regret}_\text{t} = \sum_{t=0} R(a^*) - r_t$$

- UCB1 algorithm [Auer et al. 02]:

$$\arg\max_{a \in A} \hat{R}(a) + \sqrt{2\ln(t)/n_a}$$

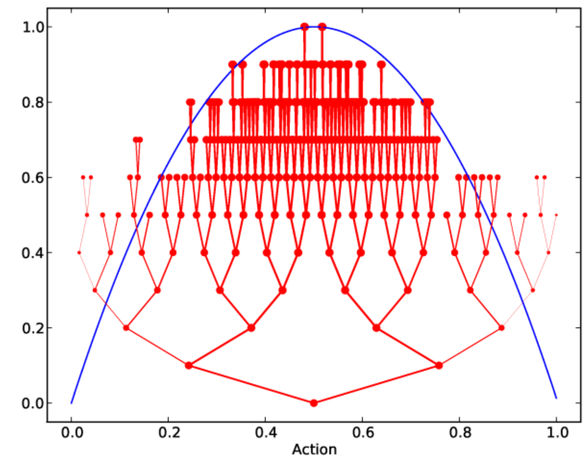
  - Has optimal regret: O(log(*t*))
- Setting can be extended to continuum of arms with some smoothness over the expected reward of "nearby" arms

Hierarchical Optimistic Optimization (HOO)

[Bubeck et al. 08]

- For use in continuous bandit domains (stochastic global optimization)
- Partition action space by a tree, matintain rewards for each subtree

- Follow B-scores from root to leaf, bisect leaf on sampling
- Blue is the bandit, red is the decomposition of HOO tree
  - Thickness represents estimated reward



- Tree grows deeper and builds estimates at high resolution where reward is highest

HOO continued

- Exploration bonuses for number of samples and size of each subregion
  - Regions with large volume and few samples are unknown, vice versa
- Pull arm in region according to maximal *B* from root

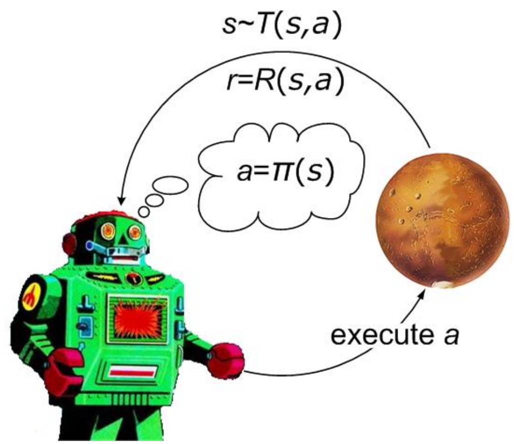
$$U_{h,i}(t) = \hat{\mu}_{h,i}(t) + \sqrt{\frac{2\ln(t)}{N_{h,i}(t)}} + v_1\rho^h$$

$$B_{h,i}(t) = \min\left\{U_{h,i}(t), \max\left\{B_{h+1,2i-1}(t), B_{h+1,2i}(t)\right\}\right\}$$

- Has optimal regret: O(sqrt(*t*)), independent of action dimension

Markov Decision Processes (MDP)

- Composed of:
  - States *S* (*s, s'* from *S*)
  - Actions *A* (*a* from *A*)
  - Transition distribution *T*(*s,a*)
  - Reward function *R*(*s,a*)
  - Discount factor γ



- Assume *A* is infinitely large
- No assumption on *S*
- We will maximize expected discounted finite-horizon return:

$$E\left[\sum_{t=0}^{n} \gamma^t r_t\right]$$

Sample-based Planning in MDPs

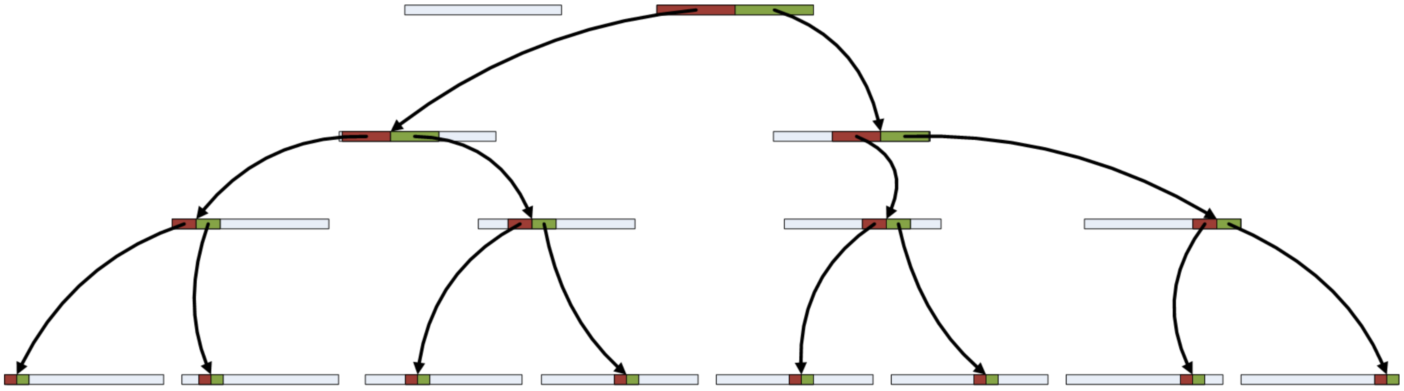
- Agent can query a generative model:
  - For any query <*s*,*a*>, generative model returns <*r*,*s'*>: r = $R(s,a)$, $s' \sim T(s,a)$
- Repeat :
  - Domain informs agent of current state, *s*
  - Agent queries generative model for any number of arbitrary <*s*,*a*> gets <*r*,*s'*>
  - Agent informs domain of true action to take, *a*

MDP Planning with HOO (HOLOP)

- HOLOP – Hierarchical Open Loop Optimistic Planning
- Introduced and analyzed theoretically in Bubeck and Munos 10
- Casts the *n*-step planning problem as a large optimization problem
- Planning is open-loop; a sequence of actions is executed in order and return is observed
- Use HOO to optimize *n*-step planning, and then only use action recommended for first step.

1-Step Lookahead in HOLOP

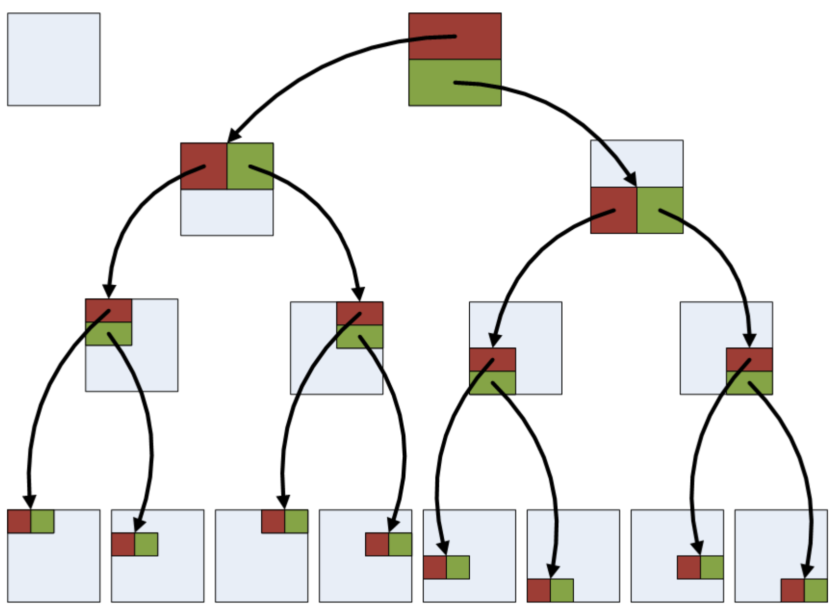
- Maximizes immediate reward, r_1
- 1 dimensional; horizontal axis represents immediate action
- Equivalent to bandit setting



...

2-Step Lookahead in HOLOP

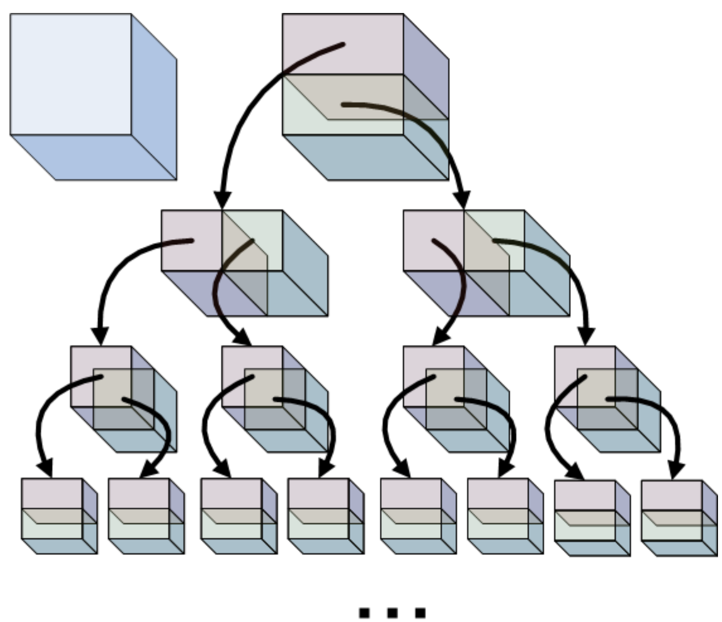
- Maximizes $r_1 + \gamma\, r_2$
- 2 dimensional; horizontal axis represents immediate action, vertical represents next action



...

3-Step Lookahead in HOLOP

- Maximizes $r_1 + \gamma\, r_2 + \gamma^2\, r_3$
- 3 dimensional; horizontal axis represents immediate action, vertical represents next action, depth represents third action
- Can be extended to arbitary dimensions/depth



Properties of HOLOP

- Regret of HOO/HOLOP improves at rate of O(sqrt(*t*)), and independent of |*A*|, *n*
  - True as *t* >> |*A*| x *n* so in most settings size does matter
- Open loop control means state agnostic
  - Cost independent of |*S*|
  - Functions identically in discrete, continuous, hybrid state
- Anytime planner
  - Policy continuously improves over time
  - Unlike PAC-style planners, can be interrupted at any time
- Open loop control means performance can be poor in noisy domains with particular structures

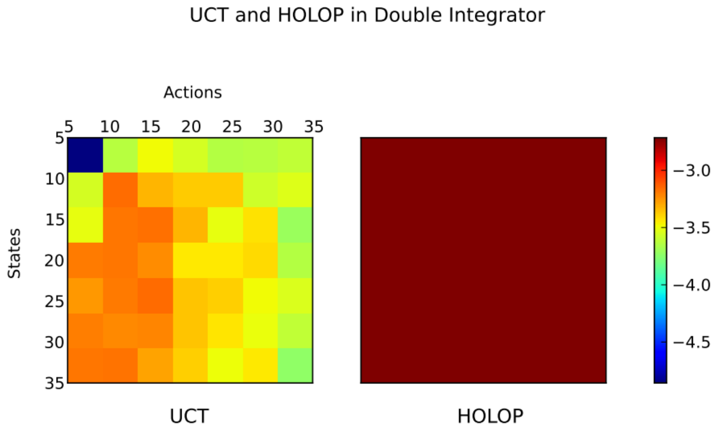
Comparison of HOLOP to UCT

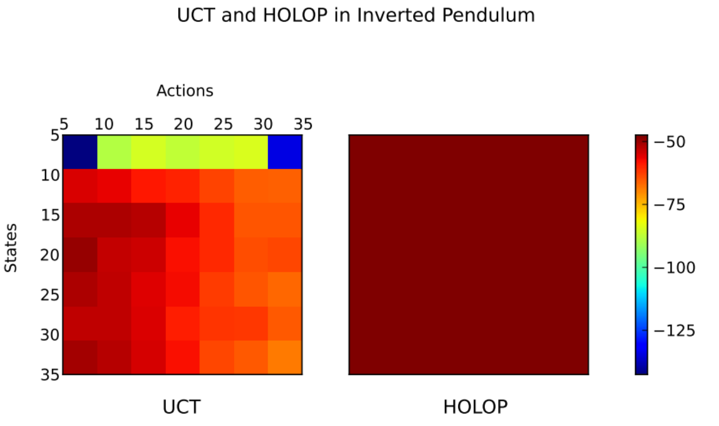
- Upper Confidence Bounds applied to Trees (UCT) is a sample-based planning algorithm [Kocsis Szepesvari 06]
  - Functions in discrete MDPs
  - Has had significant empirical success in Go [Silver et al. 08]
- Because the domains are continuous, UCT uses a uniform coarse discretization over both the state and action spaces

Comparison: UCT and HOLOP

- Double integrator: Object with position(p) and velocity(v) [Santamaría et al. 98]
  - Control acceleration (a)
  - $R((p,v), a) = -(p^2+a^2)$
- Inverted Pendulum: [Pazis and Lagoudakis 09]
  - Reward penalizes high-magnitude actions, angles off balance, and high angular velocities

Comparison: UCT and HOLOP





- Graphs depict average cumulative reward of episodes
- UCT has a “heat map” because it must discretize state and actions; HOLOP functions natively in both domains and needs no tuning
- HOLOP outperforms all 49 parameterizations of UCT (almost all cases with statistical significance)

Conclusions

- HOLOP is an effective continuous-action planner
- It is almost entirely parameter-free
  - Good performance is easy to achieve
- Open loop control means algorithm functions without modification in discrete, continuous, hybrid domains
- Although domains with some structures can cause problems for open-loop planners, we did not encounter this in practice
- Because of poor generalization, algorithms that are effective in discrete domains are usually not effective in continuous domains when a coarse discretization is applied

Citations

- Auer et al. 02: Auer, P., Fischer, P., and Cesa-Bianchi, N. Finite-time analysis of the multi-armed bandit problem. Machine Learning, 47, 2002.
- Bubeck et al 08:, S. Bubeck, R. Munos, G. Stoltz, C. Szepesvari. Online Optimization in X-Armed Bandits, NIPS 08
- Bubeck Munos 10: Bubeck, S., and Munos, R. 2010. Open loop optimistic planning. COLT 2010.
- Moore 91: Efficient Memory-based Learning for Robot Control. PhD. Thesis; University of Cambridge, 1991.
- Nouri Littman 08:Nouri, A. and Littman, M. L. Mutli-resolution Exploration in Continuous Spaces. NIPS 2008
- Pazis and Lagoudakis 09: Pazis, J., and Lagoudakis, M. 2009. Binary action search for learning continuous-action control policies. ICML 2009.
- Santamaría et al. 98: Santamaría, Juan C., Sutton, R., and Ram, Ashwin. Experiments with reinforcement learning in problems with continuous state and action spaces. In Adaptive Behavior 6, 1998.
- Silver et al. 08: Silver, D.; Sutton, R.; and Muller, M. 2008. Sample-based learning and search with permanent and transient memories. ICML 2008.
- Tassa et al. 07: Tassa, Yuval, Erez, Tom, and Smart, William D. Receding horizon differential dynamic programming. In Advances in Neural Information Processing Systems 21. 2007.
- Kocsis Szepesvari 06: Kocsis, L., and Szepesvari, C. 2006. Bandit based monte-carlo planning. In Proceedings of the 17th European Conference on Machine Learning, 282–293.