Markovian State and Action Abstractions for MDPs via Hierarchical MCTS

Aijun Bai[†], Siddharth Srivastava[‡], Stuart Russell[†]

July 12, 2016

UC Berkeley[†] | UTRC[‡]

Background

State Abstraction

State abstraction groups a set of states into a unit:

- Ground MDP: $M = \langle S, A, T, R, \gamma \rangle$
- Abstract states: $X = \{x_1, x_2, \cdots\}$
  - A partition on state space S
- Abstraction function: $\varphi : S \to X$
  - $\varphi(s) \in X$ is the abstract state corresponding to ground state $s \in S$

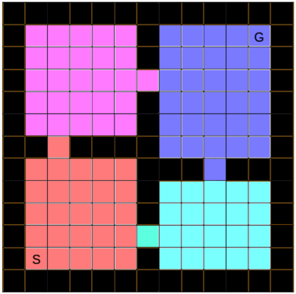


Figure 1: Rooms domain

Non-Markovianess

State abstraction results into a reduced high-level abstract state space. A well-known difficulty for state abstraction:

- Non-Markovianess: $\Pr(x' \mid x, a)$
- Aggregation probability: $\Pr(s \mid x)$
  - Depending on past actions and abstract states
  - Depending on the policy being executed/computed

Safe State Abstraction

Safe state abstraction avoids the non-Markovian problem:

- Ignore only irrelevant state variables (Dietterich, 1999; Andre & Russell, 2002; Jong & Stone, 2005)
- Exploit particular model structure (e.g. bisimulation or homomorphism) (Dearden & Boutilier, 1997; Givan et al., 2003; Jiang et al., 2014; Anand et al., 2015)

However, safe state abstraction is lossless:

- Not always possible
- Computationally difficult to find

The Weighting Function Approach

The *weighting function* approach approximates $\Pr(s \mid x)$ using a fixed weighting function $w(s, x)$ (Bertsekas et al., 1995; Singh et al., 1995; Li et al., 2006):

- Superficially, the state-abstracted model can be written in a Markovian way:
  - $T_\varphi(x' \mid x, a) = \sum_{s' \in \varphi^{-1}(x')} \sum_{s \in \varphi^{-1}(x)} T(s' \mid s, a) w(s, x)$
  - $R_\varphi(x, a) = \sum_{s \in \varphi^{-1}(x)} R(s, a) w(s, x)$
- Abstract MDP: $\langle X, A, T_\varphi, R_\varphi, \gamma \rangle$

However, a fixed weighting function can not capture the true dynamics of the abstract system!

Our Approach

State Abstraction from a POMDP Perspective

Doing state abstraction φ on a ground MDP $M = \langle S, A, R, T, \gamma \rangle$ actually creates a POMDP:

- Abstract states X as observations
- Observation function: $\Omega(x \mid s) = \mathbf{1}[x = \varphi(s)]$
- $\text{POMDP}(M, \varphi) = \langle S, A, X, T, R, \Omega, \gamma \rangle$
  - Underlying MDP: M
- The belief state $b(s)$ in $\text{POMDP}(M, \varphi)$ replaces the *ad-hoc* weighting function

Solving POMDP($\mathcal{M}, \varphi$) via Monte Carlo Tree Search

Exactly solving POMDP($\mathcal{M}, \varphi$) via dynamic programming is intractable. From a tree-based online planning perspective, branching factors:

- $\mathcal{M}$: up to $|S| \times |A|$
- POMDP($\mathcal{M}, \varphi$): up to $|X| \times |A|$

We consider solving it online via MCTS:

- POMCP($\mathcal{M}, \varphi$): POMCP running on POMDP($\mathcal{M}, \varphi$)
  - Build a search tree in the history space via sampling
  - Provided with a simulator for the ground MDP

Action Abstraction on POMDP$(\mathcal{M}, \varphi)$

A given state abstraction naturally induces an action abstraction:

- Extend the theory of options to a POMDP
- Obtain an SMDP with options $\mathcal{O}$ in history space $\mathcal{H}$
- Options connect histories in a one high-level step
  - E.g., option $o_{x \to y}$ connects histories ending with $x \in X$ to histories ending with $y \in X$

Value Function Decomposition for Options

Hierarchical policy for POMDP(M, φ) — $\Pi = \{\mu, \pi_{o_1}, \pi_{o_2}, \cdots\}$:

- $\mu : \mathcal{H} \to \mathcal{O}$ is the overall option-selection policy
- π_o is the inner policy for option $o \in \mathcal{O}$
- MAXQ-like value function decomposition in history space:
  - $Q^{\mu}(h, o) = V^{\pi_o}(h) + \sum_{h' \in \mathcal{H}} \gamma^{|h'|-|h|} \Pr(h' \mid h, o) V^{\mu}(h')$
  - $Q^{\pi_o}(h, a) = R(h, a) + \gamma \sum_{x \in X} \Pr(x \mid h, a) V^{\pi_o}(hax)$

Exploit Action Abstraction via Hierarchical MCTS

The resulting hierarchical MCTS algorithm — POMCP($\mathcal{M}, \varphi, \mathcal{O}$):

- Learn μ by running high-level POMCP over options
- Learn π_o by running low-level POMCP over primitive actions
- Invoke a nested MCTS when evaluating an option
- Update option/action values according to the value function decomposition

Theoretical Results

Theorems

- POMCP($\mathcal{M}, \varphi$) finds the optimal policy for a ground MDP $\mathcal{M}$ consistent with input state abstraction φ
- The performance loss of POMCP($\mathcal{M}, \varphi$) is bounded by a constant multiple of an aggregation error introduced by grouping states with different optimal actions
- POMCP($\mathcal{M}, \varphi, \mathcal{O}$) converges to a recursively optimal hierarchical policy for POMDP($\mathcal{M}, \varphi$) over the hierarchy defined by input state and action abstractions

Experimental Evaluation

The Rooms Domain

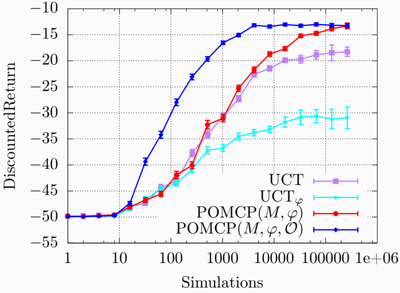
The ROOMS$[m, n, k]$ problem:

- A robot navigates in a $m \times n$ grid map containing k rooms
- Primitive actions: E, S, W and N
- Probability 0.2 of executing a random action

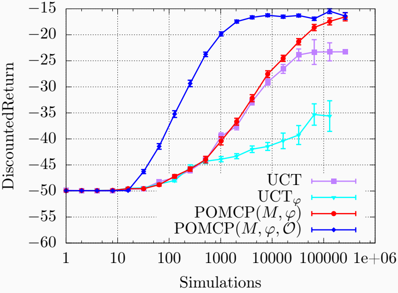
The input state and action abstractions:

- Abstract states: rooms
- Options: transitions between rooms

Experimental Results: The Rooms Domain



(a) ROOMS[17, 17, 4]



(b) ROOMS[25, 13, 18]

Conclusions

Conclusions

- Propose state- and action-abstracted MDPs can be viewed as POMDPs
- Bound the performance loss induced by the abstraction
- Describe a hierarchical MCTS algorithm for approximately solving the abstract POMDP
  - Converge to a recursively optimal hierarchical policy
  - Improve ground MCTS by orders of magnitude empirically

Questions?

References

Anand, A., Grover, A., Mausam, M., & Singla, P. (2015). ASAP-UCT: abstraction of state-action pairs in UCT. In *Proceedings of the 24th International Conference on Artificial Intelligence*, (pp. 1509–1515). AAAI Press.

Andre, D., & Russell, S. J. (2002). State abstraction for programmable reinforcement learning agents. In *AAAI/IAAI*, (pp. 119–125).

Bertsekas, D. P., Bertsekas, D. P., Bertsekas, D. P., & Bertsekas, D. P. (1995). *Dynamic programming and optimal control*, vol. 1. Athena Scientific Belmont, MA.

Dearden, R., & Boutilier, C. (1997). Abstraction and approximate decision-theoretic planning. *Artificial Intelligence*, *89*(1), 219–283.

Dietterich, T. G. (1999). State abstraction in MAXQ hierarchical reinforcement learning. *arXiv preprint cs/9905015*.

Givan, R., Dean, T., & Greig, M. (2003). Equivalence notions and model minimization in Markov decision processes. *Artificial Intelligence*, *147*(1), 163–223.

Jiang, N., Singh, S., & Lewis, R. (2014). Improving UCT planning via approximate homomorphisms. In *Proceedings of the 2014 international conference on Autonomous agents and multi-agent systems*, (pp. 1289–1296). International Foundation for Autonomous Agents and Multiagent Systems.

Jong, N. K., & Stone, P. (2005). State abstraction discovery from irrelevant state variables. Citeseer.

Li, L., Walsh, T. J., & Littman, M. L. (2006). Towards a unified theory of state abstraction for MDPs. In *ISAIM*.

Singh, S. P., Jaakkola, T., & Jordan, M. I. (1995). Reinforcement learning with soft state aggregation. *Advances in neural information processing systems*, (pp. 361–368).

MDPs and POMDPs

Markov decision processes (MDPs) provide a rich framework for planing and learning under uncertainty in fully observable environments:

- An MDP is a tuple $\langle S, A, T, R, \gamma \rangle$

Partially observable Markov decision processes (POMDPs) extend MDPs to partially observable environments:

- A POMDP is a tuple $\langle S, A, Z, T, R, \Omega, \gamma \rangle$
  - Underlying MDP: $\langle S, A, T, R, \gamma \rangle$

The Pseudo Code

Agent (s_0 : *initial state*, φ : *abstraction function*, $\Pi_{rollout}$: *rollout policy*)
$h \leftarrow \varnothing$
$\mathcal{P}(h) \leftarrow \{s_0\}$
repeat
  $\mathcal{T} \leftarrow$ an empty search tree
  $a \leftarrow$ **OnlinePlanning** $(h, \mathcal{T}, \varphi, \Pi_{rollout})$
  Execute a and observe abstract state x
  $h \leftarrow hax$
  $\mathcal{P}(h) \leftarrow$ **ParticleFilter** $(\mathcal{P}(h), a, x)$
until *termination conditions*

Rollout (t : *task*, s : *state*, h : *history*, d : *depth*, φ : *abstraction function*, $\Pi_{rollout}$: *rollout policy*)
if $d \geq H$ **or** t *terminates at* h **then**
  return $\langle 0, 0, h, s\rangle$
else
  $a \leftarrow$ **GetPrimitive** $(\Pi_{rollout}, t, h)$
  $\langle s', r'\rangle \leftarrow$ **Simulate** (s, a)
  $x \leftarrow \varphi(s')$
  $\langle r'', n, h'', s''\rangle \leftarrow$ **Rollout** $(t, s', hax, d+1, \varphi, \Pi_{rollout})$
  $r \leftarrow r' + \gamma r''$
  return $\langle r, n+1, h'', s''\rangle$

GetGreedyPrimitive (t : *task*, h : *history*)
if t *is primitive* **then**
  return t
else
  $a^* \leftarrow \mathrm{argmax}_a\, Q[t, h, a]$
  return GetGreedyPrimitive (a^*, h)

GetPrimitive (Π : *policy*, t : *task*, h : *history*)
if t *is primitive* **then**
  return t
else
  return GetPrimitive $(\Pi, \pi_t(h), h)$

OnlinePlanning (h : *history*, $\mathcal{T}$: *search tree*, φ : *abstraction function*, $\Pi_{rollout}$: *rollout policy*)
repeat
  $s \sim \mathcal{P}(h)$
  Search (*root task*, $s, h, 0, \mathcal{T}, \varphi, \Pi_{rollout}$)
until *resource budgets reached*
return GetGreedyPrimitive (*root task*, h)

Search (t : *task*, s : *state*, h : *history*, d : *depth*, $\mathcal{T}$: *search tree*, φ : *abstraction function*, $\Pi_{rollout}$: *rollout policy*)
if t *is primitive* **then**
  $\langle s', r\rangle \sim$ **Simulate** (s, t)
  $x \leftarrow \varphi(s')$
  return $\langle r, 1, htx, s'\rangle$
else
  if $d \geq H$ **or** t *terminates at* h **then**
    return $\langle 0, 0, h, s\rangle$
  else
    if *node* $\langle t, h\rangle$ *is not in tree* $\mathcal{T}$ **then**
      Insert node $\langle t, h\rangle$ to $\mathcal{T}$
      return Rollout $(t, s, h, d, \varphi, \Pi_{rollout})$
    else
      $a^* \leftarrow \mathrm{argmax}_a \left\{Q[t,h,a] + c\sqrt{\frac{\log N[t,h]}{N[t,h,a]}}\right\}$
      $\langle r', n', h', s'\rangle \leftarrow$ **Search** $(a^*, s, h, d, \mathcal{T}, \varphi, \Pi_{rollout})$
      $\langle r'', n'', h'', s''\rangle \leftarrow$ **Search** $(t, s', h', d+n', \mathcal{T}, \varphi, \Pi_{rollout})$
      $N[t,h] \leftarrow N[t,h] + 1$
      $N[t,h,a^*] \leftarrow N[t,h,a^*] + 1$
      $r \leftarrow r' + \gamma^{n'} r''$
      $Q[t,h,a^*] \leftarrow Q[t,h,a^*] + \frac{r - Q[t,h,a^*]}{N[t,h,a^*]}$
      return $\langle r, n'+n'', h'', s''\rangle$

Figure 3: The overall POMCP($\mathcal{M}, \varphi, \mathcal{O}$) algorithm

Aggregation Error

Definition

The aggregation error of state abstraction $\langle X, \varphi \rangle$ for a ground MDP $M = \langle S, A, T, R, \gamma \rangle$ is e, if $\exists a \in A$, such that for all $x \in X$, $\varphi(s) = x$ and $d \in [0, H]$, $|V_d(s) - Q_d(s, a)| \leqslant e$, where V_d and Q_d are the optimal value and action-value functions at depth d in the search tree of M, and H is the maximal planning horizon.

Optimality Results for State Abstraction

Theorem

For state abstraction $\langle X, \varphi \rangle$ *for a ground MDP* $M = \langle S, A, T, R, \gamma \rangle$ *with aggregation error* e*, let* s_0 *be the current state in the ground MDP* M *and let* h_0 *with* $\mathcal{P}(h_0) = \{s_0\}$ *be the corresponding history in POMDP(*M, φ*). Let* $Q^*(s, \cdot)$ *and* $Q^*(h, \cdot)$ *be the optimal action values of* M *and POMDP(*M, φ*) respectively. Let* $a^* = \text{argmax}_{a \in A} Q^*(h_0, a)$ *be the optimal primitive action found in POMDP(*M, φ*) at history* h_0*, and define an action-value error as* $E(a^*) = |\max_{a \in A} Q^*(s_0, a) - Q^*(s_0, a^*)|$*. Suppose the maximal planning horizon is* H*, then* $E(a^*)$ *is bounded by* $E(a^*) \leqslant 2He$ *if* $\gamma = 1$*, else* $E(a^*) \leqslant 2\gamma \frac{1-\gamma^H}{1-\gamma} e$*.*

Convergence Results with Action Abstraction

Theorem

With probability 1, POMCP($\mathcal{M}, \varphi, \mathcal{O}$) converges to a recursively optimal hierarchical policy for POMDP($\mathcal{M}, \varphi$) over the hierarchy defined by the input state and action abstractions.

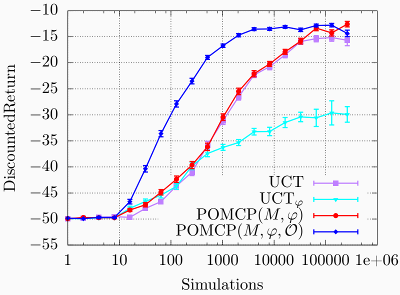
The Continuous Rooms Domain

The C-ROOMS$[m, n, k]$ problem:

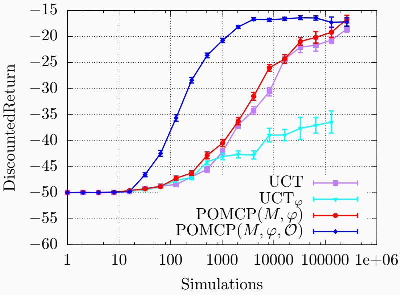
- Each cell has a size of 1 (m^2)
- The position of the agent is represented as (x, y) coordinates
- An action moves the agent by a distance of 1 (m) expectedly
- Gaussian noise is added to each movement

The input state and action abstractions remain the same as in the rooms domain.

Experimental Results: The Continuous Rooms Domain



(a) C-ROOMS[17, 17, 4]



(b) C-ROOMS[25, 13, 18]