State Aggregation in Monte Carlo Tree Search

Jesse Hostetler and **Alan Fern** and **Tom Dietterich**
Department of Electrical Engineering and Computer Science
Oregon State University
{hostetje, afern, tgd}@eecs.oregonstate.edu

Abstract

Monte Carlo tree search (MCTS) algorithms are a popular approach to online decision-making in Markov decision processes (MDPs). These algorithms can, however, perform poorly in MDPs with high stochastic branching factors. In this paper, we study state aggregation as a way of reducing stochastic branching in tree search. Prior work has studied formal properties of MDP state aggregation in the context of dynamic programming and reinforcement learning, but little attention has been paid to state aggregation in MCTS. Our main result is a performance loss bound for a class of value function-based state aggregation criteria in expectimax search trees. We also consider how to construct MCTS algorithms that operate in the abstract state space but require a simulator of the ground dynamics only. We find that trajectory sampling algorithms like UCT can be adapted easily, but that sparse sampling algorithms present difficulties. As a proof of concept, we experimentally confirm that state aggregation can improve the finite-sample performance of UCT.

Introduction

Monte Carlo tree search (MCTS) algorithms (Browne et al. 2012) have increasingly demonstrated state-of-the-art performance on a wide range of Markov decision problems (MDPs) with enormous state spaces (e.g. (Gelly and Silver 2007; Balla and Fern 2009)). MCTS methods work by estimating action values at the current environment state by means of a lookahead search tree built using a simulator of the MDP. A variety of MCTS algorithms have been proposed, such as UCT (Kocsis and Szepesvári 2006) and sparse sampling variants (Kearns, Mansour, and Ng 2002; Walsh, Goschin, and Littman 2010).

One of the key theoretical advantages of MCTS is that the performance bounds are typically independent of the size of the state space. Rather, performance is usually dominated in practice by the effective search depth that can be reached within the decision-time bound. As in any search approach, this depth is determined by the tree branching factor. In MCTS, the branching factor includes both *action branching*, equal to the number of actions available at each state, and *stochastic branching*, equal to the number of possible successor states for each action. Stochastic branching in particular can be enormous if there are many random state variables. Certain MCTS algorithms, such as UCT, simply degenerate to depth 1 search in such cases. This is in spite of the fact that often, much of the stochastic variation is unimportant for selecting the optimal action.

In this paper, we consider state aggregation as a way of reducing the stochastic branching factor for MCTS. The idea is to build trees over abstract states, where each abstract state is a set of aggregated ground states that are effectively "equivalent" with respect to decision making. We introduce a class of Q-function-based state space partitions, which generalizes state aggregation criteria identified by Li, Walsh, and Littman (2006). Our main result is a performance loss bound for decision-making with expectimax tree search over the abstract state spaces induced by partitions in this class. An important consequence of our analysis is that under some conditions it is safe to aggregate states based only on the fact that they share the same optimal action. This is not true in general outside of tree search (Li, Walsh, and Littman 2006) and is significant, since it implies that an ideal abstraction can reduce the stochastic branching factor to be no larger than the number of actions, while preserving optimality. We then consider how to do MCTS in the abstract space, given only a simulator of the ground problem. We show that trajectory sampling algorithms like UCT are easily modified to use abstraction, but that difficulties arise when adapting sparse sampling algorithms due to the need to model the abstract dynamics. Finally, we give illustrative experimental results that show the benefits of abstraction in practice.

Background

We assume basic familiarity with MDP concepts. An MDP is a tuple $M = \langle \mathcal{S}, \mathcal{A}, P, R \rangle$, where $\mathcal{S}$ and $\mathcal{A}$ are finite sets of states and actions, P is the transition function such that $P(s, a, s')$ is the probability of reaching state s' after taking action a in state s, and R is the reward function giving the real-valued reward $R(s)$ for being in state s. We assume that the MDP is absorbing in the sense that all policies are guaranteed to reach a zero-reward terminal state in a finite number of steps. The objective is to find a policy that maximizes the expected sum of rewards until reaching a terminal state, which is finite due to our assumptions. All of our results can be adapted to alternative settings as well, such as finite-horizon and discounted, infinite-horizon problems.

Our discussion of state aggregation will require us to keep track of several different state spaces and their elements. To simplify our notation, we consistently will employ the idiom that s' means a successor of s in $\mathcal{S}$. We will extend this idiom to other state spaces as we introduce them.

Abstractions for General MDPs

We first review state aggregation and abstractions for general MDPs, following prior work (Li, Walsh, and Littman 2006). Let $\mathcal{X}$ be a partition of $\mathcal{S}$ and let $\chi : \mathcal{S} \mapsto \mathcal{X}$ be a surjective function mapping ground states to their equivalence classes in $\mathcal{X}$. We call $\mathcal{X}$ the abstract state space and χ the abstraction function. We stipulate that the set of terminal states maps to a single designated abstract state Ω. An *MDP abstraction* of M is a tuple $\langle \chi, \mu \rangle$, containing the abstraction function χ and a set of functions $\mu = \{\mu_x : x \in \mathcal{X}\}$ such that each μ_x is a probability mass function on the ground states $s \in x$. We call μ the *weighting function*. It can be viewed as defining the relative contribution of each ground state to its corresponding abstract state. An MDP abstraction $\langle \chi, \mu \rangle$ induces an abstract MDP $M/\langle \chi, \mu \rangle = \langle \mathcal{X}, \mathcal{A}, \mathcal{P}_\mu, \mathcal{R}_\mu \rangle$, where

$$\mathcal{P}_\mu(x, a, x') = \sum_{s \in x} \mu_x(s) \sum_{s' \in x'} P(s, a, s')$$
$$\mathcal{R}_\mu(x) = \sum_{s \in x} \mu_x(s) R(s).$$

In keeping with our notational idiom, x' will always refer to a successor of x in $\mathcal{X}$.

Abstraction for Expectimax Trees

Given a ground MDP M and a designated start state s_0, define a trajectory $t \in \mathcal{T} = \{s_0\} \times (\mathcal{A} \times \mathcal{S})^*$ to be any state-action sequence of any length that starts in s_0. A trajectory need not reach a terminal state. Given a depth bound d, we can define the corresponding depth d ground *expectimax search tree* $T(s_0)$ rooted at s_0. The possible nodes of $T(s_0)$ correspond to the set of trajectories of length d or less. The children of node t are its one-step extensions $t' = tas'$ for each $a \in \mathcal{A}$ and $s' \in \mathcal{S}$. It is understood that when t and t' are clear from context, s and s' refer to their tail states.

The tree dynamics are defined with respect to M,

$$P(t, a, t') = P(s, a, s'), \qquad R(t) = R(s).$$

The value and Q functions for each node/trajectory t in an expectimax tree are defined as usual,

$$Q^*(t, a) = R(t) + \sum_{t' \in \mathcal{T}} P(t, a, t') V^*(t'),$$
$$V^*(t) = \max_{a \in \mathcal{A}} Q^*(t, a).$$

Given a state abstraction function χ mapping ground states in $\mathcal{S}$ to abstract states in $\mathcal{X}$, we define application of χ to a trajectory $t \in \mathcal{T}$ as element-wise application of χ to the ground states in t, so that $\chi(t) = \chi(s_0) a_1 \chi(s_1) \dots a_k \chi(s_k)$. We will refer to abstract trajectories as "histories" to distinguish them from ground trajectories. Note that the actions in a history are not abstracted.

Under χ, we can define the history space $\mathcal{H} = \{h \in \{t_0\} \times (\mathcal{A} \times \mathcal{X})^* : |h| \leq d\}$. Each h is an equivalence class over ground trajectories, and hence $\mathcal{H}$ is a partition of $\mathcal{T}$. To fully specify the abstraction, we need weighting functions $\mu = \{\mu_h : h \in \mathcal{H}\}$, which are now indexed by histories. Each μ_h is a distribution over ground trajectories at history node h, where it is assumed that the root node weighting function assigns all weight to t_0, i. e. $\mu_{h_0}(t_0) = 1$.

A tree abstraction is a pair $\langle \chi, \mu \rangle$ defined over trajectories, and it yields an abstract expectimax tree $T(s_0)/\langle \chi, \mu \rangle$ as follows. The tree is rooted at $h_0 = \{t_0\}$ and the tree nodes correspond to histories in $\mathcal{H}$. The children of node h are its one-step extensions $h' \in \{h\} \times \mathcal{A} \times \mathcal{X}$. By convention, h denotes a history and h' its one-step extension, x and x' are the abstract states at the ends of those histories, and t and t' are elements of h and h'.

The dynamics of the abstract tree are defined with respect to the ground tree, just as they were for general MDPs,

$$\mathcal{P}_\mu(h, a, h') = \sum_{t \in h} \mu_h(t) \sum_{t' \in h'} P(t, a, t')$$
$$\mathcal{R}_\mu(h) = \sum_{t \in h} \mu_h(t) R(t)$$
$$\mathcal{Q}^*_\mu(h, a) = \mathcal{R}_\mu(h) + \sum_{h' \in \mathcal{H}} \mathcal{P}_\mu(h, a, h') \max_{a' \in \mathcal{A}} \mathcal{Q}^*_\mu(h', a')$$
$$\mathcal{V}^*_\mu(h) = \max_{a \in \mathcal{A}} \mathcal{Q}^*_\mu(h, a).$$

The branching factor of an abstract expectimax tree is bounded by $|\mathcal{A}||\mathcal{X}|$ rather than the potentially much larger branching factor for ground trees $|\mathcal{A}|B$, where B is the maximum number of successor states for any state-action pair. Thus, given a good abstraction and the ability to run MCTS on the abstract tree, there is potential for significant gains.

State Aggregation Criteria in MCTS

The properties shared by the ground trajectories in each history $h \in \mathcal{H}$ determine the properties of the resulting abstract expectimax tree and its solutions. We will consider a family of state space partitions parameterized by $p, q \in \mathbb{R}^{\geq 0}$. We say that a partition $\mathcal{H}$ is (p, q)-*consistent* if for all $h \in \mathcal{H}$,

$$\begin{aligned} &\exists a^* \ : \ V^*(t) - Q^*(t, a^*) \leq p && \forall t \in h \\ &\left|V^*(t_1) - V^*(t_2)\right| \leq q && \forall t_1, t_2 \in h \end{aligned} \tag{1}$$

The p bound requires that the value of a^* is close to the optimal value of each ground trajectory in h. The q bound requires that the optimal values of different ground trajectories in h are close to one another. Note that there are many (p, q)-consistent abstractions for a given problem. Ordinarily, we would prefer the one with the smallest abstract tree. We will use the notation χ_p^q to denote an abstraction function that induces a (p, q)-consistent partition.

A key consequence of (1) is that if $\mathcal{H}$ is a (p, q)-consistent partition, then for all $h \in \mathcal{H}$ and for any μ,

$$\left| \max_{a \in \mathcal{A}} \sum_{t \in h} \mu_h(t) Q^*(t, a) - \sum_{t \in h} \mu_h(t) V^*(t) \right| \leq p. \tag{2}$$

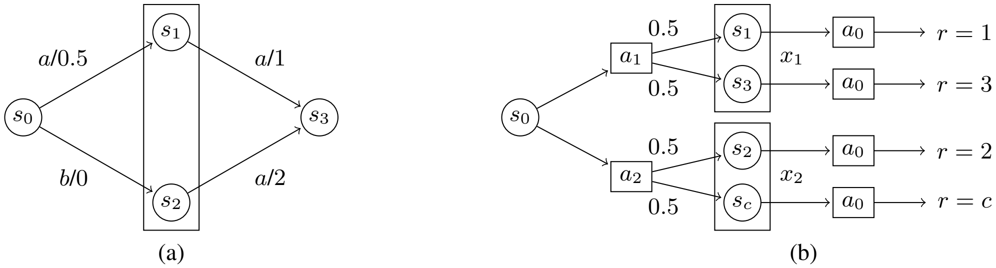

Figure 1: (a) Counterexample for χ_0^∞ in general MDPs (Li, Walsh, and Littman 2006). "$a/0.5$" means action a yields immediate reward 0.5. (b) Counterexample for χ_0^∞ in tree MDPs. Edge labels now denote transition probabilities.

Our concept of (p, q)-consistency generalizes and extends aggregation criteria for general MDPs identified by Li, Walsh, and Littman (2006). In particular, their π^*-irrelevance condition, which requires that aggregated states have the same optimal action, is equivalent to $(0, \infty)$-consistency. Their a^*-irrelevance criterion, which additionally requires that aggregated states have the same value, is equivalent to $(0, 0)$-consistency. We focus on these types of abstractions because they are the coarsest of the hierarchy identified by Li, Walsh, and Littman (2006), and thus they reduce the state space more than, for example, abstractions based on exact bisimilarity (Givan, Dean, and Greig 2003).

Li, Walsh, and Littman (2006) found that in general MDPs under π^*-irrelevance abstraction, the optimal policy in the abstract MDP need not be optimal in the ground MDP, and Q-learning may even fail to converge. Similar convergence failures were noted for the SARSA algorithm by Gordon (1996). In general MDPs, π^*-irrelevance abstractions can make action values non-Markovian with respect to the abstract state space. To illustrate the problem, consider the MDP of Figure 1a, analyzed by Li, Walsh, and Littman (2006) and Gordon (1996). Under a π^*-irrelevance abstraction, s_1 and s_2 can be aggregated (as shown) because a is optimal in both states. For any weighting function μ, the solution to this abstract MDP will select action a in s_0 due to its larger immediate reward, despite action b being optimal in the ground MDP. Stricter notions of abstraction such as a^*-irrelevance avoid these problems, but usually provide much less savings than π^*-irrelevance.

Weighting Function Criteria

A key difference between general MDPs and trees is that the counterexample of Figure 1a is not a problem under a tree structured abstraction, since trajectories ending at s_1 and s_2 will not be aggregated due to their different action prefixes. Still, it is easy to find expectimax trees where a poor choice of μ can lead to unsound decision-making in an abstract tree under a $(0, \infty)$-consistent abstraction. Consider the expectimax tree in Figure 1b, where the abstract tree merges s_1 with s_3 and s_2 with s_c since they agree on optimal actions and share common prefixes. If we choose μ such that $\mu_{x_1}(s_3) = 1$ and $\mu_{x_2}(s_2) = 1$, we will conclude that a_1 is optimal in s_0, losing an expected return of $c/2$.

This example shows that the quality of an abstraction depends on the weighting function μ, as well as the abstraction function χ. This raises two questions: What is the optimal weighting function μ^*, and how does divergence from μ^* affect performance?

For expectimax trees it turns out to be straightforward to define a proper notion of the optimal weighting function. We say that a weighting function μ^* is optimal if for any parent h and child h' histories it satisfies:

$$\mu^*_{h'}(t') = \mathbb{1}_{t' \in h'} \frac{\sum_{t \in x} \mu^*_h(t) P(t, a, t')}{\mathcal{P}_{\mu^*}(h, a, h')} = \mathbb{P}(t'|h'). \quad (3)$$

In the next section, we will show that using appropriate abstraction functions along with such a μ^* will result in sound decision making using the abstract expectimax tree. Further, we will see that certain MCTS algorithms such as UCT can search in abstract spaces defined by abstractions of the form $\langle \chi, \mu^* \rangle$ without computing μ^* explicitly.

Optimality Results

This section establishes our main results. We analyze the application of state aggregation abstractions of the form $\langle \chi_p^q, \mu \rangle$ to expectimax trees and derive a performance loss bound for decision-making under the abstraction.

We bound the performance loss for an abstraction $\langle \chi_p^q, \mu \rangle$ in terms of p, q, and μ. The dependence on μ is in terms of the *single-step divergence* of μ,

$$\delta(\mu, h) = \frac{1}{2} \left\| \mu_h - [\mu]^*_h \right\|_1, \quad (4)$$

$$\text{where } [\mu]^*_{h'}(t') = \mathbb{1}_{t' \in h'} \frac{\sum_{t \in h} \mu_h(t) P(t, a, t')}{\mathcal{P}_\mu(h, a, h')}. \quad (5)$$

This quantity measures the amount of deviation of μ from the optimal weighting function μ^* introduced in the single step from h to h'.

Theorem 1. *Consider an abstraction $\langle \chi_p^q, \mu \rangle$. Let $T(s_0)/\langle \chi_p^q, \mu \rangle$ be the full abstract expectimax search tree rooted at $h_0 = \{t_0\}$ of depth d. Let $m = \max_{h \in \mathcal{H}} \delta(\mu, h) \leq 1$ be the maximum single-step divergence of μ (4). Then the optimal action in the abstract tree, $a^* = \arg\max_{a \in \mathcal{A}} \mathcal{Q}^*_\mu(h_0, a)$, has Q^* error in the root state bounded by*

$$\left| \max_{a \in \mathcal{A}} Q^*(s_0, a) - Q^*(s_0, a^*) \right| \leq 2d(p + mq).$$

Proof. Consider an arbitrary subtree rooted at h in the abstract expectimax tree. If we view μ_h as a distribution over ground states, we see that the optimal action in h is $a^* = \arg\max_{a\in\mathcal{A}} \sum_{t\in h} \mu_h(t) Q^*(t,a)$.

We introduce an action-specific error measure,

$$E(h,a) = \Big|\mathcal{Q}^*_\mu(h,a) - \sum_{t\in h} \mu_h(t) Q^*(t,a)\Big|$$

Our proof will establish a bound on $E(h,a)$ for arbitrary h and a, which necessarily holds for the maximizing action a^*.

The immediate reward terms do not affect $E(h,a)$. The difference between the optimal value in the abstract tree and the true optimal value is thus the error in the future return estimates,

$$\begin{aligned} E(h,a) = \Big| &\sum_{h'\in\mathcal{H}} \mathcal{P}_\mu(h,a,h') \mathcal{V}^*_\mu(h') \\ &- \sum_{t\in h} \mu_h(t) \sum_{t'\in\mathcal{T}} P(t,a,t') V^*(s')\Big|. \end{aligned}$$

We decompose the error as $E(h,a) \leq E_Q + E_\chi$, where,

$$\begin{aligned} E_Q = \Big| &\sum_{h'\in\mathcal{H}} \mathcal{P}_\mu(h,a,h') \mathcal{V}^*_\mu(h') \\ &- \sum_{h'\in\mathcal{H}} \mathcal{P}_\mu(h,a,h') \sum_{t'\in h'} \mu_{h'}(t') V^*(t')\Big|, \\ E_\chi = \Big| &\sum_{h'\in\mathcal{H}} \mathcal{P}_\mu(h,a,h') \sum_{t'\in h'} \mu_{h'}(t') V^*(t') \\ &- \sum_{t\in h} \mu_h(t) \sum_{t'\in\mathcal{T}} P(t,a,t') V^*(t')\Big|. \end{aligned}$$

E_Q is the error due to using the abstract value function below the current node. E_χ is the error introduced by aggregating states at the current level.

The proof will be by induction on the depth of the tree, from the leaf states upwards. Consider a subtree of depth $k+1$ rooted at state h. Assume the inductive hypothesis,

$$E(h',a) \leq k(p+mq),$$

for all $h' \in \{h\} \times \mathcal{A} \times \mathcal{X}$ and all $a \in \mathcal{A}$. Clearly this holds in the absorbing state Ω, which establishes the base case.

For the inductive step, we first derive a bound on E_Q using the inductive hypothesis. Note that,

$$\max_{a\in\mathcal{A}} E(h',a) \geq \Big|\max_{a\in\mathcal{A}} \mathcal{Q}^*_\mu(h',a) - \max_{a\in\mathcal{A}} \sum_{t'\in h'} \mu_{h'}(t') Q^*(t',a)\Big|.$$

By $(p,\cdot)$-consistency of χ, equation (2), and the triangle inequality, we can exchange the max and sum in the above at a cost of at most p. Combining this with the inductive hypothesis gives,

$$\Big|\max_{a\in\mathcal{A}} \mathcal{Q}^*_\mu(h',a) - \sum_{t'\in h'} \mu_{h'}(t') \max_{a\in\mathcal{A}} Q^*(t',a)\Big| \leq k(p+mq)+p,$$

for any $h' \in \mathcal{H}$. We then plug this bound into E_Q to obtain,

$$\begin{aligned} E_Q &= \Big|\sum_{h'\in\mathcal{H}} \mathcal{P}_\mu(h,a,h') \Big[\mathcal{V}^*_\mu(h') - \sum_{t'\in h'} \mu_{h'}(t') V^*(t')\Big]\Big| \\ &\leq k(p+mq)+p. \end{aligned}$$

We now analyze the single-step abstraction error E_χ. This error comes from assigning incorrect weights to ground states within the current abstract state. We can write the second part of E_χ in terms of the "exact" weight update (5),

$$\begin{aligned} &\sum_{h'\in\mathcal{H}} \sum_{t'\in h'} \Big(\sum_{t\in h} \mu_h(t) P(t,a,t')\Big) V^*(t') \\ &\qquad = \sum_{h'\in\mathcal{H}} \sum_{t'\in h'} \mathcal{P}_\mu(h,a,h') [\mu]^*_{h'}(t') V^*(t'). \end{aligned}$$

We can then express E_χ as

$$\begin{aligned} E_\chi = \Big| &\sum_{h'\in\mathcal{H}} \mathcal{P}_\mu(h,a,h') \\ &\cdot \Big[\sum_{t'\in h'} \mu_{h'}(t') V^*(t') - \sum_{t'\in h'} [\mu]^*_{h'}(t') V^*(t')\Big]\Big|. \end{aligned}$$

Let $v(h) = \min_{t\in h} V^*(t)$ be the minimum value among states in h, and let $\Delta(h,t) = V^*(t) - v(h)$. By $(\cdot,q)$-consistency of $\mathcal{H}$, we have $\Delta(h,t) \leq q$. We can express the difference in value estimates in E_χ in terms of Δ,

$$\begin{aligned} D(h') = \Big| &\sum_{t'\in h'} \mu_{h'}(t') [v(h') + \Delta(h',t')] \\ &- \sum_{t'\in h'} [\mu]^*_{h'}(t') [v(h') + \Delta(h',t')]\Big| \\ = \Big| &\sum_{t'\in h'} \mu_{h'}(t') \Delta(h',t') - \sum_{t'\in h'} [\mu]^*_{h'}(t') \Delta(h',t')\Big| \\ \leq &\sum_{t'\in h'} \Big|\mu_{h'}(t') \Delta(h',t') - [\mu]^*_{h'}(t') \Delta(h',t')\Big| \\ \leq &\ q \cdot \frac{1}{2} \sum_{t'\in h'} \Big|\mu_{h'}(t') - [\mu]^*_{h'}(t')\Big| \\ = &\ q \cdot \frac{1}{2} \big\|\mu_{h'} - [\mu]^*_{h'}\big\|_1 = q\delta(\mu,h') \leq mq, \end{aligned}$$

where we have used the fact that $\Delta(h',t') \geq 0$ to introduce the factor of $\frac{1}{2}$. Since E_χ is a convex combination of $D(h')$ for different h', we conclude that $E_\chi \leq mq$. Combining the two sources of error, we obtain,

$$E_Q + E_\chi \leq k(p+mq) + p + mq = (k+1)(p+mq).$$

We thus have $E(h,a) \leq (k+1)(p+mq)$ for any h, a, which completes the inductive argument.

At the root node, we choose an action $a^* = \arg\max_{a\in\mathcal{A}} \mathcal{Q}^*_\mu(h_0,a)$. If $b^* = \arg\max_{a\in\mathcal{A}} Q^*(t_0,a)$ and $a^* \neq b^*$, then $\mathcal{Q}^*_\mu(h_0,a^*) \geq \mathcal{Q}^*_\mu(h_0,b^*)$. In the worst case, the estimate for a^* is $d(p+mq)$ too high, and the estimate for b^* is $d(p+mq)$ too low. Thus the maximum error is $2d(p+mq)$. □

Corollary 1. *Abstraction $\langle\chi_0^\infty, \mu^*\rangle$, corresponding to π^*-irrelevance with μ^*, preserves optimality in search trees.*

Corollary 2. *Abstraction $\langle\chi_0^0, \mu\rangle$, corresponding to a^*-irrelevance, preserves optimality for any μ in search trees.*

State Aggregation in MCTS Algorithms

MCTS algorithms use a simulator, or generative model, of an MDP to construct a sample-based approximation of the expectimax tree rooted at the current state s_0. The action with the best estimated Q-value is then executed. We have seen how state abstraction can reduce the stochastic branching factor of expectimax trees while providing performance guarantees. If we could run MCTS algorithms directly on an abstract tree, we could expect to achieve improved performance given a fixed amount of decision time. However, in practice we only have a simulator for the ground problem, not the abstract problem. In this section, we show that a simple modification of UCT allows us, given only a ground simulator and an abstraction function χ, to replicate the performance of running UCT directly on an abstract tree $T/\langle\chi, \mu^*\rangle$, *without* computing μ^* explicitly. We then consider the case of sparse sampling, where we find that the natural approach to abstracting the algorithm does not weight states according to μ^*, and thus unlike abstract UCT it can incur error for χ_0^q abstractions when $q > 0$.

Abstracting UCT

The UCT algorithm (Kocsis and Szepesvári 2006) builds a tree over sampled trajectories, adding one node to the tree each iteration. Each iteration uses the simulator to produce a trajectory from the root s_0 until reaching the depth bound d, selecting actions as described below. The first node along the trajectory that is not already in the tree is added to the tree as a new leaf. Each tree node t stores the number of times $n(t)$ that the node has been visited, the number of times $n(t, a)$ each action a has been tried in t, and the average return $Q(t, a)$ received for each action a in t.

To choose an action at node t, UCT uses the UCB rule to select an action a that maximizes $Q(t, a) + C \cdot \sqrt{\log n(t)/n(t, a)}$, where C is a parameter of the algorithm. The first term, $Q(t, a)$, favors actions that have led to good rewards, while the second term gives a bonus to infrequently selected actions to encourage exploration. When the current trajectory is outside of the tree, a random action is selected.

Extending UCT with state aggregation is simple. We continue to simulate ground trajectories at each iteration, but we build a tree over the corresponding abstract histories, accumulating node statistics at the abstract history level. Each history node h stores $n(h)$, $n(h, a)$, and $Q(h, a)$, which summarize all simulated ground trajectories going through h. In this way, when generating a ground trajectory, the UCB rule selects actions based on the statistics of the current history node h of the partial trajectory. This new algorithm, which we call χ-UCT, requires both the simulator for ground trajectories and an abstraction function χ for abstracting the ground trajectories during tree construction.

Since UCT and χ-UCT are randomized algorithms, their action choices at the root state are random variables. We now show that χ-UCT faithfully replicates the behavior of running UCT directly on an abstract problem, by showing that these action choice variables are equal in distribution.

Theorem 2. *Consider a ground expectimax tree $T(s_0)$ and an abstract expectimax tree $H = T(s_0)/\langle\chi, \mu^*\rangle$. Let π^w be a random variable giving the action chosen by UCT at s_0 when run on H for w iterations with parameter C. Let $\hat{\pi}^w$ be a random variable giving the action chosen by χ-UCT in s_0 when run on $T(s_0)$ for w iterations with parameter C and abstraction function χ. Then for any w, π^w and $\hat{\pi}^w$ are equal in distribution.*

Proof. (Sketch). We show by induction on the number of iterations w that both algorithms produce the same distribution over trees after w iterations, which implies that the distributions over root decisions are the same. Since the ground trajectories for χ-UCT are sampled from the ground dynamics, the abstract histories seen by χ-UCT exactly replicate the distribution over abstract histories induced by $\mathcal{P}_{\mu^*}$ in H. The base case is trivial since both algorithms start with a single node tree containing just t_0. We then show that given a particular tree over abstract histories produced after w iterations, the distributions over the abstract histories produced by χ-UCT and UCT at iteration $w + 1$ are the same, which results in equivalent distributions over trees at $w + 1$. □

Since χ-UCT running on T is effectively simulating UCT running on $T/\langle\chi, \mu^*\rangle$, the convergence results and sample complexity bounds for UCT (Kocsis and Szepesvári 2006) apply with respect to the abstract tree. Further, when $\chi = \chi_p^q$, Theorem 1 applies, and in the limit χ-UCT incurs root state error bounded above by $2dp$. In particular, if $\chi = \chi_0^\infty$, then χ-UCT converges to the optimal action, which means that the maximum necessary branching factor for optimal decision-making with UCT is $|\mathcal{A}|^2$.

Abstracting Sparse Sampling

Sparse sampling (SS) (Kearns, Mansour, and Ng 2002) is an MCTS algorithm that is perhaps best known for being the first algorithm to achieve approximately optimal MDP planning in time independent of the number of states. Recall that a ground expectimax tree has a branching factor of $|\mathcal{A}|B$, where B is the maximum number of successor states for any state-action pair. To remove the dependence on B, which can scale as the size of the state space, SS samples only w successor states for any state-action pair, where the sampling width w is a parameter of SS. The branching factor of this tree is $|\mathcal{A}|w$ and the main SS theoretical result is that w can be set independently of the number of MDP states in a way that yields near-optimal action choices at the root.

Our abstract version of SS, called χ-SS, constructs a sparse tree over abstract histories to approximate the abstract expectimax tree. For simplicity, we describe χ-SS as an iterative process for producing a depth d abstract tree, but a depth-first implementation of χ-SS is also straightforward. Initially, the tree contains just t_0 as the root. Each iteration of the algorithm picks a non-terminal leaf node of depth less than d and expands it. The iteration stops when all leaf nodes are terminals or have depth d. Each node corresponds to an abstract history h and stores bookkeeping information corresponding to a *multiset* of ground states $S_h = \{t : \chi(t) = h\}$.

From each S_h in the sampled tree, we estimate $\hat{\mu}_h$, the empirical weight function for h, using any density estimator (e.g. a histogram). To expand node h, for each action

a the algorithm samples w ground states $t_1, \ldots, t_w$ from $\hat{\mu}_h$, and then samples a successor state t'_i from $P(t_i, a, \cdot)$ for each t_i. This produces a multiset of ground next states $S' = \{t'_1, \ldots, t'_w\}$, which are then partitioned into equivalence classes using the abstraction function χ, yielding some number $k \leq w$ of abstract states $\{h'_1, \ldots, h'_k\}$. The children of h corresponding to action a are the histories $\{h'_1, \ldots, h'_k\}$. The bookkeeping information for child h'_i is the multiset $S_{h'_i} = \{t' \in S' : \chi(t') = h'_i\}$. Since the abstract nodes might contain different numbers of ground states, each such node h is assigned a weight of $|S_h|/w$. When action values are finally computed for the tree, these weights are taken into account when averaging values.

Since $\hat{\mu}$ is a finite-sample estimate, in general $\hat{\mu} \neq \mu^*$. This means that χ-SS can incur error due to weight function inaccuracy (the m term in Theorem 1). The performance of χ-SS thus depends on the properties of $\hat{\mu}$.

Theorem 3. *Consider a ground expectimax tree $T(s_0)$. Let $\tilde{\pi}^w$ be a random variable giving the action chosen by χ-SS when run on $T(s_0)$ with abstraction function χ, density estimator $\hat{\mu}$, and sampling width w. Let π^w be a random variable giving the action chosen by SS when run on $H = T(s_0)/\langle\chi, \hat{\mu}\rangle$ with sampling width w. Then for all w, π^w and $\tilde{\pi}^w$ are equal in distribution.*

Proof. Since χ-SS directly estimates μ^* with $\hat{\mu}$, we are actually constructing a simulator for $T(s_0)/\langle\chi, \hat{\mu}\rangle$ and running SS in the abstract space directly. □

The convergence results for SS (Kearns, Mansour, and Ng 2002) then apply to χ-SS with respect to the abstract problem. Unfortunately, the problem of estimating the weight functions is non-trivial and may introduce an error bound on m that depends on the size of the ground state space. Algorithms based on trajectory sampling such as χ-UCT are thus preferable when using χ_p^q abstractions for which q is significant.

Experimental Illustration

This section presents a small experiment that demonstrates the sample complexity benefits of abstraction. Our experimental domain is a version of the card game Blackjack. We play to a maximum score of 32, instead of 21 for ordinary Blackjack. This makes the planning horizon longer, which allows abstraction to have a larger effect. We draw from an infinite deck so that card counting is not helpful, and we do not allow doubling down, splitting pairs, or surrendering.

We compared four different state representations. The *flat* representation discriminates states based on the actual cards, so K♠Q♢ is considered distinct from K♠Q♡. The *value* representation treats hands with the same numeric value as equivalent. The χ_0^∞ representation aggregates states according to $(0, \infty)$-consistency. The "noisy" χ_0^∞ representation adds random noise to χ_0^∞. We first compute the optimal policy, then flip 30% of the optimal actions and construct the noisy χ_0^∞ abstraction based on this corrupted policy.

We ran χ-UCT with the four representations for varying sample limits. The performance measure is the average return over 10^5 games. As Figure 2 shows, the coarser abstractions performed better for a given number of samples. Further, the noisy χ_0^∞ abstraction, despite putting 30% of ground states in the wrong abstract state, outperformed the (exact) value abstraction. These results show both that abstraction can improve search performance, and that inaccurate abstractions can perform better than exact ones if the inaccurate abstractions achieve a smaller state space.

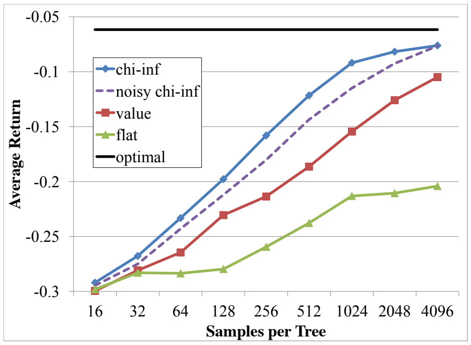

Figure 2: Small-sample performance of χ-UCT with flat, value, and χ_0^∞ representations. The x-axis shows the number of samples per tree on a log scale. The flat line is the optimal value. 95% confidence intervals are smaller than ±0.006 for all data points.

Related Work

State abstractions have been employed in search-based classical planning (e.g. (Hoffmann, Sabharwal, and Domshlak 2006) and references therein). Much of the theory of state abstraction in MDPs has used the framework of bisimilarity, both exact (Givan, Dean, and Greig 2003) and approximate (Ferns, Panangaden, and Precup 2004). Our work is most closely related to (Jiang, Singh, and Lewis 2014), which proposes state aggregation based on approximate bisimilarity in MCTS. The analysis in that work is in the context of the UCT algorithm. In contrast, our abstractions are generalizations of the Q-function-based π^*- and a^*-irrelevance abstractions studied by Li, Walsh, and Littman (2006), which are generally much coarser than abstractions based on exact bisimilarity. Also, our analysis is in the context of full expectimax trees, so it applies to any method of sampling. Van Roy (2006) derived error bounds that look similar to ours for value iteration with state aggregation.

Summary

We have analyzed state aggregation in exact expectimax search and MCTS. Our results established a performance loss bound for state aggregation in expectimax trees for a family of state space partitions called (p, q)-consistent partitions, which includes the π^*-irrelevance and a^*-irrelevance conditions of Li, Walsh, and Littman (2006). We showed how to extend UCT and sparse sampling to build abstract search trees, and combined our results with existing theory to analyze the extended algorithms. Our experiments with χ-UCT demonstrated the benefits of abstraction

Acknowledgments

This research was supported by NSF grant IIS 1320943, NSF grant 0958482, and ARO grant W911NF-08-1-0242. The views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing the official policies, either expressed or implied, of ARO or the United States Government.

References

Balla, R.-K., and Fern, A. 2009. UCT for tactical assault planning in real-time strategy games. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence (IJCAI)*, 40–45.

Browne, C. B.; Powley, E.; Whitehouse, D.; Lucas, S. M.; Cowling, P. I.; Rohlfshagen, P.; Tavener, S.; Perez, D.; Samothrakis, S.; and Colton, S. 2012. A survey of Monte Carlo tree search methods. *IEEE Transactions on Computational Intelligence and AI in Games* 4(1):1–43.

Ferns, N.; Panangaden, P.; and Precup, D. 2004. Metrics for finite Markov decision processes. In *Proceedings of the 20th Conference on Uncertainty in Artificial Intelligence (UAI)*, 162–169.

Gelly, S., and Silver, D. 2007. Combining online and offline knowledge in UCT. In *Proceedings of the 24th International Conference on Machine Learning (ICML)*, 273–280.

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

Gordon, G. J. 1996. Chattering in SARSA(λ). Technical report, Carnegie Mellon University.

Hoffmann, J.; Sabharwal, A.; and Domshlak, C. 2006. Friends or foes? An AI planning perspective on abstraction and search. In *Proceedings of the 16th International Conference on Automated Planning and Scheduling (ICAPS)*, 294–303.

Jiang, N.; Singh, S.; and Lewis, R. 2014. Improving UCT planning via approximate homomorphisms. In *Proceedings of the 13th International Conference on Autonomous Agents and Multiagent Systems (AAMAS)*.

Kearns, M.; Mansour, Y.; and Ng, A. Y. 2002. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. *Machine Learning* 49(2-3):193–208.

Kocsis, L., and Szepesvári, C. 2006. Bandit based Monte-Carlo planning. In *Proceedings of the European Conference on Machine Learning (ECML)*, 282–293.

Li, L.; Walsh, T. J.; and Littman, M. L. 2006. Towards a unified theory of state abstraction for MDPs. In *Proceedings of the Ninth International Symposium on Artificial Intelligence and Mathematics*, 531–539.

Van Roy, B. 2006. Performance loss bounds for approximate value iteration with state aggregation. *Mathematics of Operations Research* 31(2):234–244.

Walsh, T. J.; Goschin, S.; and Littman, M. L. 2010. Integrating sample-based planning and model-based reinforcement learning. In *Proceedings of the 24th AAAI Conference on Artificial Intelligence (AAAI)*.