Efficient Bounds in Heuristic Search Algorithms for Stochastic Shortest Path Problems

Eric A. Hansen and **Ibrahim Abdoulahi**
Dept. of Computer Science and Engineering
Mississippi State University
Mississippi State, MS 39762
hansen@cse.msstate.edu, ia91@msstate.edu

Abstract

Fully observable decision-theoretic planning problems are commonly modeled as stochastic shortest path (SSP) problems. For this class of planning problems, heuristic search algorithms (including LAO*, RTDP, and related algorithms), as well as the value iteration algorithm on which they are based, lack an efficient test for convergence to an ϵ-optimal policy (except in the special case of discounting). We introduce a simple and efficient test for convergence that applies to SSP problems with positive action costs. The test can detect whether a policy is proper, that is, whether it achieves the goal state with probability 1. If proper, it gives error bounds that can be used to detect convergence to an ϵ-optimal solution. The convergence test incurs no extra overhead besides computing the Bellman residual, and the performance guarantee it provides substantially improves the utility of this class of planning algorithms.

Introduction

Fully observable decision-theoretic planning problems are commonly modeled as stochastic shortest path (SSP) problems. For this class of problems, actions can have probabilistic outcomes, and the objective is to find a conditional plan, or policy, that reaches a goal state from a start state with minimum expected cost. Classic dynamic programming and linear programming methods solve such problems for the entire state space. By contrast, several algorithms developed in the AI community use heuristic search to focus computation on relevant states, and can find a conditional plan while evaluating only a fraction of the state space.

There are two broad strategies for using heuristic search in solving SSP problems. Real-time dynamic programming (RTDP) uses trial-based search methods that generalize real-time heuristic search techniques (Barto, Bradtke, and Singh 1995; Bonet and Geffner 2003b; McMahan, Likhachev, and Gordon 2005; Smith and Simmons 2006; Sanner et al. 2009). Another strategy employs systematic search methods that generalize techniques for traditional AND/OR graph search (Hansen and Zilberstein 2001; Bonet and Geffner 2003a; 2006; Warnquist, Kvarnström, and Doherty 2010).

Both strategies are based on the insight that if the initial cost-to-go function is an admissible heuristic (that is, if it underestimates the optimal cost-to-go function), then an asynchronous value iteration algorithm that only updates states that are reachable from the start state under a greedy policy is guaranteed to converge to the optimal cost-to-go function *in the limit*, without necessarily evaluating the entire state space. Given this guarantee of eventual convergence, the question we address in this paper is how to detect convergence to an ϵ-optimal solution after a finite number of steps, that is, how to detect convergence in practice.

The original RTDP algorithm does not include a convergence test (Barto, Bradtke, and Singh 1995). The first heuristic search algorithm for SSP problems to include a test for convergence to an ϵ-optimal policy is the LAO* algorithm (Hansen and Zilberstein 2001). However, its test for convergence relies on bounds that are as expensive to compute as exact policy evaluation, and its approach has not been used in subsequent algorithms; it is even omitted in some implementations of LAO*. Bonet and Geffner describe several related algorithms, including HDP (2003a), LRTDP (2003b), and LDFS (2006), that adopt a similar strategy as LAO*, with the addition of a labeling technique that accelerates convergence. But the algorithms do not attempt to detect convergence to an ϵ-optimal policy. Instead, they test for convergence to an ϵ-*consistent* cost-to-go function, which means they test whether the Bellman residual (the largest change in state value in a given iteration) is less than some $\epsilon > 0$. Although this convergence test is much more efficient than the one proposed for LAO*, it does not provide a genuine error bound. In fact, it does not guarantee that a policy found by this approach will reach the goal state.

Another approach to testing for convergence is to compute both lower and upper bounds, and test whether the gap between the bounds is less than ϵ. This approach has been implemented in extensions of RTDP (McMahan, Likhachev, and Gordon 2005; Smith and Simmons 2006; Sanner et al. 2009) and LAO* (Warnquist, Kvarnström, and Doherty 2010), and it has the advantage that it provides meaningful error bounds. It also has the advantage that if the upper-bound function is computed in such a way that it is a monotone upper bound, a greedy policy with respect to a monotone upper-bound function is guaranteed to be proper (McMahan, Likhachev, and Gordon 2005). A drawback of this approach is that it computes both lower and upper-bound functions, requiring twice as much computation. It must also compute an initial monotone upper-bound function, which is more difficult than computing an initial lower-bound function, and can require an expensive policy evaluation step that is performed for the entire state space.

In this paper, our contribution is to introduce a simple and efficient convergence test that applies to any SSP problem for which all action costs are positive. The convergence test has two steps. The first step detects whether a greedy policy with respect to a lower-bound function is *proper* (which means a goal state is reached with probability 1). If so, the second step gives error bounds that can be used to test whether the policy is ϵ-optimal. The two-step convergence test can be used in any algorithm for solving SSP problems that computes a Bellman residual, including LAO*, HDP, LDFS, LRTDP, related algorithms, and value iteration itself. Moreover, computing the bounds incurs no extra overhead besides the overhead for computing the Bellman residual.

The assumption of positive action costs is essential to our result, but hardly a limiting assumption. It is satisfied in most practical applications, and the SSP model is often formulated by assuming positive action costs. A more serious limitation is that the bounds we derive are tightest when action costs are uniform, or nearly uniform, in addition to positive. But even with this limitation, our results provide excellent bounds for most decision-theoretic planning problems.

We begin by introducing the convergence test in the context of value iteration for SSP problems. Then we describe how to integrate it in heuristic search algorithms. Finally, we report some experimental results that illustrate the behavior of the bounds and demonstrate their usefulness.

SSP problems and value iteration

A stochastic shortest path (SSP) problem is a discrete-stage Markov decision process with the following elements:

- S denotes a finite state set of non-goal states, with an initial state $s_0 \in S$, and G denotes a finite and non-empty set of goal states;
- for each state $i \in S \cup G$, $A(i)$ denotes a non-empty set of feasible actions, with $A(i)$ assumed to be a singleton set for each goal state $i \in G$;
- P^a_{ij} denotes the probability that action $a \in A(i)$ taken in state i results in a transition to state j; and
- $g^a_i \in \Re$ denotes the immediate cost received when action $a \in A(i)$ is taken in state i.

By assumption, goal states are zero-cost and absorbing, that is, for all $i \in G$, $g^a_i = 0$ and $P^a_{ii} = 1$ for $a \in A(i)$. Although an SSP problem is an infinite-horizon problem, the decision process effectively terminates once a goal state is reached.

Let M denote a set of deterministic and stationary policies, where a policy $\mu \in M$ maps each state $i \in S$ to an action $\mu(i) \in A(i)$. A policy is said to be *proper* if it ensures that a goal state is reached with probability greater than zero from *every* state $i \in S$. Under this condition, it follows that a goal state is reached with probability 1 from every state $i \in S$. A policy that is not proper is said to be improper.

The cost-to-go function $J_\mu : S \rightarrow \Re \cup \{\pm\infty\}$ of a policy μ gives the expected total cost incurred by following policy μ starting from any state i, defined as

$$J_\mu(i) = 0, \; i \in G, \quad (1)$$

$$J_\mu(i) = g^a_i + \sum_{j \in S} P^a_{ij} J_\mu(j), \; i \in S. \quad (2)$$

For any proper policy μ, we have $J_\mu(i) < \infty, i \in S$. The optimal cost-to-go function J^* is defined as

$$J^*(i) = \min_{\mu \in M} J_\mu(i), \; i \in S, \quad (3)$$

and an optimal policy μ^* satisfies

$$J^*(i) = J_{\mu^*}(i) \leq J_\mu(i), \qquad \forall \mu \in M, i \in S. \quad (4)$$

For SSP problems, Bertsekas and Tsitsiklis (1991) show that an optimal policy is proper under the following assumptions: (i) there is at least one proper policy $\mu \in M$, and (ii) any improper policy incurs positive infinite cost for some initial state. The second condition is satisfied when all actions (taken in a non-goal state) have positive cost. In the rest of the paper, we assume positive-cost actions.

Value iteration

SSP problems can be solved by linear programming, or the dynamic programming algorithms of value iteration or policy iteration. We focus on value iteration because all heuristic search algorithms for SSP problems perform a form of asynchronous value iteration on a cost-to-go function that is a lower-bound function. (Policy iteration improves a cost-to-go function that is a monotone upper bound.)

Given an initial cost vector J_0, value iteration generates an improved cost vector J_k each iteration $k = 1, 2, 3, \ldots$ by performing the following *dynamic programming update*,

$$J_k(i) = \min_{a \in A(i)} \left\{ g^a_i + \sum_{j \in S} P^a_{ij} J_{k-1}(j) \right\}, \; i \in S, \quad (5)$$

where $J_k(i) = 0, i \in G$. Bertsekas and Tsitsiklis (1991) prove that for any initial vector $J_0 \in \Re^n$, the sequence of vectors, $J_1, J_2, J_3, \ldots$, generated by value iteration converges in the limit to the optimal cost vector J^*. A *greedy policy* with respect to a cost vector J_{k-1} is defined as

$$\mu^k(i) = \arg \min_{a \in A(i)} \left\{ g^a_i + \sum_{j \in S} P^a_{ij} J_{k-1}(j) \right\}, \; i \in S, \quad (6)$$

and a greedy policy with respect to the optimal cost vector J^* is optimal.

The rate at which value iteration converges can often be accelerated by using Gauss-Seidel dynamic programming updates, defined as follows. For $i \in S$,

$$J_k(i) = \min_{a \in A(i)} \left\{ g^a_i + \sum_{j=1}^{i-1} P^a_{ij} J_k(j) + \sum_{j=i}^{n} P^a_{ij} J_{k-1}(j) \right\}, \quad (7)$$

where $J_k(j)$ is used in place of $J_{k-1}(j)$ when $J_k(j)$ is available. Using Gauss-Seidel updates, the cost-to-go function can improve more in a given iteration than using traditional updates, especially given an intelligent ordering of states.

Error bounds

In practice, value iteration must be terminated after a finite number of iterations, and it is useful to be able to bound the sub-optimality of a solution. Let J_k denote a cost vector that is the result of a dynamic programming update of the cost vector J_{k-1}. In keeping with our interest in heuristic search algorithms, we assume that both J_k and J_{k-1} are lower-bound functions. Let μ^k denote a greedy policy with respect to J_{k-1}, and recall that J_{μ^k} denotes the cost-to-go function of the policy μ^k. A policy μ^k is ϵ-optimal if $J_{\mu^k}(i) - J_k(i) \leq \epsilon, \forall i \in S$, where $J_k(i) \leq J^*(i) \leq J_{\mu^k}(i), \forall i \in S$. Since an improper policy has positive infinite cost for some initial state, only a proper policy can be ϵ-optimal.

For SSP problems, Bertsekas (2005, p. 413) derives error bounds for value iteration that take the following form when the cost vector J_k is a lower-bound function. For all $i \in S$,

$$J_k(i) \leq J^*(i) \leq J_{\mu^k}(i) \leq J_k(i) + (N_k(i) - 1) \cdot \overline{c}_k, \quad (8)$$

where μ^k is a greedy policy with respect to J_{k-1}, $N_k(i)$ is the expected number of stages to reach a goal state starting from state i and following policy μ^k, and

$$\overline{c}_k = \max_{i \in S} \left\{ J_k(i) - J_{k-1}(i) \right\}, \quad (9)$$

which we refer to from now on as the Bellman residual.[1]

Since J_k is a lower-bound function, all we need to bound the sub-optimality of a solution is to compute either one of the two upper bounds on the right-hand side of (8). However, there are difficulties in doing so. First, the upper bounds are finite only if the policy μ^k is proper, and there is no guarantee that a greedy policy with respect to a lower-bound function is proper. It is possible to determine whether a given policy is proper by analyzing the structure of the directed graph corresponding to the underlying Markov chain. But doing so takes linear time in the size of the problem.

Once a policy μ^k is determined to be proper, one way to get an upper bound is to compute its cost-to-go function, J_{μ^k}, using policy evaluation. But exact policy evaluation requires solving a system of $|S|$ linear equations in $|S|$ unknowns. An upper bound can also be computed using the inequality $J^*(i) \leq J_k(i) + (N_k(i) - 1) \cdot \overline{c}_k$ from (8). But determining $N_k(i)$, which is the expected number of stages to reach a goal state starting from state i and following policy μ^k, requires solving the following system of $|S|$ linear equations in $|S|$ unknowns,

$$N_k(i) = 0, \; i \in G, \quad (11)$$

$$N_k(i) = 1 + \sum_{j \in S} P_{ij}^{\mu^k(i)} N_k(j), \; i \in S. \quad (12)$$

[1]Bertsekas also defines the quantity

$$\underline{c}_k = \min_{i \in S} \left\{ J_k(i) - J_{k-1}(i) \right\}, \quad (10)$$

which can be used to compute a lower bound. It plays no role in the bounds we derive in this paper since we assume that J_k is a lower-bound function, that is, an admissible heuristic. Note that if $\underline{c}_k \geq 0$, then J_k is a monotone lower-bound function, or, equivalently, a consistent heuristic. Our results only depend on J_k being an admissible heuristic, not a consistent heuristic.

which is as difficult as policy evaluation. "Unfortunately," writes Bertsekas (2005, p. 414), the bounds of (8) "are easily computed or approximated only in the presence of special problem structure." The only example of special problem structure given by Bertsekas is discounting. As is well-known, any discounted Markov decision problem can be reduced to an equivalent SSP problem, in which case, $(N_k(i) - 1) = \beta/(1-\beta)$ for all $\mu \in M, i \in S$. Except for the special case of discounting, the bounds of (8) are generally considered too expensive to be useful in practice.

Instead of using the bounds of (8), standard practice when using value iteration to solve SSP problems is to test for convergence by simply testing whether the Bellman residual is less than some $\epsilon > 0$. In the terminology of Bonet and Geffner (2003b), a cost-to-go function that satisfies this condition is said to be ϵ-consistent. Although useful and easy to compute, a drawback of this convergence test is that it gives no real error bound, or even a guarantee that a policy is proper.

New convergence test and bounds

With this background, we are ready to introduce the main result of the paper, which is a two-step convergence test that both (i) tests whether a policy is proper, and, if so, (ii) bounds its sub-optimality. The first part of the result draws on a connection between SSP problems and average-cost Markov decision processes. The second part is based on the Bertsekas bounds of (8). Both require the assumption that all actions (taken in a non-goal state) have positive cost.

Surprisingly, the new convergence test incurs no more overhead than the test for whether a cost-to-go function is ϵ-consistent. Both tests simply look at the Bellman residual.

Theorem 1. *For an SSP problem where all actions taken in a non-goal state have positive cost, let* $\underline{g} = \min_{i \in S, a \in A(i)} g_i^a$ *denote the smallest action cost, and consider a lower-bound function* J_k *that is updated by the value iteration recursion of* (5)*, where* $\overline{c}_k$ *is the Bellman residual defined by* (9)*. If* $\overline{c}_k < \underline{g}$ *then:*

1. *a greedy policy* μ^k *with respect to* J_{k-1} *is proper, and*
2. *for each state* $i \in S$*, an upper bound* $\overline{J}_{\mu^k}(i)$ *on the cost-to-go* $J_{\mu^k}(i)$ *for the greedy policy* μ^k*, and thus an upper bound on the optimal cost-to-go* $J^*(i)$*, is given by*

$$\overline{J}_{\mu^k}(i) = \frac{(J_k(i) - \overline{c}_k) \cdot \underline{g}}{(\underline{g} - \overline{c}_k)}. \quad (13)$$

Proof. We prove the two parts of the theorem in turn.

1. Recall that a policy is proper if and only if it ensures that a goal state is reached with probability greater than zero from *every* initial state $i \in S$. It follows that for an improper policy μ, there must be at least one non-goal state from which a goal state is never reached. Let $S' \subseteq S$ denote the non-empty subset of states from which a goal state is never reached when following the improper policy μ.

Now consider the average-cost criterion. Let $\lambda_\mu : S \to \Re$ denote the average-cost vector for a policy μ, where $\lambda_\mu(i)$ gives the average cost per stage of a process that follows policy μ starting from state i. For any proper policy μ, we have

$\lambda_\mu(i) = 0, \forall i \in S$, since a zero-cost goal state is eventually reached with probability 1. For any improper policy μ, we have $\lambda_\mu(i) \geq \underline{g}$ for any state i in S', since the average cost per stage cannot be less than $\underline{g}$ when a goal state is never reached. In this case, of course, we also have $J_\mu(i) = \infty$.

Let $\lambda : S \to \Re$ denote the average-cost vector for an optimal policy, which means that $\lambda_\mu(i) \geq \lambda(i), \forall \mu \in M, i \in S$. For the sequence of cost vectors $\{J_k | k \geq 0\}$ generated by value iteration, we have for each state $i \in S$ (Kallenberg 2002, p. 49, Corollary 2.8):

$$\lambda(i) = \lim_{n\to\infty} \frac{1}{n} \sum_{k=1}^{n} (J_k(i) - J_{k-1}(i)). \tag{14}$$

Recall that $\overline{c}_k = \max_{i\in S}(J_k(i) - J_{k-1}(i))$ by definition, and that $\overline{c}_k \leq \overline{c}_{k-1}$ for all $k > 0$. It follows that $\overline{c}_k \geq \max_{i\in S} \lambda(i)$, for all $k > 0$. Thus the Bellman residual never decreases below $\underline{g}$ when μ^k is an improper policy. (The same result follows from the error bound for the average-cost criterion given by Bertsekas (2012, p. 329).)

Therefore, if $\overline{c}_k < \underline{g}$, where μ^k is a greedy policy with respect to J_{k-1}, it is not possible that $\lambda_{\mu^k}(i) \geq \underline{g}$, for any i in S, and that means $\lambda_{\mu^k}(i) = 0, \forall i \in S$, which implies that the greedy policy μ^k is proper.

2. First note that the upper bound given by the rightmost inequality of Equation (8) remains valid if $N_k(i)$ is replaced by an upper bound $\overline{N}_k(i)$ on the expected number of stages to reach a goal state from state i following policy μ^k, where $N_k(i) \leq \overline{N}_k(i)$. Thus we have:

$$J_k(i) \leq J^*(i) \leq J_{\mu^k}(i) \leq J_k(i) + (\overline{N}_k(i) - 1) \cdot \overline{c}_k, \tag{15}$$

Given that μ^k is proper, its cost-to-go function J_{μ^k} must be finite, that is, $J_{\mu^k}(i) < \infty, \forall i \in S$. Thus there exist one or more finite upper-bound functions $\overline{J}_{\mu^k}$ such that $J_{\mu^k}(i) \leq \overline{J}_{\mu^k}(i) < \infty$, for all $i \in S$. Given an upper-bound function $\overline{J}_{\mu^k}$ and the fact that action costs are positive, we can derive upper bounds on the expected number of stages until termination when following policy μ^k, as follows:

$$\overline{N}_k(i) = \frac{\overline{J}_{\mu^k}(i)}{\underline{g}}. \tag{16}$$

Using (16) in the bounds of (15), we can compute another upper bound $\overline{J}'_{\mu^k}(i)$ as follows:

$$\overline{J}'_{\mu^k}(i) = J_k(i) + \left(\frac{\overline{J}_{\mu^k}(i)}{\underline{g}} - 1\right) \cdot \overline{c}_k. \tag{17}$$

Repeating this procedure recursively would eventually lead to a fixed point that satisfies the following linear equation:

$$\overline{J}_{\mu^k}(i) = J_k(i) + \left(\frac{\overline{J}_{\mu^k}(i)}{\underline{g}} - 1\right) \cdot \overline{c}_k. \tag{18}$$

Solving for the fixed point $\overline{J}_{\mu^k}(i)$ directly, we get the upper bound of Equation (13). □

The results of Theorem 1 may seem surprising at first because the Bertsekas bounds of (8) require solving a system of $|S|$ linear equations in $|S|$ unknowns, whereas the bounds of Theorem 1 can be computed immediately from J_k and $\overline{c}_k$. This simplification is possible because the condition that all action costs are positive links the problem of computing $N_k(i)$ to the problem of computing $J_{\mu^k}(i)$. In fact, when action costs are uniform, we have $N_k(i) = J_{\mu^k}(i)$, for all $i \in S$. Even when action costs are non-uniform, as long as they are positive, a bound on $N_k(i)$ can be used to bound $J_{\mu^k}(i)$, based on (15), and a bound on $J_{\mu^k}(i)$ can be used to bound $N_k(i)$, based on (16). It is the link between these problems that allows the bounds to be computed efficiently.

Although the bounds of Theorem 1 apply whenever action costs are positive, they are tightest when action costs are uniform or nearly uniform. If the smallest action cost $\underline{g}$ is much smaller than the average action cost, the bound of Equation (16) will be looser, making the error bounds looser.

Note also that although the condition $\overline{c}_k < \underline{g}$ ensures that a greedy policy μ^k is proper, it does not follow that μ^k is improper when $\overline{c}_k \geq \underline{g}$. It is possible to construct examples where a greedy policy is proper and yet $\overline{c}_k \geq \underline{g}$.

Finally, the bounds of Theorem 1 remain valid under the Gauss-Seidel updates of Equation (7), although a potential increase in $\overline{c}_k$ as a result of Gauss-Seidel updates may weaken the bounds.

We next formulate a termination condition for value iteration based on the new bounds.

Theorem 2. *For an SSP problem where all actions taken in a non-goal state have positive cost, let* $\underline{g} = \min_{i\in S, a\in A(i)} g_i^a$ *denote the smallest action cost, and consider a lower-bound function* J_k *that is updated by the value iteration recursion of* (5)*, where* $\overline{c}_k$ *is the Bellman residual defined by* (9)*. For any* $\epsilon > 0$*, a greedy policy* μ^k *with respect to* J_{k-1} *is* ϵ*-optimal when both:*

1. $\overline{c}_k < \underline{g}$*, and*
2. *for all* $i \in S$*,*

$$\overline{c}_k \leq \frac{\epsilon \cdot \underline{g}}{J_k(i) - \underline{g} + \epsilon}. \tag{19}$$

Proof. From Theorem 1, the condition $\overline{c}_k < \underline{g}$ ensures that a greedy policy μ^k is proper. Given the bounds of (13), μ^k is ϵ-optimal when the following condition holds for all $i \in S$:

$$\frac{(J_k(i) - \overline{c}_k) \cdot \underline{g}}{(\underline{g} - \overline{c}_k)} - J_k(i) \leq \epsilon. \tag{20}$$

Solving (20) for $\overline{c}_k$ on the left-hand side gives (19). □

The Bellman residual $\overline{c}_k$ that is needed to ensure convergence to an ϵ-optimal policy depends on $\max_{i\in S} J_k(i)$, which increases in the course of the algorithm. Thus the value of $\overline{c}_k$ needed for convergence decreases in the course of the algorithm, and we don't know exactly how small $\overline{c}_k$ needs to be until convergence.

Heuristic search

Theorems 1 and 2 provide bounds and a convergence test for value iteration. We next discuss how to incorporate them in heuristic search algorithms for SSP problems.

Whereas value iteration finds a *complete policy*, that is, a mapping from every state $i \in S$ to an action $a \in A(i)$, heuristic search algorithms find a *partial policy* that is defined only for a subset of the states that are reachable from the start state. Reflecting this difference, the concepts of "proper" and "optimal" policy need to be qualified so that they are defined with reference to the start state. We say that a partial policy μ is *proper* for the start state if it ensures that a goal state is reached from the start state s_0 with probability 1. We say that a partial policy μ is *optimal* for the start state if it coincides with an optimal policy for all states that are reachable from the start state s_0 under the policy μ.

As with other heuristic search algorithms, we distinguish between a set of closed (or expanded) states, denoted *CLOSED*, a set of open (or fringe) states, denoted *OPEN*, and states that have not yet been visited. For a state in *CLOSED*, all of its successor states are in either *CLOSED* or *OPEN*. A cost-to-go function J is only defined for states in *CLOSED*, and the value of states in *OPEN* is given by an admissible heuristic function $h : S \rightarrow \Re$.

Let $S^{\mu}_{s_0} \subseteq S$ denote the subset of open and closed states that are reachable from the start state s_0 under the partial policy μ. We say that a policy μ is *closed* if all states in $S^{\mu}_{s_0}$ are closed. Otherwise, we say that it is *open*. Heuristic search algorithms for SSP problems start with an open policy that is defined only for the start state, and gradually *expand* and improve the policy until it is closed, proper and optimal. Expanding an open policy μ involves identifying *open states* in $S^{\mu}_{s_0}$ and expanding them, which means evaluating them and moving them from *OPEN* to *CLOSED*. This may change the best partial policy for the start state s_0, in which case a better policy is identified, and the process continues.

Algorithm 1 shows pseudocode for a variant of the LAO* algorithm (Hansen and Zilberstein 2001). In each iteration, LAO* performs a depth-first traversal of the states reachable from the start state under the current policy in order to identify states in the set $S^{\mu}_{s_0}$, and updates the policy and cost-to-go function for these states. Instead of updating the policy and cost-to-go function for a state *after* considering its descendants (that is, in post-order traversal), as in the variant of LAO* described by Hansen and Zilberstein (2001), Algorithm 1 updates the policy and cost-to-go function for a state *before* considering its descendants (in pre-order traversal). Pre-order updates ensure that the Bellman residual always reflects the current policy. But post-order updates tend to improve the cost-to-go function faster. Thus Algorithm 1 performs an additional post-order update of the cost-to-go function. In the pseudocode, $Succ(i, a)$ denotes the set of possible successor states of state i after taking action a. Since the Bellman residual is not defined for an open policy, it is assigned the value nil when the current policy is open.

The primary change from the original algorithm is the convergence test. Note that the bound only needs to be computed for the start state s_0 to test for convergence. By contrast, value iteration must compute the bound for all states, or, equivalently, for the state i for which $J_k(i)$ is largest.

Algorithm 1: LAO^* with new convergence test

```
LAO*($s_0$)
begin
    Add $s_0$ to OPEN and let $k \leftarrow 0$
    repeat
        repeat
            $k \leftarrow k + 1$
            $\bar{c}_k \leftarrow$ LAO*-REC($s_0$)
        until $\bar{c}_k \neq nil$ // nil indicates open policy
        if $\bar{c}_k < \underline{g}$ then
            $\overline{J}_{\mu^k}(s_0) \leftarrow (J_k(s_0) - \bar{c}_k) \cdot \underline{g}/(\underline{g} - \bar{c}_k)$
        else
            $\overline{J}_{\mu^k}(s_0) \leftarrow \infty$
    until $(\overline{J}_{\mu^k}(s_0) - J_k(s_0)) < \epsilon$

LAO*-REC($i$)
begin
    if $i$ is a goal state then
        return 0
    if $i$ is in OPEN then
        Move $i$ from OPEN to CLOSED
        for each action $a \in A(i)$ do
            for each $j \in Succ(i, a)$ do
                if $j$ is not in OPEN or CLOSED then
                    add $j$ to OPEN
                    $J_k(j) \leftarrow h(j)$ // h is admissible
        return nil
    $\mu^k(i) \leftarrow \arg\min_{a \in A(i)} [g^a_i + \sum_{j \in S} P^a_{ij} J_{k-1}(j)]$
    $J_k(i) \leftarrow \min_{a \in A(i)} [g^a_i + \sum_{j \in S} P^a_{ij} J_{k-1}(j)]$
    $residual \leftarrow J_k(i) - J_{k-1}(i)$ // local variable
    for each $j \in Succ(i, \mu^k(i))$ do
        if $j$ not already updated this iteration then
            $r \leftarrow$ LAO*-REC($j$)
            if $(r = nil)$ then
                $residual \leftarrow nil$
            else if $(residual \neq nil)$ and
            $(r > residual)$ then
                $residual \leftarrow r$
    $J_k(i) \leftarrow \min_{a \in A(i)} [g^a_i + \sum_{j \in S} P^a_{ij} J_k(j)]$
    return $residual$
```

Although we use LAO* to illustrate the new convergence test, it is straightforward to integrate the convergence test in any algorithm that computes the Bellman residual. For example, Bonet and Geffner describe two algorithms that adopt a similar approach as LAO*, called HDP (2003a) and LDFS (2006), and a related algorithm based on RTDP, called Labeled RTDP (2003b). All use a solve-labeling procedure to accelerate convergence. Our bounds and convergence test can easily be integrated into this solve-labeling procedure.

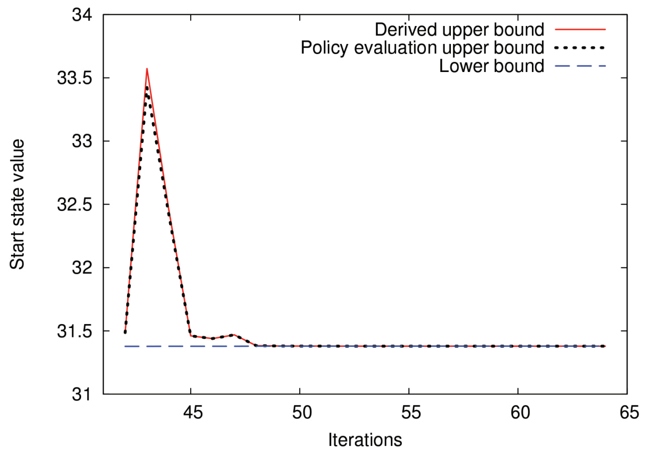

Figure 1: Bounds for racetrack problem.

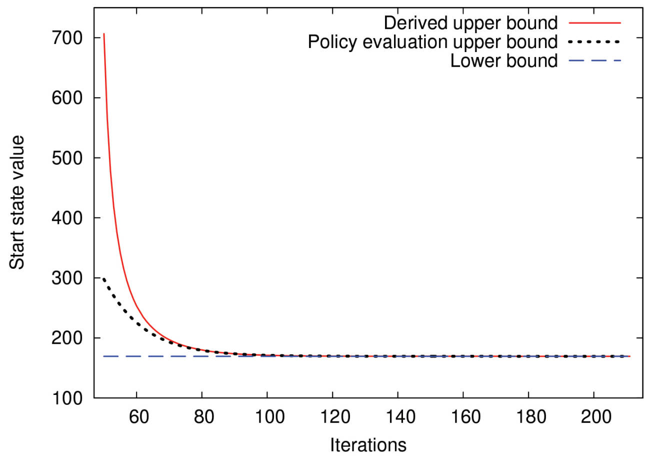

Figure 2: Bounds for mountain car problem.

Empirical performance of bounds

We implemented the new bounds and convergence test in the variant of the LAO* algorithm given by Algorithm 1, and tested its performance in solving some well-known benchmarks. We used LAO* for our experiments in part because of its simplicity, but primarily because it updates all states in $S^{\mu}_{s_0}$ each iteration, including the start state s_0, which lets us show how the bounds for the start state improve over successive iterations of the algorithm. For an admissible heuristic, we solved a deterministic relaxation of the problem.

Figure 1 shows its performance in solving the racetrack problem used as a test example by Barto et al. (1995), Hansen and Zilberstein (2001), and others. The problem has 21,371 states and 9 actions per state, and an optimal policy for the start state visits only 2,262 states. The problem has unit action costs, like our other test problems.

For the first 41 iterations of LAO*, the greedy policy is an open policy except at iterations 34, 35, and 39, where it is closed, but the residual $\bar{c}_k$ is greater than 1; thus the greedy policy is not guaranteed to be proper. After 42 iterations, LAO* has found a (provably) proper policy for which bounds can be computed. After 64 iterations, it has found an ϵ-optimal policy with $\epsilon = 10^{-6}$.

Figure 1 shows three bounds. The top line is the upper bound $\overline{J}_{\mu^k}(s_0)$ of (13). The middle line (which is almost indistinguishable from the top line) is the upper bound $J_{\mu^k}(s_0)$ computed by evaluating the current greedy policy μ^k. In practice, we would not compute this bound because it requires solving a system of linear equations. It is only included for the purpose of comparison. It shows that although $\overline{J}_{\mu^k}(s_0)$ is only guaranteed to be an upper bound on $J_{\mu^k}(s_0)$, it is so tight as to be almost equal to $J_{\mu^k}(s_0)$ for this problem. The bottom line is the lower bound $J_k(s_0)$.

Interestingly, the upper bounds do not decrease monotonically. When used with value iteration, the bound $\overline{J}_{\mu^k}$ *does* decrease monotonically because the Bellman residual decreases monotonically when it is computed for the entire state space. But for a heuristic search algorithm like LAO*, the Bellman residual is only computed for states visited by the current greedy policy. It decreases monotonically as long as the greedy policy does not change. But when it changes, the policy can visit different states for which the Bellman residual is larger. The spikes in the graph of Figure 1 corresponds to iterations where the greedy policy changes to a policy that visits different states, and is worse. Since the upper bound of (13) is not guaranteed to decrease monotonically when used with a heuristic search algorithm like LAO*, the following update ensures the best possible upper bound at each iteration, $J^u_k(i) = \min\{J^u_{k-1}(i), \overline{J}_{\mu^k}(i)\}$, where $J^u_k(i)$ denotes the best upper bound for state i found so far. But while J^u_k is an upper bound on J^*, it does not bound the sub-optimality of the current greedy policy.

Figure 2 shows the performance of the bounds in solving the mountain car problem from Wingate and Seppi (2005). The problem has 250,000 states with two actions per state, and an optimal policy for the start state visits 82,910 states. A (provably) proper policy is found at iteration 49, and an ϵ-optimal policy (with $\epsilon = 10^{-6}$) is found at iteration 263.

The closer the residual $\bar{c}_k$ is to $\underline{g}$, while still less than $\underline{g}$, the farther apart are the bounds $J_{\mu^k}(s_0)$ and $\overline{J}_{\mu^k}(s_0)$. This is illustrated by a comparison of the initial bounds for the racetrack and mountain car problems. For the mountain car problem, the first residual less than 1 is $\bar{c}_{49} = 0.83$. For the racetrack problem, it is $\bar{c}_{42} = 0.0036$. As a result, the bounds $J_{\mu^k}(s_0)$ and $\overline{J}_{\mu^k}(s_0)$ are initially far apart for the mountain car problem, but close together for the racetrack problem. Of course, the closer the residual is to 0, the closer are $J_{\mu^k}(s_0)$ and $\overline{J}_{\mu^k}(s_0)$, and they are identical when $\bar{c}_k = 0$.

Figure 3 shows the performance of the bounds in solving a double-armed pendulum problem from Wingate and Seppi (2005). The problem has 160,000 states and 2 actions per state. An optimal policy for the start state visits 146,153 states. A (provably) proper policy is found after 16 iterations, and an ϵ-optimal policy (with $\epsilon = 10^{-6}$) is found after 54 iterations. At iteration 16, the Bellman residual is 0.41 and the two upper bounds are 54.1 and 64.9 respectively. At iteration 21, the residual is 0.05, and the difference between the two upper bounds is almost indistinguishable after that.

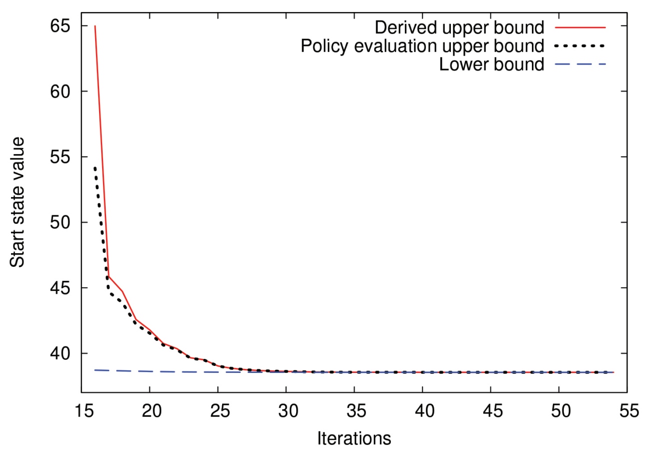

Figure 3: Bounds for double-armed pendulum problem.

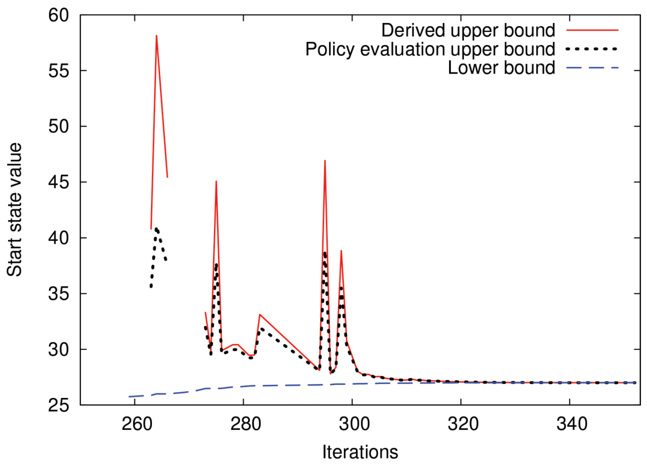

Figure 4: Bounds for elevator problem.

Finally, Figure 4 shows the performance of the bounds in solving instance 10 of the Elevator problem from the 2006 International Planning Competition. For this problem, 14,976 states are reachable from the start state and there are an average of 5 actions per state. Although an optimal policy visits only 18 states, LAO* takes 217 iterations to converge to a (provably) proper policy and 400 iterations to converge to an ϵ-optimal policy (with $\epsilon = 10^{-6}$). Interestingly, after LAO* first finds a proper policy for which it can compute error bounds, several iterations follow during which the current greedy policy is not (provably) proper before LAO* again finds a policy for which it can compute bounds. This is illustrated by a gap in the upper bounds shown in Figure 4.

Extensions

All of the test problems used in our experiments have unit action costs, which is characteristic of test problems found in the literature, or used in the ICAPS planning competitions. For problems with non-uniform action costs, it is worth noting that Theorem 1 can be reformulated so that its result depends not on the smallest action cost for the MDP, but on the smallest cost of any action used in the current policy μ. When action costs are non-uniform, this reformulation of Theorem 1 could be especially useful for heuristic search algorithms, which only compute bounds for the states in $S^{\mu}_{s_0}$. Although this reformulation would require checking in each iteration for the smallest cost of any action used in the current policy, it has two potential advantages. First, if the smallest cost of an action used by the current policy is greater than the smallest action cost for the entire MDP, the bounds will be tighter. Second, since heuristic search algorithms only explore a small fraction of the state space, it means the bounds can be computed by only considering costs in this fraction of the state space, without needing to identify the smallest action cost in the entire MDP.

It is one of several extensions of this approach that we plan to consider in future work, including using the bounds for action elimination, testing the bounds in other algorithms, and potentially generalizing the bounds.

Conclusion

We have introduced a simple and efficient method for computing bounds for value iteration when solving SSP problems. Because the method assumes the cost-to-go function is a lower bound, it is especially useful in heuristic search algorithms that improve an admissible evaluation function. Although Bertsekas (2005, pp. 413–414) derives closely-related bounds for SSP problems, including the bounds of (8), his bounds require solving a system of $|S|$ equations in $|S|$ unknowns, severely limiting their usefulness, whereas our bounds do not require any extra computation besides computing the Bellman residual. In a paper that complements our results, Hansen (2011) derives bounds that can also be computed efficiently for SSP problems with positive action costs, but only if the cost-to-go function is a monotone *upper* bound. By contrast, our approach applies when the cost-to-go function is a lower bound, allowing it to be used in heuristic search algorithms for SSP problems. Bonet (2007) considers SSP problems with positive action costs and identifies the problem of bounding the suboptimality of a greedy policy with respect to a cost-to-go function, based only on the Bellman residual, as an "important open problem." Our results provide a simple solution.

In the past, heuristic search algorithms for SSP problems have either not provided meaningful bounds, or have only provided them at significant extra expense. A common test for convergence is a test for ϵ-consistency; the algorithm terminates when the Bellman residual is less than some $\epsilon > 0$. Although this convergence test provides no guarantee of solution quality, as we have already pointed out, it often works very well in practice because there is, in fact, a problem-dependent $\epsilon > 0$ such that a solution is "good enough" when the Bellman residual is less than ϵ. Our contribution is to provide a principled and efficient way of determining how small the Bellman residual needs to be to ensure that a greedy policy is both proper and ϵ-optimal.

Acknowledgements This research was partially supported by NSF grant IIS-1219114.

References

Barto, A.; Bradtke, S.; and Singh, S. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 72(1):81–138.

Bertsekas, D., and Tsitsiklis, J. 1991. Analysis of stochastic shortest path problems. *Mathematics of Operations Research* 16(3):580–595.

Bertsekas, D. 2005. *Dynamic Programming and Optimal Control, Vol. 1*. Belmont, MA: Athena Scientific, 3rd edition.

Bertsekas, D. 2012. *Dynamic Programming and Optimal Control, Vol. 2*. Belmont, MA: Athena Scientific, 4th edition.

Bonet, B., and Geffner, H. 2003a. Faster heuristic search algorithms for planning with uncertainty and full feedback. In *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence (IJCAI-03)*, 1233–1238. Morgan Kaufmann.

Bonet, B., and Geffner, H. 2003b. Labeled RTDP: Improving the convergence of real-time dynamic programming. In *Proc. of the 13th Int. Conf. on Automated Planning and Scheduling (ICAPS-03)*, 12–21. AAAI Press.

Bonet, B., and Geffner, H. 2006. Learning depth-first search: A unified approach to heuristic search in deterministic and non-deterministic settings, and its application to MDPs. In *Proc. of the 16th Int. Conf. on Automated Planning and Scheduling (ICAPS-06)*, 142–151. AAAI Press.

Bonet, B. 2007. On the speed of convergence of value iteration on stochastic shortest-path problems. *Mathematics of Operations Research* 32(2):365–373.

Hansen, E., and Zilberstein, S. 2001. LAO*: A heuristic search algorithm that finds solutions with loops. *Artificial Intelligence* 129(1–2):139–157.

Hansen, E. 2011. Suboptimality bounds for stochastic shortest path problems. In *Proc. of the 27th Conf. on Uncertainty in Artificial Inteliigence*, 301–310. AUAI Press.

Kallenberg, L. 2002. Finite state and action MDPs. In Feinberg, E., and Shwartz, A., eds., *Handbook of Markov Decision Processes: Methods and Applications*. Boston: Kluwer Academic Publishers.

McMahan, H. B.; Likhachev, M.; and Gordon, G. 2005. Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees. In *Proc. of the 22nd Int. Conf. on Machine Learning (ICML-05)*, 569–576. ACM.

Sanner, S.; Goetschalckx, R.; Driessens, K.; and Shani, G. 2009. Bayesian real-time dynamic programming. In *Proc. of the 21st Int. Joint Conf. on Artificial Intelligence (IJCAI-09)*, 1784–1789. AAAI Press.

Smith, T., and Simmons, R. G. 2006. Focused real-time dynamic programming for MDPs: Squeezing more out of a heuristic. In *Proc. of the 21st National Conf. on Artificial Intelligence (AAAI-06)*, 1227–1232. AAAI Press.

Warnquist, H.; Kvarnström, J.; and Doherty, P. 2010. Iterative bounding LAO*. In *Proc. of 19th European Conference on Artificial Intelligence (ECAI-10)*, 341–346. IOS Press.

Wingate, D., and Seppi, K. 2005. Prioritization methods for accelerating MDP solvers. *J. of Machine Learning Research* 6:851–881.