Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

General Error 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

We consider recently-derived error bounds that can be used to bound the quality of solutions found by heuristic search algorithms for stochastic shortest path problems. In their original form, the bounds can only be used for problems with positive action costs. We show how to generalize the bounds so that they can be used in solving any stochastic shortest path problem, regardless of cost structure. In addition, we introduce a simple new heuristic search algorithm that performs as well or better than previous algorithms for this class of problems, while being easier to implement and analyze.

Introduction

Decision-theoretic planning problems are often modeled as stochastic shortest path (SSP) problems. For SSP problems, actions have probabilistic outcomes, and the objective is to find a conditional plan, or policy, that reaches a goal state from the start state with minimum expected cost. Classic dynamic programming algorithms find a solution for the entire state space, that is, for all possible start states. By contrast, heuristic search algorithms find a conditional plan for a given start state, and can do so by evaluating only a fraction of the state space. The heuristic search approach to solving SSP problems generalizes the approach of A* and related heuristic search algorithms for solving deterministic shortest-path problems, which also limit the number of states that must be evaluated in the search for an optimal path from a given start state to a goal state.

The heuristic search approach to solving SSP problems is 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 algorithm that only visits and updates states that are reachable from the start state under the current greedy policy converges *in the limit* to the optimal cost-to-go function for these states, without necessarily evaluating the full state space. Two broad strategies for using heuristic search in solving SSP problems have been studied. Real-time dynamic programming (RTDP) uses trial-based search methods that generalize real-time heuristic search techniques for deterministic shortest-path problems (Barto, Bradtke, and Singh 1995; McMahan, Likhachev, and Gordon 2005; Smith and Simmons 2006; Sanner et al. 2009). An alternative approach generalizes traditional AND/OR graph search techniques (Hansen and Zilberstein 2001; Bonet and Geffner 2003a; 2006; Warnquist, Kvarnström, and Doherty 2010). Whereas RTDP samples the reachable state space by repeated simulated trajectories from the start state, the second approach systematically explores the reachable state space by repeated depth-first traversals from the start state. Some algorithms use a mix of the two strategies (Bonet and Geffner 2003b).

The systematic approach to heuristic search shares with traditional value iteration the important advantage that it computes a *residual* each iteration, where the residual is equal to the largest improvement in value for any (reachable) state. Computation of the residual is useful because it can be used to test for convergence. The smaller the residual, the higher the quality of the solution. When the residual is equal to zero, the solution is optimal. Because the residual only converges to zero in the limit, however, it is common practice to test for convergence by testing for an ϵ*-consistent* solution, which is a solution for which the residual is less than some threshold $\epsilon > 0$. Until recently, however, this approach has lacked a principled way of selecting a threshold ϵ that provides a bound on the suboptimality of the solution.

Hansen and Abdoulahi (2015) recently derived the first easily-computed error bounds for SSP problems, where the bounds are based only on the residual and the cost-to-go function. The bounds perform very well in practice. But the approach has two limitations. First, it is only applicable if all action costs are positive. Second, the bounds are tightest if action costs are uniform, or nearly uniform. For problems with non-uniform action costs, the quality of the error bounds decreases, roughly in proportion to the difference between the smallest action cost and the average action cost.

In this paper, we overcome both limitations by showing how to compute error bounds that are equally good whether action costs are uniform or not, and are available regardless of the cost structure of the problem; in particular, the bounds do not depend on all action costs being positive. In discussing how to incorporate these bounds in different heuristic search algorithms, we also introduce a simple new heuristic search algorithm that we show performs as well or better than previous heuristic search algorithms for SSP problems, and is easier to implement and analyze.

Stochastic shortest path problem

A stochastic shortest path (SSP) problem (Bertsekas and Tsitsiklis 1991) is a discrete-stage infinite-horizon Markov decision process (MDP) with the following elements:

- S denotes a finite set of non-goal states, with initial state $s_0 \in S$, and G denotes a non-empty set of goal states;
- for each state $i \in S \cup G$, $A(i)$ denotes a finite, non-empty set of feasible actions;
- P_{ij}^a denotes the probability that action $a \in A(i)$ taken in state i results in a transition to state j; and
- $g_i^a \in \Re$ 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, which means $g_i^a = 0$ and $P_{ii}^a = 1$, for all $i \in G, a \in A(i)$.

Let M denote the 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 within some finite k number of stages 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 $i \in S$.

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 the policy μ starting from any state i. It is the solution of the following system of $|S|$ linear equations in $|S|$ unknowns,

$$J_\mu(i) \quad = \quad g_i^{\mu(i)} + \sum_{j \in S} P_{ij}^{\mu(i)} J_\mu(j), \; i \in S, \tag{1}$$

where $J_\mu(i) = 0$, for all $i \in G$. The optimal cost-to-go function J^* is defined as $J^*(i) = \min_{\mu \in M} J_\mu(i), \; i \in S$, and an optimal policy μ^* satisfies

$$J^*(i) = J_{\mu^*}(i) \leq J_\mu(i), \quad \mu \in M, i \in S. \tag{2}$$

Bertsekas and Tsitsiklis (1991) show that an optimal policy is proper under the following assumptions: (i) there is at least one proper policy, and (ii) any improper policy incurs positive infinite cost for some initial state.

Given an initial cost vector J_0, value iteration generates an improved cost vector J_k each iteration $k = 1, 2, 3, \ldots$, by updating the value of every state $i \in S$ as follows,

$$J_k(i) = \min_{a \in A(i)} \left\{ g_i^a + \sum_{j \in S} P_{ij}^a J_{k-1}(j) \right\}, \tag{3}$$

where $J_k(i) = 0$, for $i \in G$. A *greedy policy* μ^k with respect to a cost vector J_{k-1} is defined as a policy that selects, for each state $i \in S$, the action that maximizes the right-hand size of Equation (3). It is common to refer to the update of a single state value based on (3) as a *backup*, and the update of all state values based on (3) as a *dynamic programming update*. For SSP problems, Bertsekas and Tsitsiklis (1991) prove that for any initial cost vector $J_0 \in \Re^{|S|}$, 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^*, 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, where $J_k(j)$ is used in place of $J_{k-1}(j)$ on the right-hand side of (3), whenever $J_k(i)$ is already available. If the states in S are indexed from 1 to n, with $n = |S|$, a Gauss-Seidel update can be defined for each state i as,

$$J_k(i) = \min_{a \in A(i)} \left\{ g_i^a + \sum_{j=1}^{i-1} P_{ij}^a J_k(j) + \sum_{j=i}^{n} P_{ij}^a J_{k-1}(j) \right\}. \tag{4}$$

The standard update of (3) is called a Jacobi update.

Error bounds

Although value iteration converges to an optimal solution in the limit, it must be terminated after a finite number of iterations, in practice. Therefore, it is useful to be able to bound the sub-optimality of a solution. A policy μ^k is said to be *ϵ-optimal* if $J_{\mu^k}(i) - J^*(i) \leq \epsilon, \forall i \in S$, where J_{μ^k} is the cost-to-go function of the policy μ^k. By assumption, an improper policy has positive infinite cost for some initial state. Therefore, only a proper policy can be ϵ-optimal.

To test for the ϵ-optimality of a policy μ^k in practice, we need a lower bound on J^* and an upper bound on J_{μ^k}. In this paper, we assume that the initial cost-to-go function J_0 is a lower bound, that is, $J_0(i) \leq J^*(i), \forall i \in S$. By a simple inductive argument, it follows that each updated cost-to-go function J_k, for $k \geq 1$, is also a lower bound. Therefore, a greedy policy μ^k with respect to J_{k-1} is ϵ-optimal if $J_{\mu^k}(i) - J_k(i) \leq \epsilon, \forall i \in S$. Given the lower bound J_k, we still need an upper bound on J_{μ^k} in order to have a practical test for the ϵ-optimality of μ^k.

Bertsekas bounds

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 $i \in S$,

$$J_k(i) \leq J^*(i) \leq J_{\mu^k}(i) \leq J_k(i) + (N_{\mu^k}(i) - 1) \cdot \overline{c}_k, \tag{5}$$

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

$$\overline{c}_k = \max_{i \in S} \{J_k(i) - J_{k-1}(i)\}, \tag{6}$$

is the residual.[1] Bertsekas' derivation of these bounds assumes the standard Jacobi dynamic programming update of (3). For a Gauss-Seidel dynamic programming update, the error bounds of (5) remain valid under the assumption that J_k is a lower-bound function. However, establishing this requires an additional proof that is not given by Bertsekas. We give a proof in an appendix. (The proof also considers the general case where J_k is not a lower-bound function.)

Given a cost vector J_k that is a lower-bound function, all we need to bound the sub-optimality of a greedy policy μ^k is

[1]Bertsekas also defines a complimentary residual, $\underline{c}_k = \min_{i \in S} \{J_k(i) - J_{k-1}(i)\}$, that can be used to compute a lower-bound function. Since we assume that J_k itself is a lower-bound function, we do not use this residual.

to compute either one of the two upper bounds on the right-hand side of (5). If we know that μ^k is a proper policy, 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 the system of $|S|$ linear equations in $|S|$ unknowns given by (1). The other way to get an upper bound is by using the inequality $J_{\mu^k}(i) \leq J_k(i) + (N_{\mu^k}(i) - 1) \cdot \overline{c}_k$ from (5). But determining $N_{\mu^k}(i)$, which is the expected number of stages to reach a goal state starting from state i and following the policy μ^k, requires solving the following system of $|S|$ linear equations in $|S|$ unknowns:

$$N_{\mu^k}(i) \;=\; 1 + \sum_{j \in S} P_{ij}^{\mu^k(i)} N_{\mu^k}(j),\; i \in S. \tag{7}$$

Computing these values is as expensive as policy evaluation.

"Unfortunately," writes Bertsekas (2005, p. 414), the bounds of (5) "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. By a well-known reduction, any discounted infinite-horizon Markov decision problem can be reduced to an equivalent SSP problem, where $N_{\mu^k}(i) - 1 = \beta/(1-\beta)$ for all $\mu \in M, i \in S$. In the discounted case, the Bertsekas bounds of (5) reduce to the following well-known bounds (still assuming that J_k is a lower bound):

$$J_k(i) \leq J^*(i) \leq J_{\mu^k}(i) \leq J_k(i) + \left(\frac{\beta}{1-\beta}\right) \cdot \overline{c}_k. \tag{8}$$

Except for the special case of discounting, the Bertsekas bounds of (5) are too expensive to be useful in practice.

Positive-cost bounds

Hansen and Abdoulahi (2015) recently derived practical error bounds for SSP problems with positive action costs. The bounds are, in fact, bounds on the Bertsekas bounds, but have the advantage that they can be computed easily.

Theorem 1. *(Hansen and Abdoulahi 2015) For an SSP problem where all actions taken in a non-goal state have positive cost, and* $\underline{g} = \min_{i \in S, a \in A(i)} g_i^a$ *denotes the smallest action cost, if* $\overline{c}_k < \underline{g}$ *then:*

(a) a greedy policy μ^k *with respect to* J_{k-1} *is proper, and*

(b) for each state $i \in S$*, we have the following upper bound, where* $J_{\mu^k}(i) \leq \overline{J}_{\mu^k}(i)$*:*

$$\overline{J}_{\mu^k}(i) = \frac{(J_k(i) - \overline{c}_k) \cdot \underline{g}}{(\underline{g} - \overline{c}_k)}. \tag{9}$$

The upper bound given by (9) is easy to compute because it depends only on the quantities $J_k(i)$ and $\overline{c}_k$, and not also on the difficult-to-compute quantity $N_{\mu^k}(i)$ that is needed for the Bertsekas upper bound of (5). We refer to the paper of Hansen and Abdoulahi (2015) for a full and formal proof of this theorem, and just briefly review one of its key ideas.

The derivation of (9) is based on the insight that when all action costs are positive, with minimum cost $\underline{g} > 0$, an upper bound $\overline{N}_{\mu^k}(i)$ on $N_{\mu^k}(i)$ is related to an upper bound $\overline{J}_{\mu^k}(i)$ on $J_{\mu^k}(i)$ by the formula:

$$\overline{N}_{\mu^k}(i) = \frac{\overline{J}_{\mu^k}(i)}{\underline{g}}. \tag{10}$$

This formula simply states that an upper bound $\overline{N}_{\mu^k}(i)$ on the expected number of steps until termination when following a policy μ^k starting from state i is given by an upper bound $\overline{J}_{\mu^k}(i)$ on the cost-to-go $J_{\mu^k}(i)$ divided by the smallest action cost $\underline{g}$. Given this formula, the bound of (9) is derived by substituting $\overline{N}_{\mu^k}(i)$ for $N_{\mu^k}(i)$ in the Bertsekas bound of (5) to obtain the upper bound,

$$\overline{J}_{\mu^k}(i) = J_k(i) + (\overline{N}_{\mu^k}(i) - 1) \cdot \overline{c}_k, \tag{11}$$

and then substituting $\overline{J}_{\mu^k}(i)/\underline{g}$ for $\overline{N}_{\mu^k}(i)$ based on (10), and solving for $\overline{J}_{\mu^k}(i)$. Note that from (9) and (10), we have

$$\overline{N}_{\mu^k}(i) = \frac{(J_k(i) - \overline{c}_k)}{(\underline{g} - \overline{c}_k)}. \tag{12}$$

New bounds for the general case

The bounds of Theorem 1 are only available if all action costs are positive. Moreover, even when all action costs are positive, the quality of the bounds decreases when action costs are not uniform. Their quality decreases because the ratio $\overline{J}_{\mu^k}(i)/\underline{g}$ in Equation (10) increases with the difference between the smallest action cost and the average action cost, and the resulting increase in $\overline{N}_{\mu^k}(i)$ loosens the bounds.

These limitations are related to the fact that the bounds of Theorem 1 use the value of $J_k(i)$ to compute $\overline{N}_{\mu^k}(i)$, as shown by Equation (12). We next show how to compute $\overline{N}_{\mu^k}(i)$ independently of $J_k(i)$, and thus in a way that does not depend on the cost structure of the problem.

Consider a *steps-to-go function* $N_k(i)$ that estimates the number of steps, or stages, required to reach a goal state from state i. Consider also a value iteration algorithm that performs the following update,

$$N_k(i) = 1 + \sum_{j \in S} P_{ij}^{\mu^k(i)} N_{k-1}(j), \tag{13}$$

after each backup that computes $J_k(i)$ and $\mu^k(i)$ for state i. This enhanced value iteration algorithm also computes the following residual after each iteration:

$$\overline{n}_k = \max_{i \in S} \left(N_k(i) - N_{k-1}(i)\right). \tag{14}$$

When all action costs are equal to 1, it is easy to see that $J_k(i) = N_k(i)$, for $i = 1, \ldots, n$, and $\overline{c}_k = \overline{n}_k$. In that case, there is no reason to compute these additional values. But when action costs are not uniform, or when they are not all positive, the additional values $N_k(i)$ and $\overline{n}_k$ can differ greatly from the values $J_k(i)$ and $\overline{c}_k$, and they provide a way to compute bounds of the same quality as those available when action costs are uniform and positive.

The following theorem assumes that the value iteration algorithm also computes a steps-to-go function.

Theorem 2. *For any SSP problem, consider a lower-bound function J_k that is updated by value iteration, where $\overline{c}_k$ is the residual defined by* (6)*. Consider also a steps-to-go function N_k that is updated each iteration, where $\overline{n}_k$ is the residual defined by* (14)*. If $\overline{n}_k < 1$ then:*

(a) a greedy policy μ^k with respect to J_{k-1} is proper, and

(b) for each state $i \in S$, we have the following upper bound on $J_{\mu^k}(i)$, where $J_{\mu^k}(i) \leq \overline{J}_{\mu^k}(i)$:

(i) If $0 \leq \overline{n}_k < 1$, then

$$\overline{J}_{\mu^k}(i) = J_k(i) + \left(\frac{N_k(i) - \overline{n}_k}{1 - \overline{n}_k} - 1 \right) \cdot \overline{c}_k. \quad (15)$$

(ii) If $\overline{n}_k \leq 0$, then

$$\overline{J}_{\mu^k}(i) = J_k(i) + (N_k(i) - 1) \cdot \overline{c}_k. \quad (16)$$

Proof. Part (a) follows by the same logic used in the proof of part (a) of Theorem 1. In that proof, the key observation is that the residual $\overline{c}_k$ is an upper bound on the average cost per stage for any state $i \in S$ under a greedy policy μ^k (Bertsekas 2012, p. 329). For an improper policy, there is at least one state from which a goal state is never reached, and its average cost per stage cannot be less than the smallest action cost $\underline{g}$. It follows that if $\overline{c}_k < \underline{g}$, the greedy policy μ^k must be proper.

Computing the steps-to-go function N_{μ^k} for a policy μ^k can be viewed a positive-cost SSP problem where the smallest action cost is 1, and thus the greedy policy μ^k must be proper when $\overline{n}_k < 1$, by the same reasoning.

We next consider part (b). Applying the Bertsekas bounds of (5) to the problem of computing N_{μ^k}, we have:

$$N_{\mu^k}(i) \leq N_k(i) + (N_{\mu^k}(i) - 1) \cdot \overline{n}_k. \quad (17)$$

By the same reasoning used to prove part (b) of Theorem 1, if μ^k is proper, there must be an upper bound $\overline{N}_{\mu^k}(i)$, with $N_{\mu^k}(i) \leq \overline{N}_{\mu^k}(i)$, that is the solution of the linear equation:

$$\overline{N}_{\mu^k}(i) = N_k(i) + (\overline{N}_{\mu^k}(i) - 1) \cdot \overline{n}_k. \quad (18)$$

Solving for $\overline{N}_{\mu^k}(i)$, we get

$$\overline{N}_{\mu^k}(i) = \frac{N_k(i) - \overline{n}_k}{1 - \overline{n}_k}. \quad (19)$$

Substituting the value of $\overline{N}_{\mu^k}(i)$ from (19) into (11), we get the bound of (15).

The bound of (15) is based on the Bertsekas bound of (5), which assumes Jacobi dynamic programming updates. In an appendix, we show that the Bertsekas upper bound of (5) also holds under Gauss-Seidel dynamic programming updates, provided the residuals $\overline{c}_k$ and $\overline{n}_k$ are both non-negative. The assumption that J_k is a lower-bound function ensures that $\overline{c}_k \geq 0$. But N_k is not necessarily a lower-bound function. If $\overline{n}_k$ is non-positive, however, N_k must be a monotone upper bound, which gives the bound of (16). □

For an SSP problem with positive action costs that are not uniform, the upper bounds of Theorem 2 are tighter, and potentially *much* tighter, than the upper bounds of Theorem 1. Moreover, Theorem 2 can be used to compute upper bounds for any SSP problem, even if action costs are zero or negative. It only requires the slight extra overhead of updating the steps-to-go function in each iteration of value iteration.

Heuristic search and bounds

We next consider how to integrate the new bounds in a heuristic search algorithm. To facilitate this discussion, we introduce a simplified version of LAO* (Hansen and Zilberstein 2001), which we call *Focused Value Iteration* (FVI). Algorithm 1 gives the pseudocode for the algorithm.

FVI updates a cost-to-go function over a sequence of iterations, like standard value iteration. But like LAO*, it only updates the cost-to-go function for the subset $S_{s_0}^{\mu^k} \subseteq S$ of states reachable from the start state s_0 under a greedy policy μ^k. In each iteration, it performs a depth-first traversal of the states in $S_{s_0}^{\mu^k}$, beginning from s_0. When a state $i \in S_{s_0}^{\mu^k}$ is first visited, a backup is performed and the best action $\mu^k(i)$ is identified. Then each successor state $j \in Succ(i, \mu^k(i))$ is pushed on the stack used to organize the depth-first traversal, provided the state has not already been visited this iteration. The variable $visit(j)$ indicates whether state j has been vis-

Algorithm 1: Focused Value Iteration with new bounds

```
Input: SSP problem, start state $s_0$, lower-bound function $J_0$
Output: $\epsilon$-optimal policy for start state $s_0$
Algorithm FVI ($s_0$)
    $k \leftarrow 0; \forall i \in \mathcal{S}, visit(i) = 0, N_0(i) = 0$
    repeat
        $k \leftarrow k + 1$ // Iteration counter
        $visit(s_0) \leftarrow k; \overline{c}_k \leftarrow \overline{n}_k \leftarrow -\infty$ // initialize
        FVIrec($s_0$) // depth-first traversal
        $\overline{J}_{\mu^k}(s_0) \leftarrow \infty$ // trivial default upper bound
        if $\overline{n}_k < 1$ then // test for proper policy
            if $\overline{n}_k < 0$ then
                $\overline{N}_{\mu^k}(s_0) \leftarrow N_k(s_0)$
            end
            else
                $\overline{N}_{\mu^k}(s_0) \leftarrow (N_k(s_0) - \overline{n}_k)/(1 - \overline{n}_k)$
            end
            $\overline{J}_k(s_0) \leftarrow J_k(s_0) + (\overline{N}_k(s_0) - 1) \cdot \overline{c}_k$
        end
    until ($\overline{J}_{\mu^k}(s_0) - J_k(s_0) < \epsilon$)

Function FVIrec($i$)
    // Pre-order backup
    $J_k(i) \leftarrow \min_{a \in A(i)}[g_i^a + \sum_{j \in S} P_{ij}^a J_{k-1}(j)]$
    $\mu^k(i) \leftarrow a$ // best action
    $\overline{c}_k \leftarrow \max\{\overline{c}_k, J_k(i) - J_{k-1}(i)\}$
    $N_k(i) \leftarrow 1 + \sum_{j \in S} P_{ij}^a N_{k-1}(j)$
    $\overline{n}_k \leftarrow \max\{\overline{n}_k, N_k(i) - N_{k-1}(i)\}$
    // Process unvisited descendents
    foreach $j \in Succ(i, \mu^k(i))$ do
        if ($visit(j) < k$) and ($j \notin G$) then
            $visit(j) \leftarrow k$
            FVIrec($j$)
        end
    end
    // Post-order backup
    $N_k(i) \leftarrow 1 + \sum_{j \in S} P_{ij}^{\mu^k(i)} N_k(j)$
    $J_k(i) \leftarrow \min_{a \in A(i)}[g_i^a + \sum_{j \in S} P_{ij}^a J_k(j)]$
    return
```

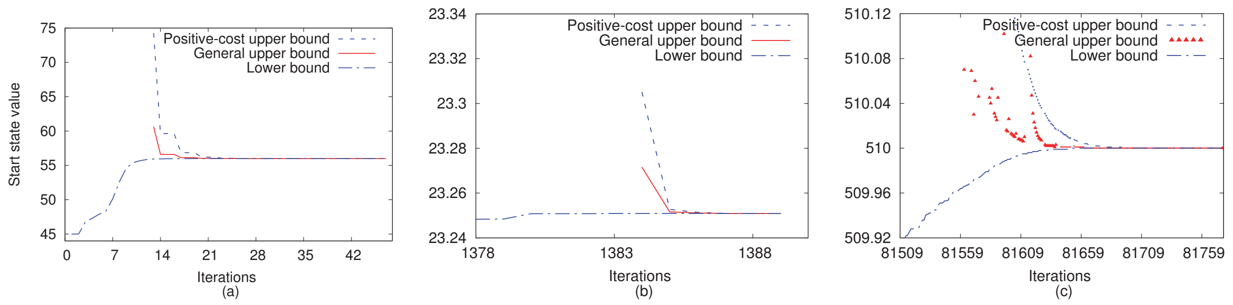

Figure 1: Convergence of bounds for (a) Tireworld problem, (b) Boxworld problem, and (c) Zenotravel problem.

ited yet in iteration k. At the conclusion of the traversal, a greedy policy μ^k has been found, and the cost-to-go function has been updated for all states in $S_{s_0}^{\mu^k}$.

The pseudocode of Algorithm 1 shows that FVI performs two backups per iteration for each state $i \in S_{s_0}^{\mu^k}$. The initial backup, performed when the state is first visited, identifies the best action for the state. The second backup, which is performed when backtracking from the state, further improves the cost-to-go function. In fact, the second backup tends to improve the cost-to-go function more than the first because it is performed after the successors of the state have been backed-up. But the second backup is not used to change the policy. The greedy policy is selected when states are first visited by the depth-first traversal to ensure that the set of states $S_{s_0}^{\mu^k}$ is exactly the set of states visited by following the greedy policy μ^k starting from s_0. The second backup is not used to compute the residual $\overline{c}_k$ either. It is computed based on the first backup only. (A residual $\overline{c}_k$ computed by post-order backups is valid if and only if the policy is not changed by the post-order backups.)

Although the residuals $\overline{c}_k$ and $\overline{n}_k$ are defined only for the states in $S_{s_0}^{\mu^k}$, they can be used by a heuristic search algorithm to test whether the policy μ^k is proper and ϵ-optimal *relative to the start state*. (We say that a policy is proper relative to the start state if it ensures that the goal state is reached with probability 1 from the start state. We say that a policy is ϵ-optimal relative to the start state if the expected cost-to-go of the start state under the policy is within ϵ of optimal.)

Corollary 1. *The bounds of Theorem 2 can be used in a heuristic search algorithm that computes the residuals $\overline{c}_k$ and $\overline{n}_k$ only for the states in $S_{s_0}^{\mu^k}$.*

Proof. Consider a restriction of the SSP problem where the state set is $S_{s_0}^{\mu^k}$, each state $i \in S_{s_0}^{\mu^k}$ has a singleton action set $A(i) = \{\mu(i)\}$, and transition probabilities and costs are the same as for the original SSP problem. For each state $i \in S_{s_0}^{\mu^k}$ and action $a \in A(i) = \{\mu^k(i)\}$, all possible successor states are in $S_{s_0}^{\mu^k}$, and so it is a well-defined MDP. The single policy μ^k is either proper or not. If proper, this restriction of the SSP problem is itself a well-defined SSP, and the bounds of Theorem 2 apply. If not proper, a goal state is not reachable from at least one state in $S_{s_0}^{\mu^k}$, and $\overline{n}_k \geq 1$. □

Performance of the new bounds

Figure 1 compares the performance of the new bounds of Theorem 2 to the positive-cost bounds of Theorem 1 in solving three test problems from the ICAPS Planning Competitions for which action costs are positive, but not uniform. The Tireworld problem (instance p07 from the 2004 competition) has one action with a cost of 100, while the other actions have unit cost. The Boxworld problem (instance c4-b3 from the 2004 competition) has actions costs of 1, 5, and 25. The Zenotravel problem (instance p01-08 from the 2008 competition) has action costs of 1, 10, and 25.

The graphs in Figure 1 show the lower bound for the start state $s0$, which is computed by FVI, and the two upper bounds. The new general bounds are better than the positive-cost bounds (by a factor that is approximately equal to the ratio $N_k(s_0)/J_k(s_0)$). However, the new bounds do not converge quite as smoothly as the positive-cost bounds, as seen in Figure 1(c), because they are affected by fluctuations in N_k, $\overline{n}_k$, and $\overline{c}_k$, and not just $\overline{c}_k$. For these three problems, the overhead for computing the steps-to-go function for the new bounds is less than 1% of the overall running time of FVI.

The results in Figure 1 show that the positive-cost bounds of Theorem 1 still perform very well for these test problems, even though action costs are not uniform. In fact, it takes only a few more iterations for the positive-cost bounds to reach the same point as the new bounds. The apparent explanation is that the two upper bounds differ by a constant factor, while they converge at a geometric rate. Even if the constant-factor difference is large, a geometric convergence rate tends to ensure that the difference in the number of iterations required to reach the same point is not that much.

Of course, the most important advantage of the new

Problem	Tireworld	Boxworld	Zenotravel
Number of states	475,078	1, 024,000	313,920
Explored states (by FVI)	273	138,364	139,874
States in final policy	46	34	9
FVI runtime	0.05	471.02	114.21
LAO* runtime	0.05	518.67	118.60
LRTDP runtime	0.09	460.58	1,330.42

Table 1: Problem characteristics and algorithm running times (in CPU seconds) to solve problems to ϵ-consistency with $\epsilon = 10^{-6}$.

Problem	Characteristics		Runtime in CPU seconds							
	$\|S\|$	|policy|	VI	LRTDP	HDP	LDFS	LDFS+	LAO*-B	LAO*	FVI
big	22,534	4,321	1.31	1.44	0.69	0.51	0.21	1.03	0.26	0.30
bigger	51,943	9,037	4.33	3.13	2.40	1.93	0.67	3.20	1.16	0.63
square-3	42,085	790	1.76	0.06	0.03	0.02	0.04	0.09	0.06	0.07
square-4	383,970	1,000	46.57	0.08	0.05	0.04	1.76	0.28	1.09	1.37
ring-5	94,396	12,374	5.47	4.37	2.22	1.90	0.70	8.77	1.38	1.53
ring-6	352,135	37,437	35.64	48.86	16.75	16.16	4.39	68.12	6.01	6.35
wet-160	25,600	1,364	1.85	7.13	70.29	50.72	4.60	0.22	0.06	0.06
wet-200	40,000	749	2.15	3.61	24.62	17.18	1.93	0.10	0.03	0.03
nav-18	262,143	2,494	90.32	55.83	2421.97	3034.58	2.07	3.43	1.67	1.68
nav-20	1,048,575	1,861	407.65	85.28	1946.06	1892.55	3.24	2.83	2.06	2.26

Table 2: Algorithm running times in CPU seconds until ϵ-consistency with $\epsilon = 10^{-8}$. Test problems from Bonet and Geffner (2006).

bounds of Theorem 2 is that they do not require all action costs to be positive, which means they apply to a broad range of SSP problems for which the positive-cost bounds of Theorem 1 cannot be used. The effectiveness of the new bounds in solving SSP problems with non-uniform positive costs suggests that they will also be effective in solving SSP problems for which not all action costs are positive.

There is another important conclusion to draw from the results shown in Figure 1. For these three test problems, there is a striking difference in the number of iterations it takes until convergence. Table 1 gives additional information about the problems that helps explain some differences. For example, convergence is fastest for the Tireworld problem because the number of states FVI evaluates is very small for this problem compared to the other two. But such differences are not easy to predict, and that highlights the value of the bounds. Without them, it can be *very* difficult to estimate how long a heuristic search algorithm should run until it has found a greedy policy that is ϵ-optimal, or even proper.

Comparison of algorithms

In discussing how to integrate the error bounds in heuristic search algorithms for SSP problems, we introduced a very simple heuristic search algorithm, called Focused Value Iteration (FVI). We conclude by considering how its performance compares to other algorithms.

FVI is most closely related to LAO*. In fact, the only difference between the two algorithms is that LAO* gradually expands an *open* policy over a succession of iterations until it is *closed*, whereas FVI evaluates the best closed policy each iteration. (A policy is said to be closed if it specifies an action for every state that is reachable from the start state under the policy; otherwise, it is said to be open.)

Bonet and Geffner describe several closely-related algorithms, including Labeled RTDP (LRTDP) (Bonet and Geffner 2003b), Heuristic Dynamic Programming (HDP) (Bonet and Geffner 2003a), and Learning Depth-First Search (LDFS) (Bonet and Geffner 2006). These algorithms differ from FVI in two ways. First, they use a technique for labeling states as solved that can accelerate convergence. Second, they adopt a *find-and-revise* approach that terminates a depth-first traversal of the reachable states as soon as the residual for a state exceeds a threshold value.

Adoption of the find-and-revise approach means that these algorithms never complete a depth-first traversal until their last iteration, and so they do not compute a residual until they terminate. It follows that they cannot use the error bounds to monitor the progress of the search and dynamically decide when to terminate. In this important respect, the error bounds are a much better fit for FVI and LAO*.

To compare the running times of these search algorithms, we repeated an experimental comparison reported by Bonet and Geffner (2006), using their publicly-available implementation and test set. Table 2 shows the results of the comparison, including the performance of value iteration (VI) and a modified version of LDFS called LDFS+. The results, averaged over ten runs, are consistent with the results reported by Bonet and Geffner (2006), although we draw attention to a couple differences. The column labeled "LAO*-B" shows the performance of their implementation of LAO*. We added a column labeled "LAO*" that shows the performance of the LAO* algorithm described by Hansen and Zilberstein (2001). The difference is related to the fact that all of the algorithms of Bonet and Geffner use an extra stack to manage a procedure for labeling states as solved, which incurs considerable overhead. They implement LAO* in the same way, using an extra stack, although LAO* does not label states as solved. When this unnecessary overhead is removed from the implementation of LAO*, its performance improves significantly. The difference is especially noticeable as the size of the policy increases.

The results in Table 2, as well as additional results in Table 1 that compare the performance of FVI, LAO*, and LRTDP in solving the ICAPS planning problems with non-uniform action costs, show that FVI performs as well or better than the other algorithms. Overall, FVI and LAO* perform best, and their performance is very similar. In experiments we do not show for space reasons, LAO* has one advantage compared to FVI: it explores fewer different states than FVI – in our experiments, about 5% to 10% fewer. That is, the strategy of gradually expanding an open path until it is closed, which LAO* inherits from A* and AO*, has the benefit of reducing the number of "expanded" states. For A*, which expands and evaluates each state only once, it is an important advantage. In solving SSP problems, where a state is "expanded" once, and then evaluated thousands of times before convergence, it has little effect on running time.

Conclusion

We have shown how to generalize recently-derived error bounds for stochastic shortest path problems so that they do not depend on the cost structure of the problem. The error bounds can be used not only by value iteration, but by heuristic search algorithms that compute a residual, such as LAO* and related algorithms. Although we tested the bounds on problems with non-uniform positive costs, the approach has greater significance for problems where action costs are not all positive, and previous results do not apply.

In the course of generalizing the bounds, we also introduced a simpler version of LAO*, called Focused Value Iteration. Somewhat surprisingly, it performs as well as LAO*, and as well or better than several other algorithms, at least on some widely-used benchmarks. This result suggests that most of the benefit from the heuristic search approach comes from the very simple strategy of only evaluating states that could be visited by a greedy policy, and identifying these states with as little overhead as possible. Other search strategies described in the literature may very well improve performance further. More study will help to clarify when they are effective and how much additional benefit they provide.

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

Appendix

Bertsekas (2005, p. 413) proves that the following upper bound holds under Jacobi dynamic programming updates,

$$J^*(i) \leq J_{\mu^k}(i) \leq J_k(i) + (N_{\mu^k}(i) - 1) \cdot \overline{c}_k. \tag{20}$$

It is the same upper bound introduced in Equation (5) and used to derive our bounds in this paper.

In this appendix, we establish the extent to which the bounds of (20) also hold under Gauss-Seidel dynamic programming updates. Our result turns on the following lemma.

Lemma 1. *Given the residual $\overline{c}_k$ for a Gauss-Seidel dynamic programming update, the residual $\overline{c}_{k+1}$ for a subsequent Jacobi dynamic programming update is bounded as follows: $\overline{c}_{k+1} \leq \max\{0, \overline{c}_k\}$.*

Proof. Assume the states in S are indexed from 1 to n, where $n = |S|$, and let

$$Q_{k+1}(i,a) = g_i^a + \sum_{j=1}^{n} P_{ij}^a J_k(j)$$

denote the result of a Jacobi update of state i for action a taken at stage k, and let

$$Q_k(i,a) = g_i^a + \sum_{j=1}^{i-1} P_{ij}^a J_k(j) + \sum_{j=i}^{n} P_{ij}^a J_{k-1}(j)$$

denote the result of a Gauss-Seidel update of state i for action a taken at stage $k-1$. Note that

$$Q_{k+1}(i,a) - Q_k(i,a) = 0 + \sum_{j=i}^{n} P_{ij}^a \left(J_k(j) - J_{k-1}(j)\right), \tag{21}$$

where we include 0 in (21) in case $\sum_{j=i}^{n} P_{ij}^a = 0$.

If the same action a is taken at both stages, then

$$\begin{aligned} J_{k+1}(i) - J_k(i) &= Q_{k+1}(i,a) - Q_k(i,a) \\ &\leq \max_{a \in A(i)} \{Q_{k+1}(i,a) - Q_k(i,a)\}. \end{aligned}$$

If a different action a' is taken at stage k than the action a taken at stage $k-1$, then $Q_{k+1}(i,a') \leq Q_{k+1}(i,a)$, and so

$$\begin{aligned} J_{k+1}(i) - J_k(i) &= Q_{k+1}(i,a') - Q_k(i,a) \\ &\leq Q_{k+1}(i,a) - Q_k(i,a) \\ &\leq \max_{a \in A(i)} \{Q_{k+1}(i,a) - Q_k(i,a)\}. \end{aligned}$$

In both cases,

$$J_{k+1}(i) - J_k(i) \leq \max_{a \in A(i)} \{Q_{k+1}(i,a) - Q_k(i,a)\}. \tag{22}$$

From (21) and (22), we have

$$J_{k+1}(i) - J_k(i) \leq \max_{a \in A(i)} \left\{ 0 + \sum_{j=i}^{n} P_{ij}^a \left(J_k(j) - J_{k-1}(j)\right) \right\}.$$

It follows that

$$\begin{aligned} \overline{c}_{k+1} &= \max_{i=1,\ldots,n} \{J_{k+1}(i) - J_k(i)\} \\ &\leq \max_{i=1,\ldots,n} \max_{a \in A(i)} \left\{ 0 + \sum_{j=i}^{n} P_{ij}^a \left(J_{k+1}(j) - J_k(j)\right) \right\} \\ &\leq \max\{0, \overline{c}_k\}. \end{aligned}$$

□

From Lemma 1, if the residual $\overline{c}_k$ after a Gauss-Seidel update is non-negative, it gives an upper bound on the residual $\overline{c}_{k+1}$ of a subsequent Jacobi dynamic programming update. It follows that the error bound of (20) can be used after a Gauss-Seidel update if $\overline{c}_k$ is non-negative. If $\overline{c}_k$ is negative, the rightmost upper bound of (20) does not hold under Gauss-Seidel updates. But in that case, J_k itself is a monotone upper bound. Thus we have the following theorem.

Theorem 3. *For a cost-to-go function J_k that is the result of a Gauss-Seidel dynamic programming update of J_{k-1}, we have the following upper bounds for states $i = 1, \ldots, n$.*

(a) If $0 < \overline{c}_k$, then

$$J^*(i) \leq J_{\mu^k}(i) \leq J_k(i) + \left(N_{\mu^k}(i) - 1\right) \cdot \overline{c}_k. \tag{23}$$

(b) If $\overline{c}_k \leq 0$, then

$$J^*(i) \leq J_{\mu^k}(i) \leq J_k(i). \tag{24}$$

When the cost vector J_k is a lower-bound function, the residual $\overline{c}_k$ must be non-negative and case (b) of Theorem 3 is not needed. But for the general-cost bounds of Theorem 2, the steps-to-go function N_k is not necessarily a lower-bound function, and so both cases of Theorem 3 are needed.

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.

Hansen, E., and Abdoulahi, I. 2015. Efficient bounds in heuristic search algorithms for stochastic shortest path problems. In *Proceedings of the 29th AAAI Conference on Artificial Intelligence (AAAI-15)*, 3283–3290. Austin, Texas: AAAI Press.

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

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.