# Spurious Solutions to the Bellman Equation

**Mance E. Harmon**
Wright Laboratory
WL/AACF
2241 Avionics Circle
WPAFB, OH 45433-7318
harmonme@aa.wpafb.mil

**Leemon C. Baird III**
U.S.A.F. Academy
2354 Fairchild Dr. Suite 6K41
USAFA, CO 80840-6234
baird@cs.usafa.af.mil

**Abstract**

Reinforcement learning algorithms often work by finding functions that satisfy the Bellman equation. This yields an optimal solution for prediction with Markov chains and for controlling a Markov decision process (MDP) with a finite number of states and actions. This approach is also frequently applied to Markov chains and MDPs with infinite states. We show that, in this case, the Bellman equation may have multiple solutions, many of which lead to erroneous predictions and policies (Baird, 1996). Algorithms and conditions are presented that guarantee a single, optimal solution to the Bellman equation.

## 1 REINFORCEMENT LEARNING AND DYNAMIC PROGRAMMING

### 1.1 THE BELLMAN EQUATION

Reinforcement learning algorithms often work by using some form of dynamic programming to find functions that satisfy the Bellman equation. For example, in a pure prediction problem, the true, optimal value of a state, $V^*(x_t)$, is defined as equation (1), where < > represents the expected value, taken over all possible sequences of states after time $t$, $\gamma$ is a discount factor between zero and one exclusive, and $R$ is the reinforcement received on each time step.

$$V^*(x_t) = \left\langle R_t + \gamma R_{t+1} + \gamma^2 R_{t+2} + \gamma^3 R_{t+3} + \mathrm{K} \right\rangle \tag{1}$$

It is clear from equation (1) that there is a simple relationship between successive states. This relationship is given in equation (2), and is referred to as the Bellman equation for this problem.

$$V^*(x_t) = \left\langle R_t + \gamma V^*(x_{t+1}) \right\rangle \tag{2}$$

Bellman equations can be derived similarly for other algorithms such as *Q*-learning (Watkins, 1989) or advantage learning (Baird, 1993, Harmon and Baird, 1996).

### 1.2 UNIQUE SOLUTIONS

A learning system will maintain an approximation $V$ to the true answer $V^*$, and the difference between the two can be called the error $e$, defined in equation (3). Equation (4) shows why dynamic programming works. If the learning system can find a function $V$ that satisfies equation (2) for all states, then equation (4) will also hold for all states.

$$V(x_t) = V^*(x_t) + e(x_t) \tag{3}$$

$$\begin{aligned} V^*(x_t) + e(x_t) &= \left\langle R_t + \gamma\left(V^*(x_{t+1}) + e(x_{t+1})\right)\right\rangle \\ V^*(x_t) + e(x_t) &= \left\langle R_t + \gamma V^*(x_{t+1})\right\rangle + \gamma\left\langle e(x_{t+1})\right\rangle \\ e(x_t) &= \gamma\left\langle e(x_{t+1})\right\rangle \end{aligned} \tag{4}$$

Suppose there are a finite number of states, and call the state with the largest error state $x_t$. The discount factor $\gamma$ is a positive number less than 1, so equation (4) says that the largest error is equal to only a fraction of a weighted average of all the errors. The only way this could happen is if all the errors were zero. Thus, for a finite number of states, the Bellman equation has a unique solution, and that solution is optimal.

On the basis of this result, reinforcement learning systems have been created that simply try to find a function $V$ that satisfies the Bellman equation (e.g., Tesauro, 1994; Crites and Barto, 1995). But will such a $V$ will be optimal, even when there are an infinite number of states? Can we assume that the finite-state results will also apply to the infinite-state case?

## 2 SPURIOUS SOLUTIONS

It would be useful to determine under what conditions dynamic programming is guaranteed to find not only a value function that satisfies equation (4), but also a value function whose *value error*, defined in equation (5), is zero for all $x$.

$$e(x) = V^*(x) - V(x) \tag{5}$$

One solution to equation (4) is the optimal value function $V^*$. However, in many cases there might exist more than a single, unique solution to the Bellman equation (Baird, 1996). If there is a finite number of states then there does exist a unique solution to equation (4). If there is an infinite number of states then there may exist an infinite number of solutions to the Bellman equation, including some with a suboptimal value function or policy.

### 2.1 THE INFINITE-HALL PROBLEM

Consider the simple case of a Markov chain with countably-infinite states, named 0, 1, 2, ..., and with a reinforcement of zero on every transition (Figure 1).

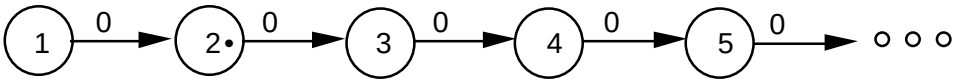



**Figure 1: Infinite Markov chain**

On each time step, the state number is increased by one. The Bellman equation, error relationship, and general solution for this Markov chain are given in equations (6), (7), and (8) respectively.

$$V(x_t) = \gamma V(x_{t+1}) \tag{6}$$

$$e(x_t) = \gamma e(x_t) \tag{7}$$

$$V(x_t) = k\gamma^{-t} \tag{8}$$

When $k$=0 this is the optimal answer, when $k$ is positive the value function will be much too high, and when $k$ is negative it will be much too low. But in every case, the function satisfies the Bellman equation. The problem is that a solution to the Bellman equation only ensures that each error will be less than the successive error. If the trajectory of states never crosses itself, then there is no reason that the error cannot grow without bound.

For most MDPs, the reinforcement $R$ is bounded between $R_{\max}$ and $R_{\min}$, and so the optimal value function $V^*$ is also bounded by $R_{\max}(1-\gamma)^{-1}$ and $R_{\min}(1-\gamma)^{-1}$. If the error is growing exponentially, and the optimal value function is bounded, that implies that $V$ will be growing exponentially. This suggests that spurious solutions to the Bellman equation are theoretically possible, but not a problem in practice, because all the suboptimal solutions will have functions that grow exponentially, and the typical function approximation system will not be able to represent them. Unfortunately, that is not true, as can be seen by a simple example.

Consider a regulator problem where the state changes at a rate proportional to the current state and control action. The problem is to find the value of the states for a given policy that causes the state to double on each step. The reinforcement $R$=0 and the discount factor $\gamma$=0.5 (Figure 2).

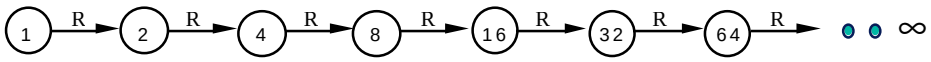



**Figure 2**

The Bellman equation and general solution to this problem are given in equations (9) and (10). The value function is correct at $x$=0, because this state loops back to itself. The value function can be grossly incorrect for all other values of $x$. In fact, if a linear function approximator is used with a single weight, then every function it can represent will also be a solution to the Bellman equation. As this illustrates, it is possible for a function-approximation system to converge to a spurious solution, or to have the initial random weights constitute a spurious solution.

$$V(x_t) = 0.5V(x_{t+1}) = 0.5V(2x_t) \tag{9}$$

$$V(x_t) = kx_t \tag{10}$$

## 2.2 WHAT CONDITIONS CAUSE SPURIOUS SOLUTIONS?

It would be useful to determine under what conditions dynamic programming is guaranteed to find not only a value function that satisfies the Bellman equation, but also a value function whose value error, defined as the difference between the optimal value function and the value function estimate, is zero for all $x$. To do this, we first determined under what conditions spurious solutions might exist.

For each of the following examples it is assumed that the reinforcement function $R$ and the optimal value function $V^*$ are bounded, the number of states is infinite, and the same state is never visited more than once by the policy. If the speed is constant, as depicted in Figure (3), there can exist an infinite number of solutions if the value function grows exponentially. Also, in the case that the speed is not fixed but decays exponentially, there can exist an infinite number of solutions to the Bellman equation if the value function goes to infinity at a point (Figure 4). When the value function does not grow exponentially, an infinite number of solutions might still exist if the speed at which states are visited is increasing exponentially (Figure 5).

Finally, Figure (6) depicts the case where the state is bounded, the value function is unbounded, and the slope of the value function is unbounded. Given these circumstances, there exists an infinite number of solutions to the Bellman equation. Each of these examples have one thing in common, no state is revisited a second time. If these two conditions hold, spurious solutions to the Bellman equation can exist. However, it is possible to remedy the situation by adding some minor constraints.

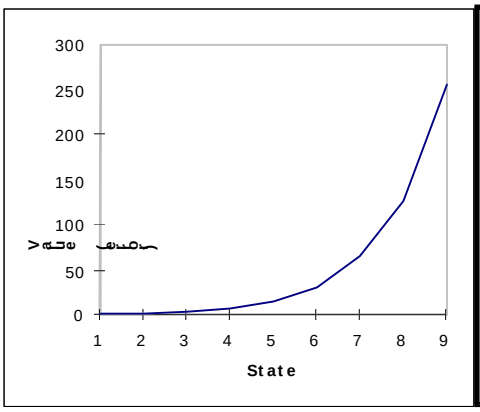



**Figure 3: Constant speed**

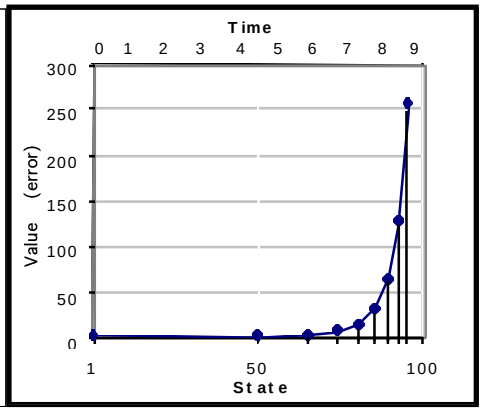
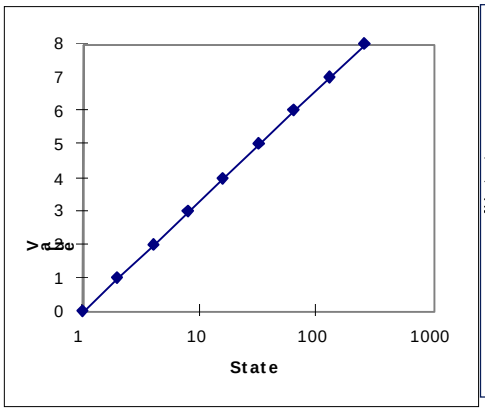



**Figure 4: Exponentially decaying speed**



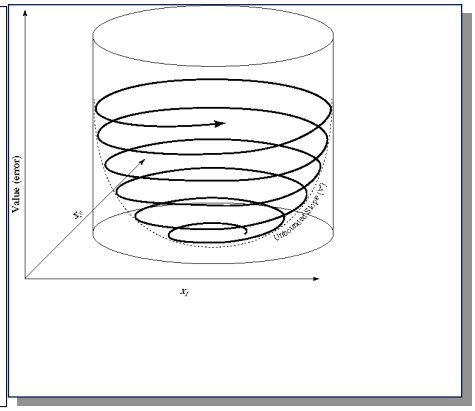



**Figure 5: Exponentially increasing speed**

**Figure 6: Bounded states, unbounded value function, unbounded slope**

## 3 ENSURING UNIQUE SOLUTIONS

In Section 2 we considered the causes of spurious solutions in prediction problems. We now discuss how to prevent spurious solutions not only in problems of prediction but also control, and we restrict our discussion to MDPs with finite action sets. The same conditions that cause spurious solutions in Markov chains also cause spurious solutions in Markov decision processes.

It may be observed that all sub-optimal value functions that satisfy the Bellman equation grow without bound. Only the value function that satisfies the Bellman equation and has a zero value error for all $x$ is guaranteed to be bounded. In fact, all spurious solutions to the Bellman equation require the value function to grow without bound, if the reinforcement is bounded and the equation must be satisfied over all of state space. However, if the reinforcement function is bounded and the discount factor $\gamma$ is less than 1, then there does exist a unique solution to the Bellman equation in which the value function does not grow without bound and the error function is zero over all of state space.

This suggests that by bounding the value function and training over all of state space a unique solution to the Bellman equation is ensured. One can accomplish this practically by passing the output of the function approximation system representing the value function through a sigmoid. The lower and upper bounds of the sigmoid are given in equations (11) and (12). This method is useful not only for Markov chains but also for both deterministic and non-deterministic Markov decision processes.

$$V_{\max} = \frac{R_{\max}}{1-\gamma} \tag{11}$$

$$V_{\min} = \frac{R_{\min}}{1-\gamma} \tag{12}$$

### 3.1 UNBOUNDED STATE SPACES

To guarantee a unique solution to the Bellman equation for both Markov chains and MDPs with finite action sets, one can bound the value function and train over all of state space. The question then becomes one of how to train over all of an unbounded state space. Training over a bounded region of state space when state space is unbounded might result in sub-optimal solutions.

One solution is to map all of state space to a bounded region, then train uniformly over this region. For example, if the state space is the real number line, then one might use the mapping $f(x) = (1+e^{-x})^{-1}$. When training in state $x$, the input to the neural network would be $f(x)$. Random states would be generated during training by choosing random numbers $r$ between 0 and 1, then letting $x=f^{-1}(r)$. In addition, for radial basis function networks, there is an additional advantage to using $f(x)$ as the network input rather than using $x$ as the input. When $x$ is the input, the RBF network will always learn to have zero outputs for sufficiently-large values of $x$. When the input is $f(x)$, the network has the flexibility to have either zero or non-zero outputs as $x$ goes to infinity.

When training on trajectories in an unbounded state space, it is possible that no terminating state exists. In this case, to perform a single update, one is required to follow all possible trajectories from an initial state, each of which may visit an infinite

number of states. This must be done for an infinite number of initial states. For this reason, training can only be accomplished in practice by training on randomly generated state samples.

### 3.2 TRAINING ON A SUBSET

In many real-world problems (e.g., regulator problems), we can only train on a subset of the state space, but we know that the optimal policy will stay within that subset. In this case (when the optimal policy is bounded), one has the option of training only on this bounded region of state space. Bounded optimal policy is defined as follows: if following the optimal policy $\pi^*$ from a given set of initial states, a bounded region of state space exists whose boundary is never crossed. This region is defined as the training region of state space. Again, it is assumed that the reinforcement function is bounded, that $\gamma$ is a positive number less than 1, and the value function is bounded (for reasons stated earlier).

If the optimal policy $\pi^*$ is bounded, the system is deterministic, and a lookup table is being used to store the function (or any function approximator that does not generalize), the following method will guarantee a unique solution to the Bellman equation. Consider the case for which the learning algorithm learns a function of state-action pairs. During training, if the exploration policy causes a transition across the boundary of the training region from state $x_t$ to $x_{t+1}$, one could train the *Q*-value for the chosen state-action pair to be $\varepsilon$ smaller than the maximum *Q* value in state $x_t$ (when trying to maximize total discounted reinforcement). This will always train the *Q*-value for the state-action pair that transitions to a state across the boundary to have a value $\varepsilon$ less than the maximum *Q*-value in the current state (assuming the exploration policy chooses state-action pairs with uniform random probability across all possible state-action pairs in a given state). This, in turn, necessarily means that a state-action pair that causes a transition across the training region boundary will not have the maximum *Q*-value for the given state. The same principle can be applied to systems that learn a value function rather than a function of state-action pairs.

### 3.3 NONDETERMINISTIC MDPs

When the system is non-deterministic, or when using a function approximator that generalizes to represent the value function, it becomes necessary to artificially value states outside the training region such that anytime a chosen action causes a transition across the boundary of the training region, the successor state has a large negative value (when maximizing total discounted reinforcement) or large positive value (when minimizing total discounted reinforcement). The magnitude of the values outside the training region must also grow over time, forcing the system to learn not to choose an action that has a non-zero probability of crossing the training boundary. This forces the system to learn a policy that stays within the training region, provides a general solution, and is independent of the reinforcement learning algorithm being used. For reinforcement learning algorithms that learn a function of state-action pairs there is an alternative. The desired effect can still be accomplished if updates to the value function (Q-function, advantage function, etc.) approximation occur only when following the current action policy causes a transition across the training region boundary. If following the exploration policy during training causes a transition across the training boundary, no learning should be accomplished for that training transition and a new trial should be started.

Two problems are associated with these methods of training. First, one must have *a priori* knowledge of the optimal policy so that a reasonable training region can be defined. Second, if the values of the states outside of the training region are defined to

be too high or too low then the value function being learned has steep slopes (spikes) at the boundary edges. These, in turn, may slow learning for function approximation systems that generalize. The effects of the "spikes" can be mitigated by defining the training region boundaries to be relatively distant from states visited when following the optimal policy from a set of initial states, and thus ensure a smooth value function across the region of state space visited when following the optimal policy.

## 4 EMPIRICAL RESULTS

To demonstrate that spurious solutions can be a problem in practice, we consider a Markov decision process in which there are countably infinite states. In each state are two possible actions: return to the current state or transition to the successor state. The MDP has states labeled 0.1, 0.2, 0.3, ... , and reinforcements as shown in Figure (7).

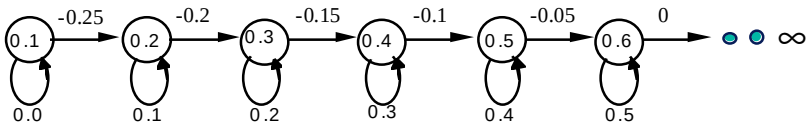



**Figure 7: Values on arcs are reinforcements received for the given action.**

For this example we used residual gradient value iteration (Baird, 1995). The value function is represented with a single sigmoidal node. The output of the node is bounded from [-1,1]. The input to the node is the weighted sum of the state and a bias. The value of a state is the weighted output of the hidden node. The update equation is:

$$w = w + \alpha \left[ \max_{x_{t+1}} \left( R(x_t, x_{t+1}) + V(x_{t+1}) \right) - V(x_t) \right] \left[ \gamma \frac{\partial V(x_{t+1})}{\partial w} - \frac{\partial V(x_t)}{\partial w} \right] \qquad (13)$$

In equation (13) $w$ is the weight vector containing the parameters of the node, $\alpha$ is the learning rate and has a fixed value of 0.09, $\gamma$ is the discount factor and has a value of 0.9, and $R(x_t,x_{t+1})$ represents the reinforcement received when transitioning from state $x_t$ to state $x_{t+1}$. The weights were initialized to lie between $[-1x10^{-8},1x10^{-8}]$. We trained for one million iterations over the first five states of the MDP. In each iteration, a state was randomly chosen for training from a uniform probability distribution over the set of training states.

The system learned a value function that reduced the Bellman residual but did not reduce the value error. Furthermore, the policy resulting from the learned function was incorrect for all states in the training set. Figure (8) shows the optimal value function $V^*$ and the learned value function $V$.

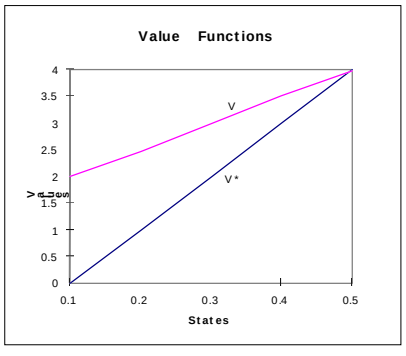



**Figure 8: Learned value function *V* and optimal value function *V****

## 5 SUMMARY

Ordinary function approximators, when used to solve simple prediction and control problems, can converge to spurious solutions of the Bellman equation, yielding grossly incorrect predictions and policies. Algorithms and conditions have been presented that guarantee a single, optimal solution to the Bellman equation. It may be important to take these into account when using reinforcement learning for prediction or control.

**Acknowledgments** This research was supported under Task 2312R1 by the United States Air Force Office of Scientific Research.

**References**

Baird, L. C. (1996). *Spurious Bellman-equation solutions.* (Internal Tech Report) Department of Computer Science, USAF Academy.

Baird, L. C. (1993). *Advantage updating* (DTIC Report AD WL-TR-93-1146, available from the Defense Technical Information Center, Cameron Station, Alexandria, VA 22304-6145). Wright-Patterson Air Force Base, OH.

Baird, L. C. (1995). *Residual Algorithms: Reinforcement Learning with Function Approximation*. In Armand Prieditis & Stuart Russell, eds. Machine Learning: Proceedings of the Twelfth International Conference, 9-12 July, Morgan Kaufman Publishers, San Francisco, CA.

Crites, R. H., and Barto, A. G. (1996). Improving elevator performance using reinforcement learning. To appear in *Advances in Neural Information Processing Systems 8*, D. S. Touretzky, M. C. Mozer, M. E. Hasselmo, eds., MIT Press.

Harmon, M. E., and Baird, L. C. (1996). *Multi-agent residual advantage learning with general function approximation*. (Internal Tech Report). Wright Laboratory, Wright-Patterson Air Force Base, OH.

Tesauro, G. (1994). TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation* **6**:215-219.

Watkins, C. J. C. H. (1989). *Learning from delayed rewards*. Doctoral thesis, Cambridge University, Cambridge, England.