Reinforcement Learning with Stability Guarantees

Theodore J. Perkins
Sascha E. Englebrecht
Andrew G. Barto

Department of Computer Science
University of Massachusetts
Amherst, MA 01003

Technical Report UM-CS-2000-06

Reinforcement Learning with Stability Guarantees

Theodore J. Perkins PERKINS@CS.UMASS.EDU
Sascha E. Engelbrecht SASCHA@CS.UMASS.EDU
Andrew G. Barto BARTO@CS.UMASS.EDU
Department of Computer Science, University of Massachusetts, Amherst, MA 01003, USA

Abstract

We present a new approach to improving the reliability and overall performance of reinforcement learning (RL) control for minimum cost-to-target problems. In particular, we propose combining RL with Lyapunov-based control, an analytical approach from control theory for designing controllers that make dynamical systems stable. By building Lyapunov-based constraints into a control architecture, we can prove that the controlled system will be brought to a desired target state. At the same time, RL is used to minimize the cost incurred by the controller. The resulting controllers combine theoretical guarantees on their behavior with improved practical performance compared to either standard RL or Lyapunov-based methods alone. Because the controllers ensure stability by design, the guarantees we establish hold during learning as well as after. In fact, the guarantees hold independently of many characteristics of the RL method being used, including the method of function approximation. We illustrate our techniques on a pendulum swing-up task.

1. Introduction

Since at least the 1950's, control theorists have been developing methods for provably stable control of dynamical systems based on the analytic techniques of A. M. Lyapunov – techniques for establishing the stability properties of dynamical systems. Controllers designed using these methods are guaranteed to bring a dynamical system to some desired target state. However, there is usually some cost to operating a controller, such as expended energy or materials, wear on controller components, etc. Lyapunov-based design approaches do not generally address costs incurred by the controller; they only guarantee that the system will be brought to the target (Vincent & Grantham, 1997).

Reinforcement learning (RL), on the other hand, comprises a broad set of numerical techniques that explicitly attempt to minimize the costs incurred while controlling a system. When a dynamical system can only be in a small number of discrete states, various algorithms can exactly minimize cost (Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 1998). However this is not generally possible when the number of possible states is large or the system state is described by continuous variables. In these cases, many RL algorithms are combined with function approximation, but the theoretical guarantees on RL systems using function approximation are much weaker than in the case of no function approximation. In fact, it has been shown that some of the more popular RL algorithms, such as Q-learning and TD(λ), can diverge when combined with function approximation (Baird, 1995; Bertsekas & Tsitsiklis, 1996).

In this paper we present a novel approach to controller design that combines Lyapunov-based control ideas with RL. Controllers designed by our approach provably take the controlled dynamical system to a target state, while simultaneously allowing considerable freedom for the minimization of the cost incurred.

In our approach, getting the system state to target, i.e. stability, is ensured by imposing Lyapunov-based constraints on the available control choices. Because stability is ensured by control choice constraints, many aspects of the RL method being used do not affect stability, including: the method of function approximation, if any; exploration; and even whether or not the RL algorithm converges to a single outcome.

We illustrate our ideas on a pendulum swing-up task – a standard nonlinear control problem. We demonstrate the Lyapunov analysis required for producing control constraints. Simulation experiments reveal some of the empirical advantages of our techniques, such as improved initial performance for on-line RL, and improved asymptotic performance compared to Lyapunov-based control alone.

2. Definitions

We consider control problems in which a dynamical system's state is described by a vector $\mathbf{x} \in \Re^n$. In each state $\mathbf{x}$, there is an admissible set of controls $\mathcal{U}(\mathbf{x}) \subseteq \Re^m$. The

system evolves in time according to a control differential equation,

$$\dot{\mathbf{x}} = \mathrm{f}(\mathbf{x}, \mathbf{u}),$$

where $\mathbf{u} \in \mathcal{U}(\mathbf{x})$ is the control command.

Any trajectory, $(\mathbf{x}(\mathrm{t}), \mathbf{u}(\mathrm{t}))$, $t \geq 0$, has an associated cost

$$J[\mathbf{x}(\cdot), \mathbf{u}(\cdot)] = \int_{\mathrm{t}} \mathrm{g}(\mathbf{x}(\mathrm{t}), \mathbf{u}(\mathrm{t}))\mathrm{dt},$$

where g is an instantaneous-cost function, mapping states and control signals to nonnegative real numbers.

In particular, our work applies to minimum cost-to-target problems, in which there is a target region $\mathcal{T} \subset \Re^n$. If a trajectory enters the target region, the cost J of that trajectory is computed only until the time of entry. Effectively, we are not interested in what happens after the target has been reached. We take the view that the controller has achieved the goal of the control problem.

The objective in minimum cost-to-target problems is to identify a way of controlling the system that takes the system's state from any starting point $\mathbf{x} \in \Re^{\mathrm{n}}$ to the target region while incurring minimum cost.

A control law is a mapping from system states to admissible controls, $w : \Re^n \mapsto \Re^m, w(\mathbf{x}) \in \mathcal{U}(\mathbf{x})$. To say that the system is controlled according to a control law w just means the system evolves as: $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x}, \mathbf{u}) = \mathbf{f}(\mathbf{x}, \mathrm{w}(\mathbf{x}))$.

The above formulation describes continuous-time control problems, but the controllers we develop have a discrete time aspect. For each unit of time, also called a "time step", these controllers select a particular control law w, and the system is controlled according to that control law during the time step. We will use the term "control action" to refer to a control law that controls the system for one time step. Thus, at the beginning of each time step, our controllers choose among a set of candidate control actions. Note that the controllers we design need not themselves be simple control laws. They may change with time or be stochastic, basing their control action choices on extraneous random variables.

There are many different types of stability for dynamical systems. For example, trajectories may be required to: approach some target point, stay close to a target point, or actually enter a target region. When we say a (controlled) dynamical system is stable we will generally mean that, with probability one, the system state will enter a specified target region at some finite, but not necessarily bounded, time. One exception is section 3, which gives background on Lyapunov-based control without reference to any particular definition of stability. All other exceptions are clearly noted.

3. Lyapunov-Based Control

The original theorems of A. M. Lyapunov were aimed at establishing the stability of dynamical systems (Vincent & Grantham, 1997). Proof of stability is made by identifying what has become known as a Lyapunov function for the system. Lyapunov functions are often thought of as generalized "energy" functions. They are real-valued functions of the state of the system. Stability is established by showing that this generalized energy is "dissipated," i.e. decreases continuously along trajectories, until the system settles into an unchanging state of (locally) minimum energy.

In control engineering, Lyapunov's theorems and others in that style are used to design controllers that are provably stable – i.e. controllers that are guaranteed to bring the system state to target. One basic approach is to (1) invent a control law, and (2) show that the resulting controlled system is stable by providing a Lyapunov function (Vincent & Grantham, 1997). If the Lyapunov function's state of minimum energy is in the target region, then the controller is guaranteed to bring the system to target. Such controllers are said to "stabilize the system" and are called "stabilizing," or just "stable."

There is no general procedure for accomplishing either of the above steps for arbitrary, nonlinear dynamical control problems. Further, a Lyapunov analysis usually requires considerable knowledge of the dynamical system – sometimes complete differential equations describing the system. Thus, establishing stability by Lyapunov methods can be difficult.

However, these problems are not insurmountable. There are general procedures for analyzing linear systems and certain classes of nonlinear systems (Vincent & Grantham, 1997). For many problems of practical interest, Lyapunov functions are already known, and there are many control engineers with expertise in constructing Lyapunov functions. Lyapunov-based design requires domain knowledge and is an open-ended task in general, but yields the great benefit of guaranteeing that the system state be controlled to target.

It is important to notice that, in this basic methodology, the cost function g plays no explicit role at all. Thus, although the controlled system is stable, the controller may incur high costs while bring the system to target. The methods we propose in this paper begin with Lyapunov-based analysis of the dynamical system, but do not end there. The control architectures we propose are provably stable because of the way they translate a Lyapunov analysis into control constraints. But, these architectures still have considerable freedom to minimize for cost, which we do by using RL.

4. Reinforcement Learning

For purposes of this paper, all that the reader needs to know about RL algorithms in general is that their goal is to compute cost-minimizing control strategies for control problems. We will only describe Q-learning (Watkins, 1989) – a popular RL algorithm and the one we use in our simulation experiments.

Q-learning is a discrete-time control method, and is usually formulated to decide among a discrete, finite set of control actions. Q-learning works by incrementally computing $Q^*(\mathbf{x}, \mathrm{w})$, the optimal action-value function, for the control problem. $Q^*(\mathbf{x}, \mathrm{w})$ can be thought of as the cost of a trajectory beginning at state $\mathbf{x}$ with control according to w for the first time step and optimal control thereafter. Q^* satisfies the Bellman optimality equation (Bertsekas & Tsitsiklis, 1996), which can be written for deterministic systems as:

$$Q^*(\mathbf{x}, \mathrm{w}) = \mathrm{c}(\mathbf{x}, \mathrm{w}) + \min_{\mathrm{w_i}} \mathrm{Q}^*(\mathbf{x}', \mathrm{w_i}),$$

where $c(\mathbf{x}, \mathrm{w})$ is the cost incurred during the first time step and $\mathbf{x}'$ is the system state resulting from $\mathbf{x}$ and the control action w. An optimal control action w_{opt} for any state $\mathbf{x}$ is easily recovered from Q^*:

$$w_{\mathrm{opt}}(\mathbf{x}) = \arg\min_{\mathrm{w_i}} \mathrm{Q}^*(\mathbf{x}, \mathrm{w_i}).$$

Q-learning is an "on-line" control method. It maintains estimates $\hat{Q}(\mathbf{x}, \mathrm{w})$ of $Q^*(\mathbf{x}, \mathrm{w})$, and updates them based on observed outcomes of controlling the system. Suppose the system is in state $\mathbf{x}$ and the system is controlled for one time step according to control action w. Suppose cost c accumulates during that time step, and the system is in state $\mathbf{x}'$ afterward. Q-learning uses the Bellman optimality equation as an update rule for its estimate of $\hat{Q}(\mathbf{x}, \mathrm{w})$:

$$\hat{Q}(\mathbf{x}, \mathrm{w}) \leftarrow \alpha \hat{\mathrm{Q}}(\mathbf{x}, \mathrm{w}) + (1 - \alpha)(\mathrm{c} + \min_{\mathrm{w_i}} \hat{\mathrm{Q}}(\mathbf{x}', \mathrm{w_i})),$$

where α is a update rate.

When state spaces are very large or continuous, estimates cannot be maintained for every state $\mathbf{x}$. A standard approach in this case is to use function approximation to represent $\hat{Q}$ (Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 1998). The state $\mathbf{x}$ and control action w are represented by input features. The right hand side of the above update is used as a training target for the function approximator.

When combined with function approximation, many RL algorithms, Q-learning in particular, have been shown to diverge in some cases (Baird, 1995; Bertsekas & Tsitsiklis, 1996). In the minimum cost-to-target problems we study, there is no guarantee that these RL algorithms would ever bring the system to target. The control architectures we propose in this paper protect against such worst-case behavior, ensuring at least that the system reaches the target eventually. The hope, however, is that the RL component of our control architectures will succeed in minimizing the costs incurred in taking the system to target.

5. Stable Reinforcement Learning (StaRL)

We now introduce novel control design methods that combine the advantages of Lyapunov-based control and RL. In particular, we propose control systems that (1) are provably tabilizing, and (2) minimize cost of control as much as possible.

We will propose three different ways of constructing a stable RL controller, but first we state and prove a very general stability theorem that will cover all three methods. Roughly summarized, the theorem states that if a controller has a fixed chance per time step of moving the system state downhill on a Lyapunov function for the problem, then that controller will take the system to target eventually.

Theorem 1 *(StaRL Stability Theorem) Consider a dynamical system $\dot{\mathbf{x}} = \mathrm{f}(\mathbf{x}, \mathbf{u})$, for $\mathbf{x} \in \Re^{\mathrm{n}}$ and $\mathbf{u} \in \mathcal{U}(\mathbf{x}) \subseteq \Re^{\mathrm{m}}$, with target region $\mathcal{T} \subset \Re^n$. Suppose:*

1. *There is a region of state space $\mathcal{X} \subseteq \Re^n - \mathcal{T}$ with the property that any trajectory generated by admissible controls and starting in $\mathcal{X}$ can only stay in $\mathcal{X}$ or enter the target region.*
2. *L is a scalar function on $\Re^n$ (L plays the role of a Lyapunov function)*
3. *$\sup_{\mathbf{x} \in \mathcal{X}} L(\mathbf{x}) < \mathrm{P} \in \Re$*
4. *$\inf_{\mathbf{x} \in \mathcal{X}} L(\mathbf{x}) > \mathrm{M} \in \Re$*
5. *Within $\mathcal{X}$, the system is controlled so that for some fixed q, $0 < q \leq 1$, with probability at least q, $L(\mathbf{x}(\mathrm{i}+1)) - \mathrm{L}(\mathbf{x}(\mathrm{i})) \leq -\delta$ for each $i = 0, 1, 2, \ldots$ and for some fixed $\delta > 0$. (I.e. during each unit time step, with probability q, L decreases by at least δ.)*

Then any trajectory beginning in $\mathcal{X}$ will, with probability one, enter the target region at some finite time.

Proof: We show that the probability of never entering the target region is zero.
During any period of $\lceil \frac{P-M}{\delta} \rceil$ time units, with probability at least $q_0 = q^{\lceil \frac{P-M}{\delta} \rceil} > 0$ the system will enter the target region regardless of its state at the beginning of the period. This is because $L(\mathbf{x}(\mathrm{t})) < \mathrm{P}$ at the start of the period, and during each time step there is chance q that L decreases by at least δ. If L decreases by δ for $\lceil \frac{P-M}{\delta} \rceil$ time steps in a row, we have $L(\mathbf{x}(\mathrm{t})) < \mathrm{M}$ afterward. This impossible for $\mathbf{x} \in \mathcal{X}$, so the trajectory must have entered the target region.

For a trajectory to never enter the target region, the probability $q_0 > 0$ event described above must not happen for any of the infinitely many blocks of $\lceil \frac{P-M}{\delta} \rceil$ time units during the trajectory, which happens with probability zero. QED.

This theorem proves the stability of controllers under rather lenient conditions. All a controller needs is to have some chance of decreasing the Lyapunov function L by some minimal amount per time step. The chance and the amount of decrease can be quite small, and the rest of the time the controller can act on the system in any way at all. Still, that the system will eventually reach the target region is assured with probability one.

We describe three different "StaRL" control architectures, which achieve the conditions of Theorem 1 in different ways. To help us concisely state our control architectures, we first define the maximum possible change in L during unit time:

$$\Delta L_{\mathbf{Z}}(\mathbf{x}) = \sup_{\mathbf{x}' \in \mathrm{A}_{\mathbf{Z}}(\mathbf{x})} (\mathrm{L}(\mathbf{x}') - \mathrm{L}(\mathbf{x})),$$

where $\mathbf{Z}$ is any controller, $\mathbf{x} \in \mathcal{X}$ is a state of the system, and $A_{\mathbf{Z}}(\mathbf{x})$ is the set of all system states that may result when $\mathbf{Z}$ controls the system for one time unit starting from state $\mathbf{x}$.

Method StaRL1:

- Choose $\mathcal{X}$ and L to satisfy the conditions 1, 2, 3, and 4 of Theorem 1.
- Constrain the control choices of an RL controller $\mathbf{Y}$ so that $\Delta L_{\mathbf{Y}}(\mathbf{x}) \leq -\delta_0$ for all $\mathbf{x} \in \mathcal{X}$ and some $\delta_0 > 0$.

StaRL1 satisfies Theorem 1 with the particular choice of $q = 1$ for conditions 5. Because L is decreased by a minimal amount on every time step, a StaRL1 controller is actually guaranteed to bring the system to target in bounded time $\lceil (P - M)/\delta \rceil$ from any starting state. However, the condition of a minimum decrease in L at all times may reduce the ability of the controller to optimize cost. The next StaRL architecture relaxes the requirement of descent at every time step.

Method StaRL2:

- Choose $\mathcal{X}$ and L to satisfy the conditions 1, 2, 3, and 4 of Theorem 1.
- Let $\mathbf{W}$ be any controller satisfying $\Delta L_{\mathbf{W}}(\mathbf{x}) \leq -\delta_0$ for all $\mathbf{x} \in \mathcal{X}$ and some $\delta_0 > 0$.
- Constrain the control choices of an RL controller $\mathbf{Y}$ so that $\Delta L_{\mathbf{Y}}(\mathbf{x}) \leq 0$ for all $\mathbf{x} \in \mathcal{X}$.
- Choose $0 < p < 1$.
- Control the system as follows: at each time step, with probability p let $\mathbf{Y}$ control the system for the next time step, and otherwise let $\mathbf{W}$ control.

The RL controller $\mathbf{Y}$ can now be viewed as a subsystem of an overall controller which switches between the RL and another controller. Instead of requiring a fixed amount of descent per time step on L, we only require such descent with some probability, but the RL subsystem is still constrained not to ascend on L. Our last control architecture affords its RL subsystem maximum freedom.

Method StaRL3:

- Choose $\mathcal{X}$ and L to satisfy the conditions 1, 2, 3, and 4 of Theorem 1.
- Let $\mathbf{W}$ be any controller satisfying $\Delta L_{\mathbf{W}}(\mathbf{x}) \leq -\delta_0$ for all $\mathbf{x} \in \mathcal{X}$ and some $\delta_0 > 0$.
- Let $\mathbf{Y}$ be an RL controller, unconstrained except by the admissibility conditions of the control problem.
- Choose $0 < p < 1$.
- Control the system as follows: at each time step, with probability p let $\mathbf{Y}$ control the system for the next time step, and otherwise let $\mathbf{W}$ control.

The StaRL2 and StaRL3 architectures differ on whether the RL subsystem must be non-ascending on L. This makes no difference in the stability theorem we have presented, though it can make a difference in other contexts. For instance, in some problems one wishes the system to approach a target point as time goes to infinity or to stay in the vicinity of a target (system regulation). If a controller never ascends on L, then whenever the system state enters some level set of L we know the trajectory will never leave that level set again. Because StaRL3 can go up or down on L, we cannot ensure the system state will stay in any particular region of state space. Thus, StaRL3 does not immediately generalize to approach-the-target or regulation type problems, as StaRL1 and StaRL2 do.

The StaRL architectures can be expected to differ in their practical performance. Because StaRL2 never increases L, it needs to use the $\mathbf{W}$ controller a total of $\lceil (P - M)/\delta \rceil$ times to ensure getting to target. A StaRL3 controller must use $\mathbf{W}$ for $\lceil (P - M)/\delta \rceil$ time steps *in a row* to ensure entering the target region. A StaRL1 controller decreases L on every time step, so is certain to reach target within time $\lceil (P - M)/\delta \rceil$ from any starting point. If we assume worst-case behavior of the RL subsystem, we would thus expect a StaRL1 controller to bring the system to target

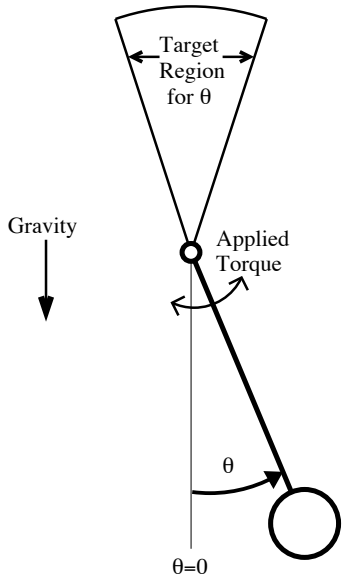

Figure 1. The pendulum swing-up problem.

most quickly, followed by a StaRL2 controller, and slowest of all, a StaRL3 controller.

StaRL2 and StaRL3 controllers also have the free parameter p, which can be used to modulate empirical behavior. For high p, a StaRL3 controller behaves almost as if it is an ordinary, unconstrained RL controller. For low p, either of StaRL2 and StaRL3 control according to $\mathbf{W}$ most of the time – and $\mathbf{W}$ has a worst-case time-to-target bound of $\lceil (P-M)/\delta \rceil$.

The three StaRL architectures provide successively more freedom to the RL controller, creating greater opportunity for the RL to optimize the quality of control. All, however, guarantee that the system will be taken to target eventually.

6. Pendulum Swing-Up Example

We illustrate the proposed StaRL methods on a single-link pendulum swing-up task. The problem is to maneuver a pendulum so that it is within a specified angle of upright as quickly as possible (see figure 1). The system is controlled by applying torques at the fulcrum of the pendulum. The pendulum is weightless, of unit length, and has a unit mass at its end. We assume the downward acceleration of gravity to be of unit strength. The control differential equations that govern the system are:

$$\frac{d}{dt}\theta = \dot{\theta}$$

$$\frac{d}{dt}\dot{\theta} = -\sin\theta + u$$

where θ is the angular position of the pendulum, $\dot{\theta}$ the angular velocity, and u the instantaneous control torque. The target region is defined as $|\theta| \geq \theta_{\text{target}}$. The control torque u is of bounded magnitude, $|u| \leq u_{max} = 0.12$, so that the pendulum cannot be driven straight up to the target. Instead, the pendulum must be swung back and forth a number of times until enough kinematic energy builds up to allow it to swing up into the target region. Because we are treating this as a minimum-time problem, we define the cost function as $g(\theta, \dot{\theta}) = 1$.

6.1 Lyapunov Analysis of the Pendulum

We begin by describing our choices of $\mathcal{X}$ and L for the pendulum. We define $\mathcal{X}$ implicitly as the set of all states reachable by admissible control from the state in which the pendulum is at rest, $(\theta, \dot{\theta}) = (0, 0)$, without entering the target region.

For L, we use the negative of the mechanical energy of the pendulum (potential energy plus kinetic energy). Mechanical energy is zero when the pendulum sits at rest. Sufficiently increasing mechanical energy ensures entering the target region. We define:

$$L(\theta, \dot{\theta}) = -1 + \cos(\theta) - \frac{1}{2}\dot{\theta}^2$$

To save space, we do not prove that L is bounded above and below on $\mathcal{X}$, conditions 3 and 4 of Theorem 1. These facts are easily established.

6.2 Stable control of the Pendulum

For a controller to be stable, we at least want it to be non-ascending on L. For fixed $\theta, \dot{\theta}$, and u, the time derivative of L is:

$$\dot{L}(\theta) = -sin(\theta)\dot{\theta} - \dot{\theta}(-\sin\theta + u) = -\dot{\theta}u$$

So, to achieve $\dot{L} \leq 0$, u should be of the same sign as $\dot{\theta}$. In other words, torque is applied in the same direction that the pendulum is moving. This does not ensure stability by itself; other conditions on control torque are required for proving stability.

Note that the obvious control law of choosing $u = -u_{max}$ or $u = +u_{max}$ to match the sign of $\dot{\theta}$, thus maximizing the instantaneous decrease of L at all times, is *not* guaranteed to reach target, and certainly does not produce minimum-time control.

Theorem 2 *If* $\mathbf{Z}$ *is any admissible controller that satisfies:*

1. *L decreases along all possible trajectories. (I.e. $L(\mathbf{x}(t_1)) > L(\mathbf{x}(t_2))$ for any $0 < t_1 < t_2 < \infty$.)*

2. *Pendulum acceleration is bounded away from zero.* ($|\ddot{\theta}| = |-\sin\theta + u| \geq \gamma > 0$.)

3. *During any unit time step there is a period of duration at least* δ_1, $0 < \delta_1 \leq 1$, *during which control torque is continuous in time and bounded away from zero.*

then $\Delta L_{\mathbf{Z}}(\theta, \dot{\theta}) \leq -\delta_0$ *for some* $\delta_0 > 0$.

The proof of this theorem is presented in the appendix.

Theorem 2 directly provides the conditions which we use to constrain the action choices for the RL subsystem in our StaRL1 controller, which we discuss below. The theorem also allows us to develop a specific control law to play the role of "$\mathbf{W}$" in our StaRL2 and StaRL3 controllers. We call this control law MEA, for Modified Energy Ascent. It minimizes $\dot{L}$ at each time instant, unless doing so would cause $\ddot{\theta}$ to be too close to zero, in which case it chooses a different torque that makes $|\ddot{\theta}|$ bigger but still decreases L. Define:

$$s(\theta, \dot{\theta}) = \begin{cases} +1 & \text{if } (\dot{\theta} > 0) \text{ or } (\dot{\theta} = 0 \text{ and } \theta \geq 0) \\ -1 & \text{otherwise} \end{cases}$$

$$\text{MEA}(\theta, \dot{\theta}) = \begin{cases} s(\theta, \dot{\theta}) u_{max} & \text{if } |-\sin\theta + \\ & s(\theta, \dot{\theta}) u_{max}| > 0.01 \\ \frac{1}{2} s(\theta, \dot{\theta}) u_{max} & \text{otherwise} \end{cases}$$

Theorem 3 *MEA satisfies the conditions of Theorem 2.*

For space reasons, we omit proof of this theorem.

6.3 StaRL experiments on the pendulum

We tested the proposed StaRL architectures on the pendulum swing-up task in simulation. For the RL (sub)system of each, we used the Q-learning algorithm. The StaRL2 and StaRL3 architectures were tested for several choices of p, the probability of control by the RL controller $\mathbf{Y}$: 0.2, 0.4, 0.6, 0.8, and 0.99. We also tried $p = 1.0$, which corresponds to Q-learning with no "$\mathbf{W}$" controller.

The Q-learning in the StaRL1 controller was allowed to choose from two control actions: $w_1 = \frac{1}{2} s(\theta, \dot{\theta}) u_{max}$, and $w_2 = s(\theta, \dot{\theta}) u_{max}$. However, we ruled out a control action w_i if it would cause $|\ddot{\theta}| = |-\sin\theta + w_i| < 0.01$. Further, if the $|\ddot{\theta}|$ condition began to be violated while executing a control action, the other control action, which did not violate the condition, was used for the rest of the time step. These actions and the additional constraints satisfy Theorem 2, and thus meet the conditions of the StaRL1 architecture.

For StaRL2, the Q-learning controller had three control action choices: $w_1 = 0, w_2 = \frac{1}{2} s(\theta, \dot{\theta}) u_{max}$, and $w_3 = s(\theta, \dot{\theta}) u_{max}$. This ensures non-ascent on L. The $\mathbf{W}$ controller was MEA.

The StaRL3 Q-learner had five action choices: $w_1 = -u_{max}, w_2 = -\frac{1}{2} u_{max}, w_3 = 0, w_4 = \frac{1}{2} u_{max}$, and $w_5 = u_{max}$. Again, MEA was used as the $\mathbf{W}$ controller.

For all of the architectures, the Q-values of each action w_i were approximated by separate CMACs (Albus, 1981) as a function of the state variables θ and $\dot{\theta}$. Each CMAC divided the state variables into 100 bins (10x10) per tiling, and had 50 tilings. The learning rate for the CMAC was 0.1.

Exploration was ϵ-greedy, with $\epsilon = 0.05$. That is, whenever the Q-learner is given control of the system, with probability ϵ it chooses a random allowable control action, and otherwise it chooses the control action currently estimated to be best.

For StaRL2 and StaRL3, the Q-learning backups were computed from one RL subsystem decision point to the next. If MEA controlled the system for some time between RL control, this was interpreted from the Q-learner's point of view as uncontrolled transitions of the pendulum. So, whenever the Q-learning controller had control of the system in state $(\theta, \dot{\theta})$, chose a control w_i, and, possibly after some intervening control by MEA, was again given control in state $(\theta', \dot{\theta}')$ having accumulated C cost along the way, the value:

$$C + \min_{w_j} \hat{Q}(\theta', \dot{\theta}', w_j)$$

was incorporated into the estimated Q-value $\hat{Q}(\theta, \dot{\theta}, w_i)$. For this minimum-time problem, C is just the time elapsed between being in state $(\theta, \dot{\theta})$ and state $(\theta', \dot{\theta}')$. In this way, Q-learning attempts to learn the optimal policy given the fact that the other controller, MEA, is sometimes in control of the system.

For each architecture and each choice of p, 20 independent learning runs were performed. Each consisted of 100,000 learning trials. In each learning trial, the pendulum began at rest, $(\theta, \dot{\theta}) = (0, 0)$. Trials ended when the pendulum entered the target region. Every 100^{th} training trial, we turned off the learning and exploration of the Q-learning and performed 20 test trials to evaluate the current learned behavior of the StaRL controller.

6.4 Results

We begin by comparing four algorithms, the MEA controller alone, StaRL1, StaRL2 with $p = 0.99$, and StaRL3 with $p = 0.99$. Figure 2 summarizes the mean times to goal of each at the beginning and end of learning, averaged across the independent runs. The upper left panel plots the time to goal during the first block of 100 learning trials, and the upper right panel corresponds to the first block of 20 test trials (which immediately followed the

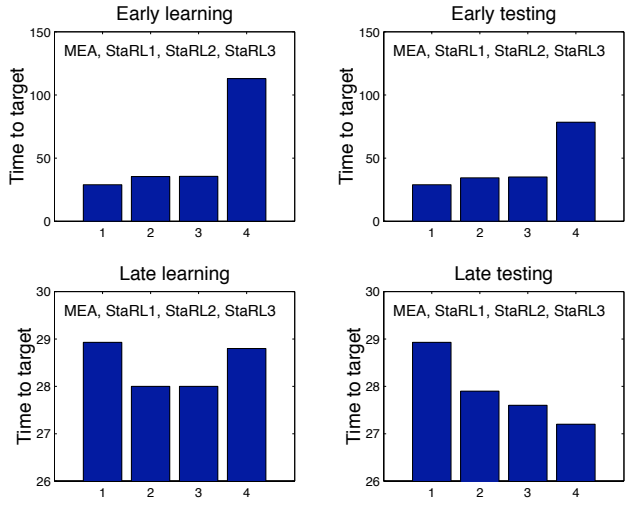

Figure 2. Performance summary of MEA, StaRL1, StaRL2 (p=0.99), and StaRL3 (p=0.99)

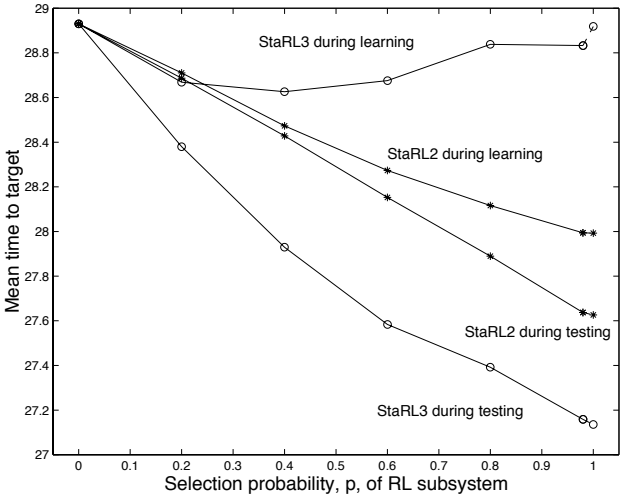

Figure 4. Late performance of StaRL2 and StaRL3

100 learning trials). The lower plots correspond to the last block of 100 learning trials (trials 99,901-100,000) and the final 20 test trials.

Early in learning, StaRL1 and StaRL2 perform nearly identically. This is not surprising. Given our parameter choices, both controllers are nearly just Q-learners constrained to descend on our Lyapunov function. StaRL3 has much worse initial performance, corresponding to its greater freedom in control choices. Late in learning, the story is similar, although all the learning controllers are now better than the Lyapunov-based controller MEA alone. The asymptotic performance during test trials tells a different story. Here, we see the expected ordering of algorithms, with the least constrained controller, StaRL3, producing best performance.

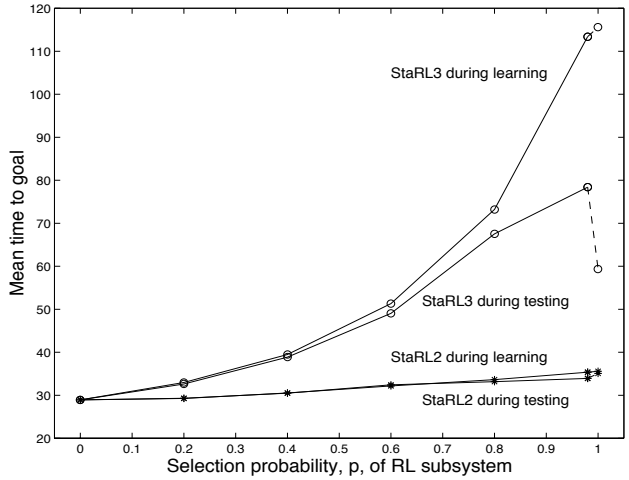

Figure 3. Early performance of StaRL2 and StaRL3

The next two figures, 3 and 4, summarize the effect of the selection probability p in the StaRL2 and StaRL3 controllers. We have included data points for $p = 0.0$, just MEA, and $p = 1.0$. StaRL3 with $p = 1.0$ amounts to ordinary, unconstrained Q-learning.

Early in learning, the effect of p is clear and as expected. Increasing dependence on the little-experienced RL subsystem decreases performance. StaRL3 performs much worse than StaRL2 because its RL subsystem is unconstrained.

Late in learning (figure 4), StaRL2 has monotonically improving performance in p. StaRL3's time to goal during learning trials is strangely unaffected by p, hovering around the performance of MEA. However, StaRL3's test performance improves with increasing p as expected, reaching the best performance of any of our learned controllers, and improving on MEA's time to goal by about 6%.

7. Conclusions

We have proposed new methods for designing RL control systems with provable performance guarantees. The StaRL control architectures ensure that the controlled system will be brought to a target state by building Lyapunov-based constraints into the controller. At the same time, RL is used to optimize the quality of control according to a cost function. The StaRL stability guarantees hold independently of many characteristics of the RL, such as method of function approximation, and apply during learning as well as after learning.

We demonstrated the StaRL approach on a simple pendulum swing-up problem. Simulation experiments revealed some of the empirical benefits, such as dramatically im-

proved initial performance compared to standard RL methods. The experiments also showed that the constraints built into the StaRL controllers, while providing stability guarantees and good initial performance, sometimes limit asymptotic performance.

We believe these methods improve the suitability of RL for on-line learning and control in real-world systems – systems where costs can be high, and prolonged learning or outright failure must be avoided.

It is interesting to note that there is nothing about the StaRL-type architectures that actually requires the use of RL. Any heuristic controller that seems to work well for a problem could be interleaved with a provably stable controller to achieve good practical performance with stability guarantees. This heuristic need not be a learning controller at all. The ideas of constraining control choices and interleaving control seem to have wide application in making stable control systems from components that are not necessarily stable by themselves.

8. Future Work

All of the StaRL control architectures proposed here potentially suffer from reduced asymptotic performance in exchange for stability guarantees. We are now investigating architectures that would not be limited in this way. One simple idea is to use a StaRL3 architecture but let chance that the RL subsystem is in control, p, go to 1. On any particular trial, $p < 1$ ensures stability, but in the limit the controller is unconstrained.

We are also developing StaRL architectures and stability theorems for problems in which conditions as strong as those in Theorem 1 cannot be met, and for other types of control problems, such as regulation problems, where the system state is to be kept near to a desired point.

Acknowledgments

This work was supported by the National Science Foundation, grant ECS-9980062; the Air Force Office of Scientific Research, grant F49620-96-1-0254; and a fellowship from the McDonnell-Pew Program in Cognitive Neuroscience, grant JSMF 96-25. We also thank Michael Rosenstein, Daniel Bernstein, and Doina Precup for very helpful comments on the manuscript.

References

Albus, J. S. (1981). *Brain, behaviour and robotics*, chap. 6. Byte Books.

Baird, L. C. (1995). Residual algorithms: Reinforcement learning with function approximation. In *Proc. of the Twelfth Int. Conf. on Machine Learning*, pp. 30–37. Morgan Kaufmann.

Bertsekas, D. P., & Tsitsiklis, J. N. (1996). *Neuro-Dynamic Programming*. Athena Scientific, Belmont, MA.

Sutton, R. S., & Barto, A. G. (1998). *Introduction to Reinforcement Learning*. MIT Press/Bradford Books.

Vincent, T. L., & Grantham, W. J. (1997). *Nonlinear and Optimal Control Systems*. John Wiler & Sons, Inc., New York.

Watkins, C. J. C. H. (1989). *Learning from Delayed Rewards*. Ph.D. thesis, Cambridge University, Cambridge, England.

Appendix

Proof of Theorem 2: We want to show that $\Delta L_{\mathbf{Z}}(\theta, \dot{\theta}) \leq -\delta_0$ for some fixed $\delta_0 > 0$ and all $(\theta, \dot{\theta})$.

For any $(\theta_0, \dot{\theta}_0)$, let $(\theta(t), \dot{\theta}(t)), 0 \leq t \leq 1$ be any arbitrary possible trajectory generated by $\mathbf{Z}$ applying controls $u(t)$.

$$\Delta L_{\mathbf{Z}}(\theta, \dot{\theta}) = \int_{t=0}^{1} \dot{L}(\theta(t), \dot{\theta}(t))dt = \int_{t=0}^{1} -\dot{\theta}(t)u(t)dt$$

Condition 3 assures us that there is a period of duration at least δ_1 during which $u(t)$ is continuous and bounded away from zero. Let us assume wlog that this period begins at t=0. Since L is always decreasing (condition 1), we can throw away the part of the integral after δ_1.

$$\Delta L_{\mathbf{Z}}(\theta, \dot{\theta}) \leq \int_{t=0}^{\delta_1} -\dot{\theta}(t)u(t)dt \leq -\beta \int_{t=0}^{\delta_1} |\dot{\theta}(t)|dt$$

The last step comes from $u(t)$ being bounded away from zero and L decreasing, hence $u(t)$ matches $\dot{\theta}(t)$ in sign.

If $|\dot{\theta}(t)|$ stays above some small $\epsilon > 0$, then we can bound the integral by substituting ϵ for $\dot{\theta}(t)$, yielding the bound $\Delta L_{\mathbf{Z}}(\theta, \dot{\theta}) \leq -\beta\epsilon\delta_1$. Otherwise, $|\dot{\theta}(t)|$ goes below ϵ, but we show that it can only remain small for a short time.

How does $|\dot{\theta}(t)|$ change over time? $\ddot{\theta} = -\sin\theta + u$ is bounded away from zero, and since θ and u are changing continuously in time, $\ddot{\theta}$ must maintain the same sign for the whole period $t \in [0, \delta_1]$. Thus $\dot{\theta}$ is either monotonically increasing or decreasing. During a period when $|\dot{\theta}| < \epsilon$, $\dot{\theta}$ can change by at most 2ϵ – from $-\epsilon$ to $+\epsilon$ or vice versa. Since this change happens at rate $|\ddot{\theta}| \geq \gamma$, $|\dot{\theta}|$ can remain less than ϵ for no longer than $2\epsilon/\gamma$. For sufficiently small ϵ, we have $(\delta_1 - 2\epsilon/\gamma) > 0$, so we get the bound

$$\Delta L_{\mathbf{Z}}(\theta, \dot{\theta}) \leq -\beta \int_{t=0}^{\delta_1} |\dot{\theta}(t)|dt \leq -\beta\epsilon(\delta_1 - 2\epsilon/\gamma)$$

by breaking the integral into cases where $|\dot{\theta}|$ is less than or greater than ϵ. QED.