Lyapunov Methods for Reinforcement Learning: A PhD Thesis Proposal Draft

University of Massachusetts
Department of Computer Science

Technical Report UM-CS-1999-60

Theodore J. Perkins

November 28, 1999

Abstract

In this thesis, I propose to investigate a novel approach to injecting prior knowledge to reinforcement learning (RL) systems for minimum cost-to-target control problems. The approach utilizes Lyapunov descent ideas from control theory to constrain the action choices of an RL controller. Such constraints can improve on-line performance and accelerate learning. The RL controller reaches the target much more quickly than an unconstrained system, and learning is focussed on "good" actions – those that lead the system state towards the target.

Further, appropriately formulated constraints can provide theoretical guarantees on reaching the target on every trial, and sometimes even worst-case time bounds. These guarantees follow from the action constraints and hold independently of many details of the RL controller, including the method of function approximation. In fact, learning may fail entirely, but a certain level of performance is maintained by the constraints imposed on the controller.

1 Introduction

Reinforcement learning (RL) [65] (heuristic dynamic programming [81], neurodynamic programming [9]) is generating increasing attention in artificial intelligence, control engineering, and operations research. The term RL stands for a collection of methods for approximating solutions to stochastic optimal control problems. RL methods are typically used when these problems are too complex or ill-defined for classical solution methods, such as dynamic programming (DP). RL methods are generally easy to implement, and are not computationally intensive – features which add to their popularity.

Because RL techniques provide decent, approximate solutions to problems that would be literally impossible to solve by classical methods, RL can be considered "infinitely faster" than such methods. In fact, though, RL is not known for speed. Perhaps the most famous application of RL methods, a world-class backgammon-playing program called TD-gammon, was the result of months of continuous computer time simulating backgammon games [71]. The elevator scheduling system of Crites and Barto [13] took approximately a year to produce, including all the time it took to tune the representation and neural network parameters by hand. Sutton [64] noted in studying a small-scale dynamical minimum time-to-target task, that an RL controller can take a long time before it reaches a target state for even the first time. The initial behavior of RL systems is often largely undirected, resulting in much wandering about the state space before a target state is found.

In the framework I propose to study, an RL controller's choices are constrained by prior knowledge in the form of a Lyapunov function. The RL controller may be required to descend on the Lyapunov function, or the constraints may be softer, as in requiring descent with some probability. Initial performance of the system can be quite good, because the Lyapunov function leads the controller toward the target, and learning is more rapid because it is focussed from the start on good control choices.

One of the particular strengths of RL is its compatibility with function approximation. This facilitates generalization across combinatorially huge/continuous state spaces, allowing decent performance to be obtained on large problems. There are few theoretical guarantees, however, on RL systems utilizing function approximation.

A handful of provably convergent learning algorithms have been proposed [78, 74, 75, 4, 3, 66] (meaning the learning process converges, not that the controller is guaranteed to converge to a target state). In the case of linear function approximation, there are associated worst-case performance bounds [78, 74, 75]. Other results guarantee only the convergence of free parameters to some local optimum of unknown quality. None of these algorithms are yet in wide use. The most commonly used RL algorithms have actually been shown to be divergent in some cases [73, 3], and no nontrivial performance bounds are known.

Using Lyapunov functions to constrain an RL system, one can theoretically guarantee the system will reach a target state on every trial. In some cases, one can derive explicit worst-case time-to-target bounds, based on the constraining Lyapunov functions. When a control problem is minimum time, this constitutes a performance bound in the primary cost metric. To my knowledge, these are the first performance bounds for RL systems that are independent of the methods of function approximation.

2 Minimum Cost-To-Target Problems

In the problems I will study, the system state is described by a vector $\mathbf{x} \in \mathcal{X}$, where $\mathcal{X}$ is a continuous subset of $\Re^n$. In each state $\mathbf{x}$, there is an allowable set of controls $\mathcal{U}(\mathbf{x}) \subseteq \Re^{\mathrm{m}}$. The system evolves in time according to a control differential equation,

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

where y is a random process and $\mathbf{u} \in \mathcal{U}(\mathbf{x})$ is the control command.

Any trajectory through state space, $(\mathbf{x}(\mathrm{t}), \mathbf{u}(\mathrm{t}))$ for $0 \leq t < \infty$, has an associated cost

$$J[\mathbf{x}(\cdot), \mathbf{u}(\cdot)] = \int_{\mathrm{t}=0}^{\infty} \mathrm{g}(\mathbf{x}(\mathrm{t}), \mathbf{u}(\mathrm{t}))\mathrm{dt},$$

where g is an instantaneous-cost function, mapping states and control signals to the nonnegative reals: $g : \{(\mathbf{x}, \mathbf{u}) | \mathbf{x} \in \mathcal{X} \text{ and } \mathbf{u} \in \mathcal{U}(\mathbf{x})\} \mapsto \{\Re^+ \cup \{0\}\}$.

In a minimum cost-to-target problem, there is a target region $\mathcal{X}_T \subset \mathcal{X}$. Once the system enters this region, no further costs are incurred. The objective in such a problem is to find a control policy that moves the system into the target region along minimum-cost trajectories.

3 Lyapunov Stability and Control

Lyapunov methods offer a unifying framework for designing controllers for (nonlinear) dynamical systems. Lyapunov-designed controllers have desirable stability properties, but they do not explicitly optimize any measure of the system's performance. Lyapunov control design is based on the intuitive idea that a "dissipative" system will always converge to a state of minimum energy. The Russian mathematician Lyapunov [36] formally introduced this idea more than a century ago. In the West, his work was essentially ignored until around 1960. Since then, however, it has become one of the fundamental tools in control theory.

3.1 Provably Stable Control

Lyapunov's original theorems addressed the stability of a dynamical system. For example, a slightly extended form of one of the Lyapunov theorems can be stated as follows:

Theorem 1 *The equilibrium solution of an autonomous dynamical system* $\dot{\mathbf{x}} = \mathbf{f}(\mathbf{x})$, $\mathbf{x} \in \Re^n$, *is globally asymptotically stable if there is a continuously differentiable, radially unbounded function* $L(\mathbf{x})$ *such that for all* $\mathbf{x} \in \Re^n$, $L(\mathbf{x})$ *is positive definite and* $\dot{L}(\mathbf{x})$ *is either negative definite, or it is negative semidefinite with the additional requirement that no solution can stay forever in the set* $\{\dot{L}(\mathbf{x}) = 0\}$ *other than the trivial solution* $\mathbf{x} \equiv 0$.

After choosing a control law, a function specifying a control choice for every state, the combination of dynamical control problem and control law can be viewed as an (uncontrolled) dynamical system. Thus, an approach to creating provably stable controllers is to:

1. Choose a control law
2. Find a Lyapunov function for the resulting dynamical system, with equilibrium solution point occurring within the target region.

This design procedure ensures that the controller will bring the system to target on every trial.

The basic Lyapunov stability theorem has been extended in many ways (e.g. [16, 17, 27, 28, 33]). For purposes of this thesis, two extensions are particularly important. The first is that a result similar to the above exists for discrete-time dynamical systems [28]. This is important because, in typical RL control methods, control choices are only made at discrete points in time. The second is that related results exist for stochastic dynamical systems (see [35] for a review). Even when the system to be controlled is deterministic, RL algorithms typically exhibit stochastic behavior.

3.2 Lyapunov Optimizing Feedback Control

The two step method described above can also be followed in reverse order. One may start by choosing a candidate Lyapunov function, and then choose a control law so that $\dot{L}$ satisfies the conditions of a Lyapunov stability theorem. In Lyapunov optimizing feedback control, the idea is to choose a control law implicitly, as a function of the Lyapunov function (in particular, as the solution to some optimization problem). For example, in steepest descent control for deterministic systems, the control is chosen so that the state space trajectory follows the gradient of the Lyapunov function as closely as possible:

$$\mathbf{u}(\mathbf{x}) = \arg \min_{\mathbf{u} \in \mathcal{U}(\mathbf{x})} \nabla \mathrm{L} \cdot \frac{\dot{\mathbf{x}}}{||\dot{\mathbf{x}}||}$$

In quickest descent control, the time derivative of the Lyapunov function is minimized:

$$\mathbf{u}(\mathbf{x}) = \arg \min_{\mathbf{u} \in \mathcal{U}(\mathbf{x})} \nabla \mathrm{L}(\mathbf{x}) \cdot \dot{\mathbf{x}}$$

In these approaches, the choice of controller is simplified. However, because the control law is specified in terms of the Lyapunov function, proper choice of a Lyapunov function is all the more important and difficult. (On the other hand, it seems that choosing a Lyapunov-like function and following one of the above approaches can be quite effective in practice, even if descent is not guaranteed in some parts of state space [79].) By design, these controllers are guaranteed to be stable, that is, convergent to a target point or region. However, no explicit attempt is made to minimize cost-to-go.

4 Reinforcement Learning and Real-Time Optimal Control

The term reinforcement learning defines a collection of methods for approximating solutions to stochastic optimal control problems, particularly those that can be formulated as Markov decision processes (MDPs) [21, 65]. Standard dynamic programming (DP) techniques for numerically solving MDPs, such as value iteration or policy iteration, require an explicit model of the process's dynamics and computational resources (time and memory) that are polynomial in the size of the state space, which, in turn, is often exponential in the number of state variables. For problems with a large number of state variables, these methods become computationally infeasible.

A variety of methods for efficiently approximating solutions to MDPs have been developed (e.g. [25, 48, 55]). Among these, RL methods are novel in their combination of Monte-Carlo, stochastic approximation, and function approximation techniques. Specifically, RL methods combine some, or all, of the following features:

1. Unlike classical DP methods, which compute the solution to an MDP off-line, RL methods naturally interleave learning and acting. This permits a RL-controlled system to begin operation long before convergence to the final policy is achieved.
2. Because computation is guided along sample trajectories, RL methods can achieve considerable computational efficiency in situations in which many states have very low probabilities of occurring in actual experience. In this respect, RL is similar to other Monte-Carlo methods in automatically allocating computation in proportion to that computation's influence on the desired result.
3. RL methods simplify the basic DP backup by sampling. Instead of generating and evaluating all of a state's successors, they typically estimate a backup's effect by sampling from the appropriate distributions. When the samples are generated from actual experience, this kind of backup can be applied without prior knowledge of the MDP's transition and reward probabilities.

RL methods have been successfully applied to a diversity of large-scale (stochastic) optimal control problems that are important to industry and government. These include optimal asset allocation [46], the pricing of exotic options [76], elevator dispatching [13], dynamic channel allocation [60], job-shop scheduling [84], production scheduling [58], force-guided assembly [19], and robot navigation [59].

One focus of recent RL research has been the attempt to further increase the computational efficiency of RL systems by incorporating prior domain knowledge (e.g., [12, 34, 56, 70, 77, 82]). The aim of this thesis is to leverage existing engineering knowledge concerning the design of stabilizing controllers for complex dynamical systems. The proposed research develops a general theory and method of incorporating such knowledge into RL systems for solving minimum cost-to-target problems.

5 Lyapunov-Constrained Reinforcement Learning (LCRL)

The principle idea to be studied in this thesis is to use Lyapunov functions to constrain an RL controller's choices in a way that theoretically ensures the system will be controlled to target on every trial. In addition to providing desirable theoretical properties, Lyapunov constraints can improve the practical performance of RL control. During learning, initial performance is good because the controller is guided toward the target, rather than wandering randomly around the state space. Learning is accelerated because it is focussed on "good" actions – those that take the system to target.

Translating a Lyapunov function into action constraints for an RL controller is not trivial. Simply requiring the system state to descend on the Lyapunov function does not guarantee convergence to a target. For example, a controller that moves 1/4 of the way toward a target, then 1/8, then 1/16, etc., with each step taking unit time, only makes it halfway to target given infinite time, yet it is always moving toward the target. RL controllers pose challenges to the standard Lyapunov-style convergence theorems because their control policies are often discontinuous in the state variables, stochastic, and exhibit complex non-stationary behavior.

Although the simplest possible scheme, that of requiring descent, does not have the desired theoretical properties, many schemes can be conceived which do guarantee convergence to target. Part of the work of this thesis will be in exploring such schemes, both theoretically and empirically.

In the remainder of this section I present two general approaches to integrating Lyapunov-style constraints and RL. These approaches are easy to implement and ensure convergence without requiring stronger-than-usual conditions on the Lyapunov functions. At the same time, they are aimed at providing RL with maximum freedom to optimize control.

5.1 Probabilistic switching with non-ascending RL

The first method I propose is to probabilistically switch between a given Lyapunov function-based controller and an RL subsystem that is restricted to be non-ascending on that same Lyapunov function. Intuitively, the non-ascending requirement prevents divergence during RL control, and the Lyapunov-based controller ensures sufficient movement toward the target. I now describe this approach more formally and prove its convergence property under the simplest set of standard Lyapunov assumptions.

Let (f, X, X_T, U, g) be a deterministic control problem. X, the state space, is a continuous subset of $\Re^n$. The target region, X_T consists of a single point $\mathbf{x}_\mathrm{T} \in \mathrm{X}$. U determines which controls are admissible as a function of state. g is the instantaneous-cost function, and f the control differential equation governing the system (*i.e.* $\dot{\mathbf{x}} = f(\mathbf{x}, \mathbf{u})$).

Definition: L is a *Lyapunov-type function* for the control problem if:

1. L is continuous on X
2. $L > 0$ for all $\mathbf{x} \in \mathrm{X} - \mathrm{X_T}$
3. $L(\mathbf{x}_\mathrm{T}) = 0$
4. L is radially unbounded around $\mathbf{x}_\mathrm{T}$.

Definition: A controller $w(x)$ is *convergent and strictly descending* on L if:

1. $\dot{L}(\mathbf{x})$ is continuous on X
2. $\dot{L}(\mathbf{x}) < 0$ on $X - X_T$
3. $\dot{L}(\mathbf{x}_\mathrm{T}) = 0$

where $\dot{L}$ is the time-derivative of L when the system is controlled by $w(\cdot)$.

Theorem 2 *Given: a deterministic control problem, a Lyapunov-type function L, and a controller $w(\mathbf{x})$ that is convergent and strictly descends on L. Suppose the system is controlled according to the following regime. At discrete times $t = 0, 1, 2, \ldots$, with probability $0 < p \leq 1$, $w(\mathbf{x})$ is used to choose control signals for one time unit. Otherwise, any admissible control signals are applied for the next time unit, so long as $\dot{L} \leq 0$. Then with probability one, the system will be controlled to target.*

Proof (sketch): Let $l_0, l_1, l_2, \ldots$ be the value of the Lyapunov function at each of the discrete times. I.e. $l_i = L(\mathbf{x}(\mathrm{i}))$. The sequence $\{l_i\}_{i=1}^{\infty}$ is non-increasing and bounded below, so has a limit. Thus, we know $\lim_{i\to\infty}(l_{i+1} - l_i) = 0$. With probability one, the controller w is used infinitely many times, and during these times the decrease in the Lyapunov function is:

$$l_{i+1} - l_i = \int_{t=i}^{i+1} \dot{L}(\mathbf{x}(\mathrm{t}))\mathrm{dt} \leq \max_{\mathrm{t}\in[\mathrm{i},\mathrm{i}+1]} \dot{\mathrm{L}}(\mathbf{x}(\mathrm{t}))$$

From this, we deduce that at a sequence of points along the trajectory, when $w(\cdot)$ is controlling the system, the time derivative of L goes to zero. These points in state space must have at least one accumulation point, and by continuity of $\dot{L}$ when $w(\cdot)$ is in control, $\dot{L}$ is zero at this accumulation point. Since $\dot{L}$ is only zero at the target point $\mathbf{x}_{\mathrm{T}}$, the target point is the accumulation point. Since overall control is non-ascending on L, other properties of L ensure that trajectories do not just pass arbitrarily close to $\mathbf{x}_{\mathrm{T}}$, but stay close as well.

Using this theorem we can construct a Lyapunov-constrained RL controller by, at fixed time intervals, probabilistically choosing between a given Lyapunov-based controller and a non-ascending RL controller. Convergence to the target is guaranteed with probability one, regardless of the choices made by the RL controller.

5.2 Probabilistic switching with unconstrained RL

One can easily imagine running the scheme of the previous section without the non-ascending requirement on the RL subsystem. If the given controller has a worst-case solution time over the whole state space, then the same switching scheme will, with probability one, achieve a target state in some finite time.

Theorem 3 *Given: a deterministic control problem, and a controller $w(\mathbf{x})$ that has finite worst-case time bound. I.e. ($\sup_{\mathbf{x}\in\mathrm{X}}$ "time to target starting from $\mathbf{x}$, under control of $w(\cdot)$") $\leq B, B > 0 \in \Re$. Suppose the system is controlled according to the following regime. At discrete times $t = 0, 1, 2, \ldots$, with probability $0 < p \leq 1$, $w(\mathbf{x})$ is used to choose control signals for one time unit. Otherwise, any admissible control signals are applied for the next time unit. Then, with probability one, the system state will enter the target region at some finite time.*

Proof (sketch): With probability one, there will eventually be a period of $\lceil B \rceil$ time units in a row during which $w(\cdot)$ is selected to control the system. $w(\cdot)$ can take the system to target from any state in a most B time, so by the end of that time, the system has assuredly entered the target region.

Note that the given controller need not be Lyapunov-based, although this would be a natural way to produce a controller with a provable time bound. In particular, if we have a Lyapunov function asymptotically leading a controller to some point, and the target region surrounds the point, then at some finite time the state will cross the boundary into the target region. And if this time-to-target has a supremum over the state space, then the above theorem can be applied.

In this scheme, the selection probability, p, of the given controller is likely to have a much stronger effect than in the previous scheme. Because the controller is less constrained, initial

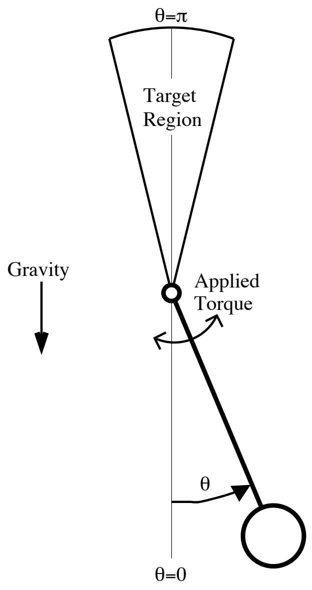

Figure 1: The pendulum swing-up problem.

performance is expected to be worse for low p, but the same freedom may allow better control in the long run. For $p = 1$ the schemes are identical.

6 An LCRL Example: Pendulum Swing-Up

I illustrate LCRL 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. Figure 1 depicts the problem. The system is controlled by applying torques at the fulcrum of the pendulum. 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 position of the pendulum, $\dot{\theta}$ the angular velocity, and u the instantaneous applied torque. The target region is defined as $|\theta| \geq \theta_{\text{target}}$. The control torque u is of limited strength, 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 speed is built up to allow it to swing up into the target region.

6.1 Lyapunov-based control of the pendulum

The negative of total mechanical energy in the pendulum is a Lyapunov-type function for the swing-up problem under a slightly broader, but quite standard, definition than the one presented earlier. It is more natural to think of increasing energy rather than decreasing negative energy, so I will describe total energy as a Lyapunov-type function on which ascent is desired:

$$L_{psu} = \text{Total Mechanical Energy} = \text{Potential} + \text{Kinetic} = (1 - \cos\theta) + (\frac{1}{2}\dot{\theta}^2)$$

This function is minimal when the pendulum is at rest, and is maximized at the points $(\theta, \dot{\theta}) = (\pi, \pm\infty)$, within the target region. Starting from rest, there is a maximal amount of energy the pendulum can acquire without entering the target region. Thus, any controller that increases energy without bound brings the system to target.

A natural controller choice is quickest-ascent on L_{psu}. Its time-derivative is:

$$\dot{L_{psu}} = (\sin\theta)\dot{\theta} + \dot{\theta}\ddot{\theta} = (\sin\theta)\dot{\theta} + \dot{\theta}(-\sin\theta + u) = u\dot{\theta}$$

To maximize $\dot{L_{psu}}$, u should be chosen as the maximum possible torque in the direction of $\dot{\theta}$. When $\dot{\theta} = 0$, the correct choice of u is intuitively clear – the pendulum should be kept swinging. The following three rules summarize this quickest energy ascent controller, which I will denote $EA(\theta, \dot{\theta})$:

- If $\dot{\theta} \neq 0$ apply maximal torque in direction of $\dot{\theta}$.
- If $\dot{\theta} = 0$ and $\theta = 0$ apply maximal clockwise torque.
- If $\dot{\theta} = 0$ and $\theta \neq 0$ apply maximal torque toward $\theta = 0$ position.

That $\dot{L_{psu}}$ can be zero outside the target region is a minor concern. Standard extensions to the basic Lyapunov stability theorems can handle such cases, as long as the system state passes through these points and does not loiter, and a similar extension to theorem 2 proves convergence of a constrained switching-style controller.

Worse, however, is that the EA controller has a sort of unstable equilibrium point. Assuming the maximum torque is not sufficient to drive the pendulum right up to target, there will be some position where the torque exactly balances the force of gravity, and the pendulum could stay motionless forever. At such a point, the controller actually torques the pendulum in the opposite direction, so the controlled system does not have a true equilibrium point there. However, approached from below, it can behave as an equilibrium point.

It is possible to construct a controller that has no such false equilibria at all. For instance, by modifying EA to use some other torque in an ϵ-ball around the pseudo-equilibrium points. For simulation purposes, I simply used EA, and never encountered a problem.

A worst-case time-to-target bound holds for the pendulum swing-up problem, at least as controlled by the modified EA described above. I have already stated that there is a maximum energy achievable outside the target region. One can show, by various simple arguments, that the energy of the system increases at least by some $\delta > 0$ on every swing, thus a finite number of swings back and forth is guaranteed to bring the system to target. And lastly, that swings cannot take arbitrarily long implies that there is some worst-case time by which the system will reach target, regardless of starting state. Such a bound means that a switching controller of the sort described in Theorem 3 is appropriate.

6.2 Experiments

The controlled pendulum system was simulated using Euler's method with a time step of 0.01 time units. For each algorithm and parameter settings described below, 20 independent learning runs were performed. Each learning run consisted of 100,000 learning trials. On each trial, the pendulum started at rest, $(\theta, \dot{\theta}) = (0, 0)$. Trials ended when the pendulum entered the target region, or 300 time units had elapsed. (Ending long-running trials without reaching target is better for the RL and speeds experiments too. 300 time units is approximately 10 times as long as needed to swing the pendulum up by the EA controller.)

For purposes of RL, the allowable control torques were limited to the discrete set {-0.09, -0.045, 0, 0.045, 0.09}. The RL method I used was ϵ-greedy Q-learning. Whenever the RL is given control of the system, with probability ϵ it chooses a random allowable control, and otherwise it chooses the control currently estimated to be best. The values of actions were estimated using separate function approximators, each a CMAC with 10 by 10 bins on the two state variables, and 50 tilings.

The Q-learning backups were computed from one RL decision point to the next. If the Lyapunov-based controller controlled the system for some time between RL decision points, this was interpreted from the RL controller's point of view as uncontrolled transitions of the pendulum. So, suppose the RL controller is given control of the system in state s, chooses a control torque a, and, after some or no intervening Lyapunov-based control, RL is again given control in state s' having accumulated C cost along the way. The value

$$C + \max_{a' \in U(s')} Q(s', a')$$

is incorporated into the estimated Q-value $Q(s, a)$. For this minimum-time problem, C is just the time elapsed between s and s'. In this way, Q-learning attempts to learn the optimal policy given the fact that the other, Lyapunov-based controller is sometimes in control of the system.

The experiments I report used $\epsilon = 0.05$ and $\alpha = 0.1$ (the learning rate for the CMAC). I tested control regimes that switched probabilistically between EA, with probability p, and either unconstrained Q-learning (UncQ) or Q-learning that was constrained to ascend on L_{psu}(AQ). I ran experiments for six different choices of p: 0.8, 0.6, 0.4, 0.2, 0.01, and 0.0. For $p = 0.0$, UncQ is simply ordinary Q-learning.

6.3 Results

After every 100 learning trials, 20 test trials were run, during which there was no learning and no exploration ($\epsilon = 0.0$), to test the current quality of the learned control. Figure 2 summarizes the time to target of during learning and testing when Q-learning is constrained to ascend on $L_{psu}$for several different switching probabilities. Each group of four columns plots, from left to right: 1) the mean time to target during the first 100 learning trials, 2) the mean time to target during the first 20-trial test period, 3) the mean time to target during the last 100 learning trials (of 100,000), 4) the mean time to target during the final 20 test trials.

The probability $p = 1.0$ means the quickest energy ascent controller is selected all the time, whereas $p = 0.0$ means the Q-learning controller is always in control of the system. The convergence theorem, Theorem 2, applies when $p > 0$. Several trends are clear. By looking at the left pair within each group, we see that initial performance decreases as the probability of selecting the Q-learning controller increases. On the other hand, the right pairs of each group improve as Q-learning is chosen more often; thus, as the RL controller is given more control of the system, it is able to produce better and better solutions. Note also that performance on training trials is better than that during learning trials, due at least in part to the random exploration that occurs during learning. These results are all unsurprising, but encouraging.

Figure 3 plots the same values when Q-learning is not restricted to be energy ascending. As before, initial performance is worse the more often Q-learning controls the system. Further, we see that initial performance is far worse than when the Q-learning was constrained to be ascending. This is consistent with the intuition that more 'freedom' leads to worse initial performance. The values plotted here are even a bit generous, because trials that timed out

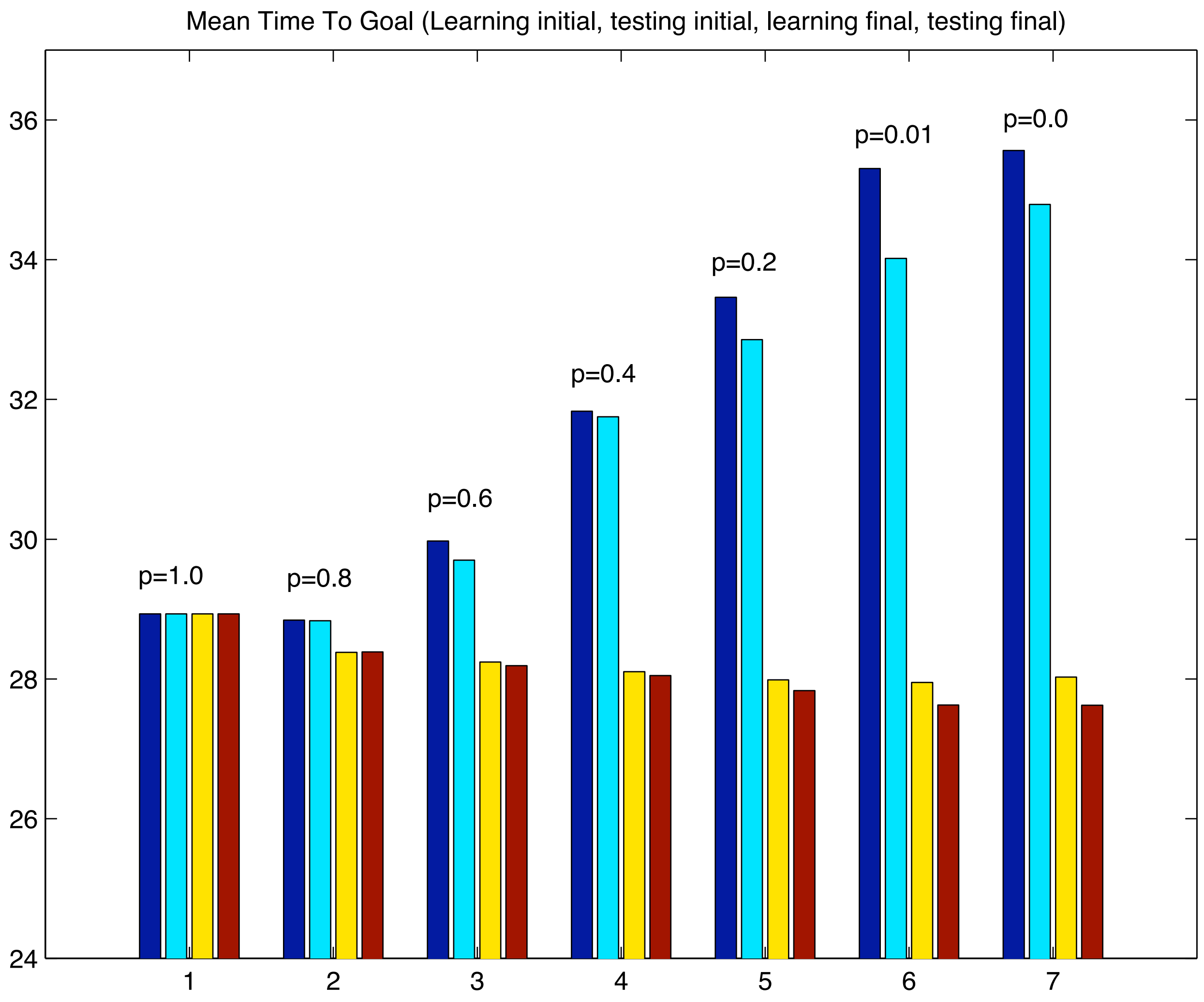

Figure 2: Time to target when Q-learning restricted to be energy-ascending (AQ).

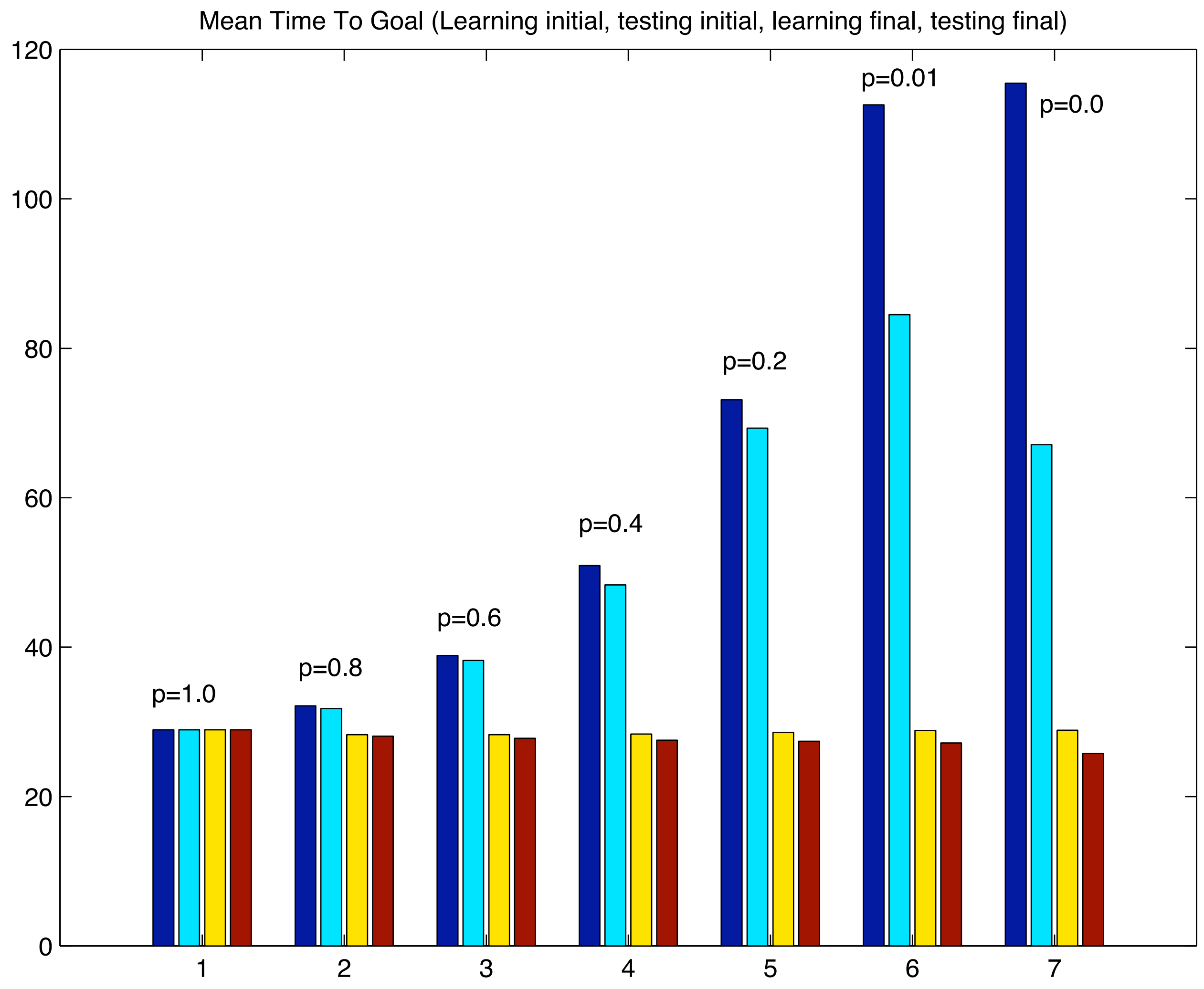

Figure 3: Time to target when Q-learning unconstrained (UncQ).

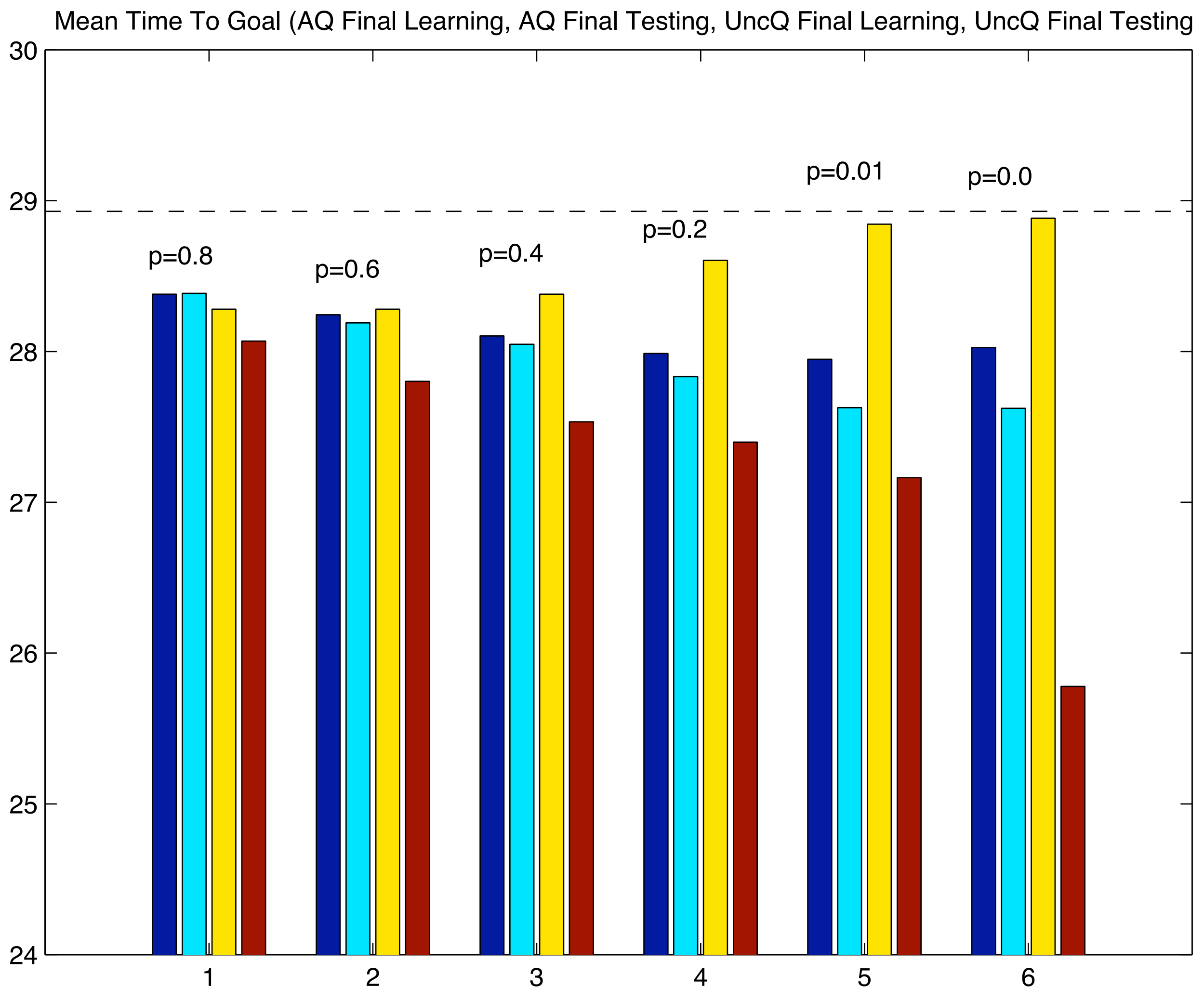

Figure 4: Eventual time to target for AQ and UncQ (after 100,000 learning trials)

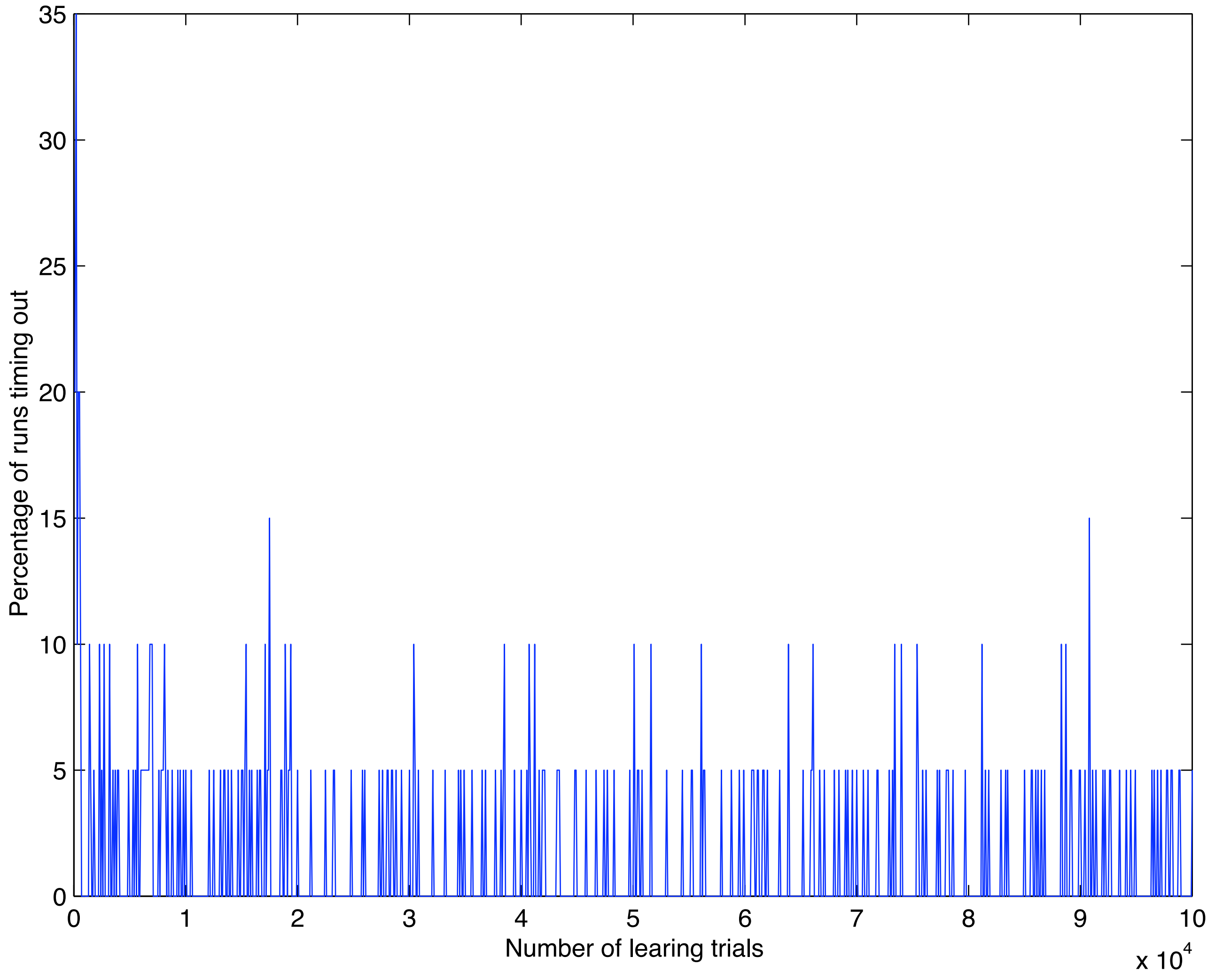

Figure 5: Percentage of test trials timing out for ordinary Q-learning

(the pendulum did not reach target within 300 time units) were not included in the average. I further discuss time outs shortly.

Figure 4 plots the final performance of AQ and UncQ next to each other. In each block of four columns, the left two are AQ's time to target during the final 100 learning trials and final 20 test trials, and the right two columns are the same values for UncQ. We already saw that AQ's performance improves as p decreases. The same is true of UncQ during testing trials, but the opposite holds during learning trials. This is somewhat surprising. After so much learning, the main difference between learning and test trials is just the ϵ exploration probability. It is possible that the unconstrained Q-learning learns an efficient policy that is not robust to disturbances, so that during testing it performs well, but under the randomizing effect of exploration it performs poorly. Pure Q-learning, UncQ with $p = 0.0$, discovers a very good policy, approximately 10% faster than the EA controller, which performs at the black dotted line that crosses the plot.

Finally I discuss the timing-out behavior of the algorithms. AQ never times out, and neither does UncQ when $p \geq 0.2$. Of the 20 independent runs of pure Q-learning (UncQ with $p = 0.0$) and UncQ with $p = 0.01$, 7 and 4 runs respectively had a single learning trial time out, and the other runs never timed out. On testing trials the story is quite different. UncQ with $p = 0.01$ again behaved well. Only two time-outs ever occurred, both in the first block of 20 test trials of one of the runs. Pure Q-learning, however, had (a largely different) 7 of 20 independent runs timing out during the first test period. Further, time-outs continued to occur during testing no matter how much learning took place. Figure 5 plots the percentage of 20 Q-learning runs that time out during testing. (With pure Q-learning, nothing random occurs during test trials, so it either times out or not. There is no "chance" involved.) Over the whole course of learning, approximately 1.3 percent of all testing trials time out. If these trials were averaged into the above performance figures, even at 300 time units, the performance of pure Q-learning would look significantly worse. UncQ with $p = 0.01$, though seemingly little different, might actually be superior. This radical difference between $p = 0$ and $p = 0.01$, and only on test trials, was quite surprising and warrants further analysis.

From all these results, it appears that integrating Lyapunov and RL control seems a promising approach. In the techniques tested here, different parameters allow different tradeoffs between initial and eventual performance. The RL was able to improve significantly on the Lyapunov-based controller, while retaining good theoretical properties.

7 Related Work

The objectives of the proposed work are to obtain performance and safety guarantees for RL systems, to improve initial performance, and to accelerate learning. The Lyapunov function-based constraints are a form of prior knowledge, and also form connections to work in exploration control and Lyapunov optimizing feedback control.

7.1 Prior Knowledge

It is generally recognized that providing prior knowledge to an artificial intelligence system can have a considerable impact on that system's success. Research on this subject within RL has taken a variety of forms.

Research on giving "advice" to RL agents is reported in [77, 12, 37, 38, 34, 70, 82, 18, 56, 2]. Typically, the agent is instructed on which actions are desirable or which actions are to be avoided in certain situations. Researchers have also experimented with incorporating prior knowledge, suitably expressed, directly into the value function approximator [70, 37, 38, 18]. The goals of these works include generality and expressiveness of the advice-giving method, and ease of use. In LCRL, the form of advice-giving is relatively limited; the designer specifies a single Lyapunov function at the start of learning, and that is all. This method, however, makes the application of the prior knowledge to RL control fairly straightforward.

Sometimes, prior knowledge can be expressed simply by the formulation of control choices. In work closely related to that proposed in this thesis, Singh et al. [59] suggest a parameterization of control choices for motion planning in which the controller chooses a linear mixture of distinct Lyapunov functions; the control applied to the system is then the gradient of this mixed Lyapunov function. More generally, recent work in modularity and temporal abstraction has provided solid mathematical foundations for RL when a single control choice can cause the execution of an entire sequence of control actions [52, 53, 54, 49, 50, 15, 67, 69, 68, 20]. This allows a designer to specify intelligent courses of action, which simplifies learning to solve the control problem. Abstractly, the LCRL framework shares a common spirit with this work in that action sets are manipulated for the purpose of improving learning performance.

7.2 Performance Guarantees

To date, there are few theoretical performance guarantees that apply to RL systems. Within the RL community, the only convergence-to-target time bounds appear to be those of Koenig and Simmons [30, 31]. They assume a minimum time-to-target task in a finite, deterministic MDP, and a Q-learning controller [80] using a perfect state table to store its Q-function. By analyzing how the estimated value of actions not leading to a target drops during learning, they are able to bound the total time the Q-learner takes to reach the target state on the first trial.

A handful of results apply to RL systems that use function approximation to estimate the cost-to-go function. Tsitsiklis and Van Roy prove two convergence and error bound results for linear approximation [78, 74, 75]. One result applies to estimating the cost-to-go function of a Markov chain, the other to learning control for an optimal stopping problem. In both cases, they establish probability one convergence of the linear approximation and derive an error bound for the point of convergence.

Several special gradient-based RL algorithms have been proposed for function approximators whose output is continuous in their parameters [4, 3, 66]. For these algorithms, convergence to a local minimum in parameter space of some error function can be guaranteed. However, the convergence of the learning process tells one nothing about how well the resulting controller actually performs. Williams and Baird [83] give straightforward bounds on the suboptimality of performance caused by an imperfect (i.e., approximate) value function, though the bound is in terms of an unknown quantity—the maximum, over all system states, of the difference between the optimal and approximate value functions.

In the LCRL framework, convergence guarantees can be established independently of many details of the learning process and the method of function approximation. The RL system is constrained from the beginning to make choices that will get the system to a target state.

7.3 Safety Guarantees

A good reason to restrict the set of admissible control choices is to impose safety constraints. Singh et al.'s [59] previously mentioned work falls in this category. Although their use of Lyapunov functions differs from that in LCRL, both approaches guarantee collision-free control if the navigation problem is purely kinematic. Neuneier and Mihatsch [57] and Schneider [47] propose learning methods that pay attention to the variability of outcomes, resulting in learned policies that are risk averse. Huber and Grupen [23, 24] apply RL to robotics problems in which the system state is defined by the status of a set of lower-level closed-loop controllers. RL is used to choose which low-level controllers to activate, but, from the start, actions that violate basic safety constraints are eliminated from the set of admissible actions.

7.4 Speed of Learning

Lyapunov constraints focus a learning controller's attention on promising or important parts of the state/action space. This can be expected to lead to a significant speed-up of learning. When function approximation is involved, the issue of the distribution of updates over the state space is particularly important because the approximation will tend to be most accurate for the states receiving the most updates. RL work along these lines has its origin in the theory of asynchronous dynamic programming [8], which shows that value-function backups need not be done in full sweeps across the state space. This result is exploited in Barto et al.'s [5] real-time dynamic programming algorithm, which performs value iteration updates along system trajectories, thus concentrating computation on the most frequently visited states. Moore and Atkeson [42] and Peng and Williams [51] take this a step further and systematically concentrate computation in areas of the state space where large prediction errors occur.

Closely related is work on variable-resolution dynamic programming or RL methods [43, 44, 40, 41, 39]. These methods adaptively increase the discriminatory power, or resolution, of the function approximation in parts of the state space where it is most needed.

7.5 Exploration Control

Lyapunov constraints can be viewed as a method to control an RL system's exploration of the state/action space. The problem of modulating the exploration of an unknown environment so that excessive cost is not incurred while ensure asymptotic optimality of the resulting controller is known variously as the exploration-exploitation dilemma, optimal learning, and the dual control problem. Theoretical formulations and solutions to this problem are known [32], but are computationally intractable. A variety of approximate methods, often based on statistical mechanisms, have been proposed (e.g. [61, 26, 14]), but have only been applied to small test problems. Lyapunov constraints can be interpreted as strong prior beliefs that certain control choices are suboptimal, and thus never explored, while the remainder are all equally good candidates – a rather binary view on the exploration problem.

7.6 Lyapunov Optimizing Feedback Control

The idea of Lyapunov optimizing feedback control is to choose a Lyapunov function, and then select a controller based on that Lyapunov function [79]. For example, steepest descent and quickest descent controllers are common choices. Such controllers are guaranteed to be stabilizing (i.e. converge to target), but take no explicit account of cost-to-go. LCRL follows the same pattern of specifying a Lyapunov function, and finding a good controller that descends on it. However, the RL system directly attempts to optimize cost-to-go.

8 General Plan of Work

The initial work of this thesis will explore the basic LCRL approach. A number of questions, having both theoretical and practical implications, need to be addressed. How can one use a Lyapunov function to constrain an RL controller? How should one formulate control choices for an RL controller? Of what value are non-Lyapunov heuristic functions?

Further work will explore extensions of the LCRL approach. In section 8.2 I describe some more complex combinations of Lyapunov and RL control that attempt to retain the theoretical benefits of Lyapunov methods without limiting the quality of control. Section 8.3 discusses the case in which multiple Lyapunov functions are known for a problem, which come up, for instance, when Lyapunov functions are parameterized. Section 8.4 describes a completely different use for Lyapunov functions in RL, aimed at accelerating RL for minimum-time-to-target tasks. This topic is independent of the others, and I intend to pursue it in parallel with the other work mentioned here.

8.1 Basic Framework

My first task will be to address several basic questions that arise when we attempt to integrate Lyapunov-based and RL control.

- **How to constrain RL?** I have presented two probabilistic switching methods that combine Lyapunov and RL control, but there are many other possibilities. Desirable properties of an LCRL regime include: 1) Provable convergence-to-target during and after learning, 2) Easy translation of a Lyapunov function/controller to some form of constraints, 3) Generality – the necessary conditions on the Lyapunov function/controller should be as weak as possible, and 4) Good empirical behavior.
- **Stochastic control problems.** So far, I have mainly discussed deterministic control problems. Lyapunov theory exists for stochastic control problems, and RL is well-suited to such problems. How do we combine the two in this case? What kinds of guarantees can we achieve?
- **How to formulate RL controls?** A common method used in RL for continuous control spaces is discretization. Is this sufficient? Should one include actions along the directions of steepest or quickest descent on the Lyapunov function? Should one include actions of different durations?
- **Heuristic control functions.** Can one use Lyapunov-like heuristic functions that cannot actually be decreased (by any action) in some parts of the state space? Vincent and Grantham [79] report that such functions can be quite beneficial in practice. How can such functions be used in an LCRL approach, and how well do they perform?

These questions have both theoretical and empirical components. I will perform experimental studies on small-scale, simulated dynamical control problems, testing different algorithms and the effects of their parameters. I would also like to test LCRL approaches on at least one larger-scale, more realistic control problem, to demonstrate that the combination of Lyapunov-based methods and RL can produce better control systems for problems that matter than either approach can alone.

8.2 Extensions for Achieving Optimality

The Lyapunov constraints imposed on an RL controller encourage good system performance right from the start of learning. Ultimately, however, the same constraints may bound the controller's performance away from optimality (or at least from some superior performance that RL could achieve). Several possible extensions to the LCRL approach seem to have the potential to remedy this problem:

- Use one of the two switching regimes described earlier, but reduce the probability of choosing the Lyapunov-based controller towards zero over time. On any particular trial, the finite probability will ensure convergence, while asymptotically the behavior of the controller approaches that of pure, unconstrained RL.

- Allow the controller to act unconstrained during some initial time window in each trial, but enforce the Lyapunov constraints thereafter. Assuming the initial window is chosen sufficiently wide to contain the trajectories of the optimal policy, it should be possible to obtain an optimal policy. At the same time, the convergence-to-target guarantee for individual trials is maintained because Lyapunov-constrained control will take over if the unconstrained RL controller does not reach the target within the specified time window.
- Replace conditions on the time derivative of the Lyapunov function with conditions on the expected value of this quantity. By choosing an appropriate probability distribution over the control set for a given state, it is possible to satisfy such a condition while allowing ascending actions with some probability. Combined with an off-policy learning method, such as Q-learning, this approach could allow convergence to an optimal policy if appropriate conditions are met.
- Divide time into intervals and require descent on the Lyapunov function for entire intervals rather than for each individual time step. During individual time steps, the controller is free to ascend on the Lyapunov function, as long as descent on the function is ensured for the overall interval. This gives the controller additional flexibility, which should allow it to improve performance beyond the limits of the basic framework.

8.3 Multiple Lyapunov Functions

In some cases, a controller designer may know more than one Lyapunov function for a given control problem. This is common, for instance, when a Lyapunov function is parameterized. If we want to take an LCRL approach, what do we do when we have a set of Lyapunov functions? In some sense, we want to know which is "best." We might use the following framework for making this determination automatically, at least for a finite, discrete set of candidate Lyapunov functions:

1. On each trial, a high-level controller chooses a Lyapunov function that will be used to constrain control choices for that trial.
2. A separate action-value function is learned for each Lyapunov function.
3. The value of each Lyapunov function for LCRL is estimated based on the costs incurred on trials when that Lyapunov function was active.

The high-level control problem, that of choosing a Lyapunov function to constrain the current trial based on a finite sample of outcomes for each, bears significant resemblance to an N-armed bandit problem. It is natural to apply methods for solving bandit problems to the problem of selecting good Lyapunov functions.

8.4 Dense Reward Structures

I also intend to investigate an alternative use for Lyapunov functions – that of generating more informative reward structures for RL applications. One example is minimum-time tasks.

Consider defining a system's rewards as the difference in the value of a given Lyapunov function before and after an action is taken. If the value of a trajectory to target is defined as the average reward per time step, then the optimal policy is the minimum time policy—because the total reward for any trajectory from a fixed starting point is the total drop in the Lyapunov function in getting to the target (which is invariant across policies), and so a minimal duration trajectory has the highest average reward per unit time.

The relative values of different policies are maintained, compared to the usual formulations of minimal time reward structures, but the absolute values of the policies are distorted. Lyapunov-based reward structures have the potential to speed learning because the controller is immediately rewarded for approaching the target and immediately penalized for moving away (where "approach" and "away" are defined by the Lyapunov function).

References

[1] Asada, H, Slotine, J-J (1986). *Robot analysis and control.* Wiley, New York.

[2] Atkeson, CG, Schaal, S (1997). Robot Learning From Demonstration. In DH Fisher, Jr. (ed.), *Machine Learning: Proceedings of the Fourteenth International Conference (ICML '97)*, 12–20, Morgan Kaufmann, San Francisco, CA.

[3] Baird, LC (1995). Residual algorithms: Reinforcement learning with function approximation. In *Machine Learning: Proceedings of the Twelfth International Conference*, Morgan Kaufmann, San Francisco, CA. (22 Nov 95 errata corrects errors in the published version).

[4] Baird, LC, Moore, AW (1999). Gradient descent for general reinforcement learning. In M Kearns, S Solla, D Cohn (eds.) *Advances in Neural Information Processing Systems 11*, pp. 968–974, MIT Press, Cambridge, MA.

[5] Barto, AG, Bradtke, SJ, Singh, SP (1995). Learning to act using real-time dynamic programming. *Artificial Intelligence, 72*, 81–138.

[6] Bertram, JE, Sarachik, PE (1959). Stability of circuits with randomly time-varying parameters. *Trans IRE, PGIT-5, Special Supplement,* 260.

[7] Bertsekas, DP (1987). *Dynamic programming: Deterministic and stochastic models.* Prentice-Hall, Englewood Cliffs, NJ.

[8] Bertsekas, DP, Tsitsiklis, JN (1989). *Parallel and distributed computation: Numerical methods.* Prentice-Hall, Englewood Cliffs, NJ.

[9] Bertsekas, DP, Tsitsiklis, JN (1996). *Neuro-dynamic programming.* Athena Scientific, Belmont, MA.

[10] Connell, J, Mahadevan, S (1993). *Robot learning.* Kluwer Academics, Boston, MA.

[11] Connolly, CI, Grupen, RA (1993). The applications of harmonic functions to robotics. *Journal of Robotic Systems, 10(7)*, 931–946.

[12] Clouse, J, Utgoff, P (1992). A teaching method for reinforcement learning. In *Proceedings of the Ninth International Conference on Machine Learning,* pp. 92–101. Aberdeen, Scotland.

[13] Crites, RH, Barto, AG (1996). Improving elevator performance using reinforcement learning. In DS Touretzky, MC Mozer, ME Hasselmo (eds.) *Advances in Neural Information Processing Systems 8*, pp. 1017–1023, MIT Press, Cambridge, MA.

[14] Dearden, R, Friedman, N, Russell, S (1998). Bayesian Q-learning. In *Proc. AAAI-98, Madison, Wisconsin.* AAAI Press.

[15] Dietterich, TG (1998). The MAXQ method for hierarchical reinforcement learning. In *Proceedings of the Fifteenth International Conference on Machine Learning* (ICML'98). Morgan Kaufmann, San Francisco, CA.

[16] Fürstenberg, H, Kesten, H (1960) Products of random matrices. *Annals of Mathematical Statistics, 31*, 457–469.

[17] Has'minskii, RZ (1967). Necessary and sufficient conditions for the asymptotic stability of linear stochastic systems. *Theory of Probability and its Applications, 12*, 144–147.

[18] Gordon, D, Subramanian, D (1994). A multistrategy learning scheme for agent knowledge acquisition. *Informatica, 17*, 331–346.

[19] Gullapalli, V, Barto, AG, Grupen RA (1994). Learning admittance mappings for force-guided assembly. In *Proceedings of the 1994 International Conference on Robotics and Automation,* 2633–2638.

[20] Hauskrecht, M, Meuleau, N, Boutilier, C, Kaelbling, LP, Dean, T (1998). Hierarchical solution of Markov decision processes using macro-actions. In *Proceedings of the Fourteenth International Conference on Uncertainty in Artificial Intelligence.* Morgan Kaufmann, San Francisco, CA.

[21] Howard, RA (1960). *Dynamic programming and Markov processes.* MIT Press, Cambridge, MA.

[22] Huber, M, Grupen, RA (1997). A feedback control structure for on-line learning tasks. *Robotics and Autonomous Systems, 22(3-4)*, 303–315.

[23] Huber M, Grupen, RA (1998). *Learning robot control–Using control policies as abstract actions.* NIPS'98 Workshop : Abstraction and Hierarchy in Reinforcement Learning, Breckenridge, CO.

[24] Huber M, Grupen, RA (1998). A Control Structure for Learning Locomotion Gaits. In *Proceedings of the Seventh International Symposium on Robotic and Applications.* TSI Press.

[25] Jacobson, DH, Mayne, DQ (1970). *Differential dynamic programming.* Elsevier, New York.

[26] Kaelbling, LP (1993). *Learning in embedded systems.* MIT Press, Cambridge, MA.

[27] Kalman, RE, Bertram, JE (1960). Control system analysis and design via the "second method" of Lyapunov. I Continuous-time systems. *Journal of Basic Engineering, 82*, 371–393.

[28] Kalman RE, Bertram, JE (1960). Control system analysis and design via the "second method" of Lyapunov. II Discrete-time systems. *Journal of Basic Engineering, 82*, 394–400.

[29] Khalil, HK, Teel, AR, Georgiou, TT, Praly, L, Sontag, E (1996). Stability. In Levine, WS (ed.), em The control handbook. CRC Press, Boca Raton, FL.

[30] Koenig, S, Simmons, RG (1993). Complexity analysis of real-time reinforcement learning. In *Proceedings of the National Conference on Artificial Intelligence*, 99–105.

[31] Koenig, S, Simmons, RG (1996). The effect of representation and knowledge on goal-directed exploration with reinforcement-learning algorithms. *Machine Learning*, 22, pp. 227–250.

[32] Kumar, PR (1985). A survey of some results in stochastic adaptive control. In *SIAM J. Control and Optimization*, vol. 23, No.3, pp. 329-380.

[33] Kushner, H (1971). *Introduction to stochastic control.* New York: Holt, Rinehart, and Winston.

[34] Lin, L (1992). Self-improving reactive agents based on reinforcement learning, planning, and teaching. *Machine Learning, 8*, 293–321.

[35] Lopara, KA, Feng, X (1996). Stability of stochastic systems. In WS Levine (ed.), *The control handbook.* CRC Press, Boca Raton, FL.

[36] Lyapunov, AM (1907). Problème général de la stabilité du mouvement. Ann. Fac. Sci. Toulouse, 9, 203–474. Reprinted in *Annals of Mathematical Study, 17, 1949.* Princeton University Press, Princeton, NJ.

[37] Maclin, R, Shavlik, JW (1994). Incorporating advice into agents that learn from reinforcements. In *Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI-94)*, 694-699, Seattle, WA.

[38] Maclin, R, Shavlik, JW (1996). Creating advice-taking reinforcement learners. *Machine Learning, 22,* pp. 251–281.

[39] McCallum, AK (1995). *Reinforcement learning with selective perception and hidden state.* PhD Thesis, Computer Science Department, University of Rochester.

[40] Moore, AW (1993). The Parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In JD Cowan, G Tesauro, J Alspector (eds.) *Advances in Neural Information Processing Systems 6*, pp. 711–718, Morgan Kaufmann, San Francisco.

[41] Moore, AW (1994). Variable resolution reinforcement learning. In *Proceedings of the Eighth Yale Workshop on Adaptive and Learning Systems.*

[42] Moore, AW, Atkeson CG (1993). Prioritized sweeping: Reinforcement learning with less data and less real time. *Machine Learning, 13*, 103–130.

[43] Moore, AW, Atkeson, CG (1995). The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. *Machine Learning, 21(3)*, 199–233.

[44] Munos, R, Moore, A (1999). *Variable resolution discretization for high-accuracy solutions of optimal control problems.* International Joint Conference on Artificial Intelligence (IJCAI99).

[45] Nakamura, Y (1991). Advanced robotics: Redundancy and optimization. Reading, MA: Addisson-Wesley.

[46] Neuneier, R (1997). Enhancing Q-learning for optimal asset allocation. In M Jordan, M Kearns, S Solla (eds.) *Advances in Neural Information Processing Systems 10*, pp. 936–942, MIT Press, Cambridge, MA.

[47] Neuneier, R, Mihatsch, O (1999). Risk sensitive reinforcement learning. In M Kearns, S Solla, D Cohn (eds.) *Advances in Neural Information Processing Systems 11*, pp. 1031–1037, MIT Press, Cambridge, MA.

[48] Norman, JM (1972). *Heuristic procedures in dynamic programming.* Manchester University Press, Manchester, UK.

[49] Parr, R (1998). *Hierarchical control and learning for Markov decision processes.* PhD thesis, University of California, Computer Science Division, Berkely, CA.

[50] Parr, R (1998). Reinforcement Learning with Hierarchies of Machines. In M Jordan, M Kearns, S Solla (eds.) *Advances in Neural Information Processing Systems 10*, pp. 1043–1049, MIT Press, Cambridge, MA.

[51] Peng, J, Williams, RJ (1993). Efficient learning and planning within the DYNA framework. *Adaptive Behavior, 1(4)*, 437–454.

[52] Precup, D, Sutton, RS (1997). Multi-time models for reinforcement learning. In *Proceedings of the ICML'97 Workshop on Modeling in Reinforcement Learning.*

[53] Precup, D, Sutton, RS (1998). Multi-time models for temporally abstract planning. In M Jordan, M Kearns, S Solla (eds.) *Advances in Neural Information Processing Systems 10*, pp. 1050–1056, MIT Press, Cambridge, MA.

[54] Precup, D, Sutton, RS, Singh, S (1998). Theoretical results on reinforcement learning with temporally abstract options. In *Proceedings of the Tenth European Conference on Machine Learning.*

[55] Rust, J (1996). Numerical dynamic programming in economics. In H Amman, D Kendrick, J Rust (eds.), *Handbook of Computational Economics,* 614–722. Elsevier, Amsterdam.

[56] Schaal, S (1997). Learning from demonstration. In MC Mozer, MI Jordan, T Petsche (eds.) *Advances in Neural Information Processing Systems 9*, pp. 1040–1046, MIT Press, Cambridge, MA.

[57] Schneider, JG (1997). Exploiting model uncertainty estimates for safe dynamic control learning. In MC Mozer, MI Jordan, T Petsche (eds.) *Advances in Neural Information Processing Systems 9*, pp. 1047–1053, MIT Press, Cambridge, MA.

[58] Schneider, JG, Boyan, JA, Moore, AW (1998). *Value function based production scheduling.* International Conference of Machine Learning, Madison, June/July 1998.

[59] Singh, SP, Barto, AG, Grupen, RA, Connolly, CI (1993). Robust reinforcement learning in motion planning. In JD Cowan, G Tesauro, J Alspector (eds.) *Advances in Neural Information Processing Systems 6*, pp. 655–662, Morgan Kaufmann, San Francisco.

[60] Singh, SP, Bertsekas, DP (1997). Reinforcement learning for dynamic channel allocation in cellular telephone systems. In MC Mozer, MI Jordan, T Petsche (eds.) *Advances in Neural Information Processing Systems 9*, pp. 970–980, MIT Press, Cambridge, MA.

[61] Sutton, RS (1990). Integrated architectures for learning, planning, and reacting based on approximate dynamic programming. In *Proceedings of the Seventh International Conference on Machine Learning*, 216–224. Morgan Kaufmann, San Mateo, CA.

[62] Sutton, RS (1991). Dyna, an integrated architecture for learning, planning, and reacting. *SIGART Bulletin, 2,* 160–163.

[63] Sutton, RS (1991). Planning by incremental dynamic programming. In LA Birnbaum, GC Collins (eds.), *Proceedings of the Eighth International Workshop on Machine Learning,* 353–357. Morgan Kaufmann, San Mateo, CA.

[64] Sutton, RS (1996). Generalization in reinforcement learning: Successful examples using sparse coarse coding. In DS Touretzky, MC Mozer, ME Hasselmo (eds.) *Advances in Neural Information Processing Systems 8*, pp. 1038–1044, MIT Press, Cambridge, MA.

[65] Sutton, RS, Barto, AG (1998). *Reinforcement learning: An introduction.* MIT Press, Cambridge, MA.

[66] Sutton, RS, McAllester, D, Singh, S, Mansour, Y (accepted). Policy gradient methods for reinforcement with function approximation. In *Advances in Neural Information Processing Systems 12.*

[67] Sutton, RS, Precup, D, Singh, S (1998). Intra-option learning about temporally abstract actions. In *Proceedings of the 15th International Conference on Machine Learning.*

[68] Sutton, RS, Precup, D, Singh, S (1998). *Between MDPs and semi-MDPs: Learning, planning, and representing knowledge at multiple temporal scales.* Technical Report UM-CS-1998-074, Department of Computer Science, University of Massachusetts, Amherst, MA.

[69] Sutton, RS, Precup, D, Singh, S, Ravindran, B (1998). Improved switching among temporally abstract actions. In M Kearns, S Solla, D Cohn (eds.) *Advances in Neural Information Processing Systems 11*, pp. 1066–1072, MIT Press, Cambridge, MA.

[70] Thrun, S, Mitchell, T (1993). Integrating inductive neural network learning and explanation-based learning. In *Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence,* 930–936.

[71] Tesauro, GJ (1992). TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Computation, 6(2)*, 215–219.

[72] Tsitsiklis, JN, Van Roy, B (1996). Feature-based methods for large-scale dynamic programming. *Machine Learning, 22*, 59–94.

[73] Tsitsiklis, JN, Van Roy, B (1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control, AC-42,* 674–690.

[74] Tsitsiklis, JN, Van Roy, B (1997). Analysis of temporal-difference learning with function approximation. In MC Mozer, MI Jordan, T Petsche (eds.) *Advances in Neural Information Processing Systems 9*, pp. 1075–1081, MIT Press, Cambridge, MA.

[75] Tsitsiklis, JN, Van Roy, B (1997). Approximate solutions to optimal stopping problems. In MC Mozer, MI Jordan, T Petsche (eds.) *Advances in Neural Information Processing Systems 9*, pp. 1082–1088, MIT Press, Cambridge, MA.

[76] Tsitsiklis JN, Van Roy, B (in press). Optimal stopping of Markov processes: Hilbert space theory, approximation algorithms, and an application to pricing high-dimensional financial derivatives. To appear in *IEEE Transactions on Automatic Control.*

[77] Utgoff, P, Clouse, J (1991). Two kinds of training information for evaluation function learning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 596–600. Anaheim, CA.

[78] Van Roy, B (1998). *Learning and value function approximation in complex decision processes.* PhD Thesis, Massachusetts Institute of Technology.

[79] Vincent, TL, Grantham, WJ (1997). *Nonlinear and optimal control systems.* Wiley, New York.

[80] Watkins, CJCJ (1989). *Learning from delayed rewards.* Ph.D. thesis. Cambridge University, Cambridge, UK.

[81] Werbos, PJ (1992). Approximate dynamic programming for real-time control and neural modeling. In DA White, DA Sofge (eds.), *Handbook of intelligent control: Neural, Fuzzy, and Adaptive Approaches,* 493–525. Van Nostrand Reinhold, New York.

[82] Whitehead, S (1991). A complexity analysis of cooperative mechanisms in reinforcement learning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 607–613. Anaheim, CA.

[83] Williams, RJ, Baird, LC (1993). *Tight performance bounds on greedy policies based on imperfect value functions.* Technical Report NU-CCS-93-14. Northeastern University.

[84] Zhang, W, Dietterich, TG (1996). High-performance job-shop scheduling with a time-delay TD(λ) network. In DS Touretzky, MC Mozer, ME Hasselmo (eds.) *Advances in Neural Information Processing Systems 8*, pp. 1024–1030, MIT Press, Cambridge, MA.