Linear Complementarity for Regularized Policy Evaluation and Improvement

Jeff Johns **Christopher Painter-Wakefield** **Ronald Parr**
Department of Computer Science
Duke University
Durham, NC 27708
{johns, paint007, parr}@cs.duke.edu

Abstract

Recent work in reinforcement learning has emphasized the power of L_1 regularization to perform feature selection and prevent overfitting. We propose formulating the L_1 regularized linear fixed point problem as a linear complementarity problem (LCP). This formulation offers several advantages over the LARS-inspired formulation, LARS-TD. The LCP formulation allows the use of efficient off-the-shelf solvers, leads to a new uniqueness result, and can be initialized with starting points from similar problems (warm starts). We demonstrate that warm starts, as well as the efficiency of LCP solvers, can speed up policy iteration. Moreover, warm starts permit a form of modified policy iteration that can be used to approximate a "greedy" homotopy path, a generalization of the LARS-TD homotopy path that combines policy evaluation and optimization.

1 Introduction

L_1 regularization has become an important tool over the last decade with a wide variety of machine learning applications. In the context of linear regression, its use helps prevent overfitting and enforces sparsity in the problem's solution. Recent work has demonstrated how L_1 regularization can be applied to the value function approximation problem in Markov decision processes (MDPs). Kolter and Ng [1] included L_1 regularization within the least-squares temporal difference learning [2] algorithm as LARS-TD, while Petrik et al. [3] adapted an approximate linear programming algorithm. In both cases, L_1 regularization automates the important task of selecting relevant features, thereby easing the design choices made by a practitioner.

LARS-TD provides a homotopy method for finding the L_1 regularized linear fixed point formulated by Kolter and Ng. We reformulate the L_1 regularized linear fixed point as a linear complementarity problem (LCP). This formulation offers several advantages. It allows us to draw upon the rich theory of LCPs and optimized solvers to provide strong theoretical guarantees and fast performance. In addition, we can take advantage of the "warm start" capability of LCP solvers to produce algorithms that are better suited to the sequential nature of policy improvement than LARS-TD, which must start from scratch for each new policy.

2 Background

First, we introduce MDPs and linear value function approximation. We then review L_1 regularization and feature selection for regression problems. Finally, we introduce LCPs. We defer discussion of L_1 regularization and feature selection for reinforcement learning (RL) until section 3.

2.1 MDP and Value Function Approximation Framework

We aim to discover optimal, or near-optimal, policies for Markov decision processes (MDPs) defined by the quintuple $M = (S, A, P, R, \gamma)$. Given a state $s \in S$, the probability of a transition to a state $s' \in S$ when action $a \in A$ is taken is given by $P(s'|s, a)$. The reward function is a mapping from states to real numbers $R : S \mapsto \mathbb{R}$. A policy π for M is a mapping from states to actions $\pi : s \mapsto a$ and the transition matrix induced by π is denoted P^π. Future rewards are discounted by $\gamma \in [0, 1)$.

The value function at state s for policy π is the expected total γ-discounted reward for following π from s. In matrix-vector form, this is written:

$$V^\pi = T^\pi V^\pi = R + \gamma P^\pi V^\pi,$$

where T^π is the *Bellman operator* for policy π and V^π is the fixed point of this operator. An optimal policy, π^*, maximizes state values, has value function V^*, and is the fixed point of the T^* operator:

$$T^*V(s) = R(s) + \gamma \max_{a \in A} \sum_{s' \in S} P(s'|s, a)V(s').$$

Of the many algorithms that exist for finding π^*, *policy iteration* is most relevant to the presentation herein. For any policy π_j, policy iteration computes V^{π_j}, then determines π_{j+1} as the "greedy" policy with respect to V^{π_j}:

$$\pi_{j+1}(s) = \arg\max_{a \in A} [R(s) + \gamma \sum_{s' \in S} P(s'|s, a)V^{\pi_j}(s')].$$

This is repeated until some convergence condition is met. For an exact representation of each V^{π_j}, the algorithm will converge to an optimal policy and the unique, optimal value function V^*.

The value function, transition model, and reward function are often too large to permit an exact representation. In such cases, an approximation architecture is used for the value function. A common choice is $\hat{V} = \Phi w$, where w is a vector of k scalar weights and Φ stores a set of k features in an $n \times k$ matrix with one row per state. Since n is often intractably large, Φ can be thought of as populated by k linearly independent *basis functions*, $\varphi_1 \ldots \varphi_k$, implicitly defining the columns of Φ.

For the purposes of estimating w, it is common to replace Φ with $\hat{\Phi}$, which samples rows of Φ, though for conciseness of presentation we will use Φ for both, since algorithms for estimating w are essentially identical if $\hat{\Phi}$ is substituted for Φ. Typical linear function approximation algorithms [2] solve for the w which is a fixed point:

$$\Phi w = \Pi(R + \gamma \Phi'^\pi w) = \Pi T^\pi \Phi w,$$

where Π is the L_2 projection into the span of Φ and Φ'^π is $P^\pi\Phi$ in the explicit case and composed of sampled next features in the sampled case. Likewise, we overload T^π for the sampled case.

2.2 L_1 Regularization and Feature Selection in Regression

In regression, the L_1 regularized least squares problem is defined as:

$$w = \arg\min_{x \in \mathbb{R}^k} \frac{1}{2}\|\Phi x - y\|_2^2 + \beta\|x\|_1, \tag{1}$$

where $y \in \mathbb{R}^n$ is the target function and $\beta \in \mathbb{R}_{\geq 0}$ is a regularization parameter. This penalized regression problem is equivalent to the *Lasso* [4], which minimizes the squared residual subject to a constraint on $\|x\|_1$. The use of the L_1 norm in the objective function prevents overfitting, but also serves a secondary purpose of promoting sparse solutions (i.e., coefficients w containing many 0s). Therefore, we can think of L_1 regularization as performing *feature selection*. The Lasso's objective function is convex, ensuring the existence of a global (though not necessarily unique) minimum.

Even though the optimal solution to the Lasso can be computed in a fairly straightforward manner using convex programming, this approach is not very efficient for large problems. This is a motivating factor for the least angle regression (LARS) algorithm [5], which can be thought of as a homotopy method for solving the Lasso for *all* nonnegative values of β. We do not repeat the details of the algorithm here, but point out that this is easier than it might sound at first because the homotopy path in β-space is *piecewise linear* (with finitely many segments). Furthermore, there exists a closed form solution for moving from one piecewise linear segment to the next segment. An important benefit of LARS is that it provides solutions for all values of β in a single run of the algorithm. Cross-validation can then be performed to select an appropriate value.

2.3 LCP and BLCP

Given a square matrix M and a vector q, a linear complementarity problem (LCP) seeks vectors $w \geq \mathbf{0}$ and $z \geq \mathbf{0}$ with $w^T z = 0$ and

$$w = q + Mz.$$

The problem is thus parameterized by LCP(q, M). Even though LCPs may appear to be simple feasibility problems, the framework is rich enough to express any convex quadratic program.

The *bounded* linear complementarity problem (BLCP) [6] includes box constraints on z. The BLCP computes w and z where $w = q + Mz$ and each variable z_i meets one of the following conditions:

$$z_i = u_i \quad \Longrightarrow \quad w_i \leq 0 \tag{2a}$$
$$z_i = l_i \quad \Longrightarrow \quad w_i \geq 0 \tag{2b}$$
$$l_i < z_i < u_i \quad \Longrightarrow \quad w_i = 0 \tag{2c}$$

with bounds $-\infty \leq l_i < u_i \leq \infty$. The parameterization is written BLCP(q, M, l, u). Notice that an LCP is a special case of a BLCP with $l_i = 0$ and $u_i = \infty, \forall i$. Like the LCP, the BLCP has a unique solution when M is a P-matrix[1] and there exist algorithms which are guaranteed to find this solution [6, 7]. When the lower and upper bounds on the BLCP are finite, the BLCP can in fact be formulated as an equivalent LCP of twice the dimensionality of the original problem. A full derivation of this equivalence is shown in the appendix (supplementary materials).

There are many algorithms for solving (B)LCPs. Since our approach is not tied to a particular algorithm, we review some general properties of (B)LCP solvers. Optimized solvers can take advantage of sparsity in z. A zero entry in z effectively cancels out a column in M. If M is large, efficient solvers can avoid using M directly, instead using a smaller M' that is induced by the nonzero entries of z. The columns of M' can be thought of as the "active" columns and the procedure of swapping columns in and out of M' can be thought of as a pivoting operation, analogous to pivots in the simplex algorithm. Another important property of some (B)LCP algorithms is their ability to start from an initial guess at the solution (i.e., a "warm start"). If the initial guess is close to a solution, this can significantly reduce the solver's runtime.

Recently, Kim and Park [8] derived a connection between the BLCP and the Karush-Kuhn-Tucker (KKT) conditions for LARS. In particular, they noted the solution to the minimization problem in equation (1) has the form:

$$\underbrace{x}_{w} = \underbrace{(\Phi^T\Phi)^{-1}\Phi^T y}_{q} + \underbrace{(\Phi^T\Phi)^{-1}}_{M}\underbrace{(-c)}_{z},$$

where the vector $-c$ follows the constraints in equation (2) with $l_i = -\beta$ and $u_i = \beta$. Although we describe the equivalence between the BLCP and LARS optimality conditions using $M \equiv (\Phi^T\Phi)^{-1}$, the inverse can take place inside the BLCP algorithm and this operation is feasible and efficient as it is only done for the active columns of Φ. Kim and Park [8] used a block pivoting algorithm, originally introduced by Júdice and Pires [6], for solving the Lasso. Their experiments show the block pivoting algorithm is significantly faster than both LARS and Feature Sign Search [9].

3 Previous Work

Recent work has emphasized feature selection as an important problem in reinforcement learning [10, 11]. Farahmand et al. [12] consider L_2 regularized RL. An L_1 regularized Bellman residual minimization algorithm was proposed by Loth et al. [13][2]. Johns and Mahadevan [14] investigate the combination of least squares temporal difference learning (LSTD) [2] with different variants of the matching pursuit algorithm [15, 16]. Petrik et al. [3] consider L_1 regularization in the context of approximate linear programming. Their approach offers some strong guarantees, but is not well-suited to noisy, sampled data.

[1] A P-matrix is a matrix for which all principal minors are positive.

[2] Loth et al. claim to adapt LSTD to L_1 regularization, but in fact describe a Bellman residual minimization algorithm and not a fixed point calculation.

The work most directly related to our own is that of Kolter and Ng [1]. They propose augmenting the LSTD algorithm with an L_1 regularization penalty. This results in the following L_1 *regularized linear fixed point* (L_1TD) problem:

$$w = \arg\min_{x \in \mathbb{R}^k} \frac{1}{2} \|\Phi x - (R + \gamma \Phi'^{\pi} w)\|_2^2 + \beta \|x\|_1. \tag{3}$$

Kolter and Ng derive a set of necessary and sufficient conditions characterizing the above fixed point[3] in terms of β, w, and a vector c of correlations between the features and the Bellman residual $T^\pi \hat{V} - \hat{V}$. More specifically, the correlation c_i associated with feature φ_i is given by:

$$c_i = \varphi_i^T (T^\pi \hat{V} - \hat{V}) = \varphi_i^T (R + \gamma \Phi'^{\pi} w - \Phi w). \tag{4}$$

Introducing the notation $\mathcal{I}$ to denote the set of indices of *active* features in the model (i.e., $\mathcal{I} = \{i : w_i \neq 0\}$), the fixed point optimality conditions can be summarized as follows:

C_1. All features in the active set share the same absolute correlation, β: $\forall i \in \mathcal{I}, |c_i| = \beta$.

C_2. Inactive features have less absolute correlation than active features: $\forall i \notin \mathcal{I}, |c_i| < \beta$.

C_3. Active features have correlations and weights agreeing in sign: $\forall i \in \mathcal{I}, \mathrm{sgn}(c_i) = \mathrm{sgn}(w_i)$.

Kolter and Ng show that it is possible to find the fixed point using an iterative procedure adapted from LARS. Their algorithm, LARS-TD, computes a sequence of fixed points, each of which satisfies the optimality conditions above for some intermediate L_1 parameter $\bar{\beta} \geq \beta$. Successive solutions decrease $\bar{\beta}$ and are computed in closed form by determining the point at which a feature must be added or removed in order to further decrease $\bar{\beta}$ without violating one of the fixed point requirements. The algorithm (as applied to action-value function approximation) is a special case of the algorithm presented in the appendix (see Fig. 2). Kolter and Ng prove that if $\Phi^T(\Phi - \gamma \Phi'^{\pi})$ is a P-matrix, then for any $\beta \geq 0$, LARS-TD will find a solution to equation (3).

LARS-TD inherits many of the benefits and limitations of LARS. The fact that it traces an entire homotopy path can be quite helpful because it does not require committing to a particular value of β. On the other hand, the incremental nature of LARS may not be the most efficient solution for any single value of the regularization parameter, as shown by Lee et al. [9] and Kim and Park [8].

It is natural to employ LARS-TD in an iterative manner within the least squares policy iteration (LSPI) algorithm [17], as Kolter and Ng did. In this usage, however, many of the benefits of LARS are lost. When a new policy is selected in the policy iteration loop, LARS-TD must discard its solution from the previous policy and start an entirely new homotopy path, making the value of the homotopy path in this context not entirely clear. One might cross-validate a choice of regularization parameter by measuring the performance of the final policy, but this requires guessing a value of β for all policies and then running LARS-TD up to this value for each policy. If a new value of β is tried, all of the work done for the previous value must be discarded.

4 The L_1 Regularized Fixed Point as an LCP

We show that the optimality conditions for the L_1TD fixed point correspond to the solution of a (B)LCP. This reformulation allows for (1) new algorithms to compute the fixed point using (B)LCP solvers, and (2) a new guarantee on the uniqueness of a fixed point.

The L_1 regularized linear fixed point is described by a vector of correlations c as defined in equation (4). We introduce the following variables:

$$A = \Phi^T(\Phi - \gamma \Phi'^{\pi}) \qquad\qquad b = \Phi^T R,$$

[3] For fixed w, the RHS of equation (3) is a convex optimization problem; a sufficient condition for optimality of some vector x^* is that the zero vector is in the subdifferential of the RHS at x^*. The fixed point conditions follow from the equality between the LHS and RHS.

that allow equation (4) to be simplified as $c = b - Aw$. Assuming A is a P-matrix, A is invertible[4] [18] and we can write:

$$\underbrace{w}_{w} = \underbrace{A^{-1}b}_{q} + \underbrace{A^{-1}}_{M}\underbrace{(-c)}_{z}.$$

Consider a solution (w and z) to the equation above where z is bounded as in equation (2) with $l = -\boldsymbol{\beta}$ and $u = \boldsymbol{\beta}$ to specify a BLCP. It is easy to verify that coefficients w satisfying this BLCP acheive the L_1TD optimality conditions as detailed in section 3. Thus, any appropriate solver for the BLCP($A^{-1}b, A^{-1}, -\boldsymbol{\beta}, \boldsymbol{\beta}$) can be thought of as a linear complementarity approach to solving for the L_1TD fixed point. We refer to this class of solvers as *LC-TD algorithms* and parameterize them as LC-TD($\Phi, \Phi'^{\pi}, R, \gamma, \beta$).

Proposition 1 *If A is a P-matrix, then for any R, the L_1 regularized linear fixed point exists, is unique, and will be found by a basic-set BLCP algorithm solving BLCP($A^{-1}b, A^{-1}, -\boldsymbol{\beta}, \boldsymbol{\beta}$).*

This proposition follows immediately from some basic BLCP results. We note that if A is a P-matrix, so is A^{-1} [18], that BLCPs for P-matrices have a unique solution for any q ([7], Chp. 3), and that the the basic-set algorithm of Júdice and Pires [19] is guaranteed to find a solution to any BLCP with a P-matrix. This strengthens the theorem by Kolter and Ng [1], which guaranteed only that the LARS-TD algorithm would converge to *a* solution when A is a P-matrix.

This connection to the LCP literature has practical benefits as well as theoretical ones. Decoupling the problem from the solver allows a variety of algorithms to be exploited. For example, the ability of many solvers to use a warm start during initialization offers a significant computational advantage over LARS-TD (which always begins with a null solution). In the experimental section of this paper, we demonstrate that the ability to use warm starts during policy iteration can significantly improve computational efficiency. We also find that (B)LCP solvers can be more robust than LARS-TD, an issue we address further in the appendix.

5 Modified Policy Iteration using LARS-TD and LC-TD

As mentioned in section 3, the advantages of LARS-TD as a homotopy method are less clear when it is used in a policy iteration loop since the homotopy path is traced only for specific policies. It is possible to incorporate greedy policy improvements into the LARS-TD loop, leading to a homotopy path for greedy policies. The greedy L_1 regularized fixed point equation is:

$$w = \arg\min_{x \in \mathbb{R}^k} \frac{1}{2}\|\Phi x - \max_{\pi}(R + \gamma \Phi'^{\pi} w)\|_2^2 + \beta\|x\|_1. \tag{5}$$

We propose a modification to LARS-TD called LARQ which, along with conditions C_1-C_3 in section 3, maintains an additional invariant:

C_4. The current policy π is greedy with respect to the current solution.

It turns out that we can change policies and avoid violating the LARS-TD invariants if we make policy changes at points where applying the Bellman operator yields the same value for both the old policy (π) and the new policy (π'): $T^{\pi}\hat{V} = T^{\pi'}\hat{V}$. The LARS-TD invariants all depend on the correlation of features with the residual $T^{\pi}\hat{V} - \hat{V}$ of the current solution. When the above equation is satisfied, the residual is equal for both policies. Thus, we can change policies at such points without violating any of the LARS-TD invariants. Due to space limitations, we defer a full presentation of the LARQ algorithm to the appendix.

When run to completion, LARQ provides a set of action-values that are the greedy fixed point for *all* settings of β. In principle, this is more flexible than LARS-TD with policy iteration because it produces these results in a single run of the algorithm. In practice, LARQ suffers two limitations.

[4]Even when A is not invertible, we can still use a BLCP solver as long as the principal submatrix of A associated with the active features is invertible. As with LARS-TD, the inverse only occurs for this principal submatrix. In fact, we discuss in the appendix how one need never explicitly compute A. Alternatively, we can convert the BLCP to an LCP (appendix A.1) thereby avoiding A^{-1} in the parameterization of the problem.

The first is that it can be slow. LARS-TD enumerates every point at which the active set of features might change, a calculation that must be redone every time the active set changes. LARQ must do this as well, but it must also enumerate all points at which the greedy policy can change. For k features and n samples, LARS-TD must check $O(k)$ points, but LARQ must check $O(k+n)$ points. Even though LARS-TD will run multiple times within a policy iteration loop, the number of such iterations will typically be far fewer than the number of training data points. In practice, we have observed that LARQ runs several times slower than LARS-TD with policy iteration.

A second limitation of LARQ is that it can get "stuck." This occurs when the greedy policy for a particular β is not well defined. In such cases, the algorithm attempts to switch to a new policy immediately following a policy change. This problem is not unique to LARQ. Looping is possible with most approximate policy iteration algorithms. What makes it particularly troublesome for LARQ is that there are few satisfying ways of addressing this issue without sacrificing the invariants.

To address these limitations, we present a compromise between LARQ and LARS-TD with policy iteration. The algorithm, LC-MPI, is presented as Algorithm 1. It avoids the cost of continually checking for policy changes by updating the policy only at a fixed set of values, $\beta^{(1)} \dots \beta^{(m)}$. Note that the β values are in decreasing order with $\beta^{(1)}$ set to the maximum value (i.e., the point such that $w^{(1)}$ is the zero vector). At each $\beta^{(j)}$, the algorithm uses a policy iteration loop to (1) determine the current policy (greedy with respect to parameters $\hat{w}^{(j)}$), and (2) compute an approximate value function $\Phi w^{(j)}$ using LC-TD. The policy iteration loop terminates when $w^{(j)} \approx \hat{w}^{(j)}$ or some predefined number of iterations is exceeded. This use of LC-TD within a policy iteration loop will typically be quite fast because we can use the current feature set as a warm start. The warm start is indicated in Algorithm 1 by $supp(\hat{w}^{(j)})$, where the function $supp$ determines the support, or active elements, in $\hat{w}^{(j)}$; many (B)LCP solvers can use this information for initialization.

Once the policy iteration loop terminates for point $\beta^{(j)}$, LC-MPI simply begins at the next point $\beta^{(j+1)}$ by initializing the weights with the previous solution, $\hat{w}^{(j+1)} \leftarrow w^{(j)}$. This was found to be a very effective technique. As an alternative, we tested initializing $\hat{w}^{(j+1)}$ with the result of running LARS-TD with the greedy policy implicit in $w^{(j)}$ from the point $(\beta^{(j)}, w^{(j)})$ to $\beta^{(j+1)}$. This initialization method performed worse experimentally than the simple approach described above.

We can view LC-MPI as approximating LARQ's homotopy path since the two algorithms agree for any $\beta^{(j)}$ reachable by LARQ. However, LC-MPI is more efficient and avoids the problem of getting stuck. By compromising between the greedy updates of LARQ and the pure policy evaluation methods of LARS-TD and LC-TD, LC-MPI can be thought of as form of modified policy iteration [20]. The following table summarizes the properties of the algorithms described in this paper.

	LARS-TD Policy Iteration	LC-TD Policy Iteration	LARQ	LC-MPI
Warm start for each new β	N	N	Y	Y
Warm start for each new policy	N	Y	Y	Y
Greedy policy homotopy path	N	N	Y	Approximate
Robust to policy cycles	Y	Y	N	Y

6 Experiments

We performed two types of experiments to highlight the potential benefits of (B)LCP algorithms. First, we used both LARS-TD and LC-TD within policy iteration. These experiments, which were run using a single value of the L_1 regularization parameter, show the benefit of warm starts for LC-TD. The second set of experiments demonstrates the benefit of using the LC-MPI algorithm. A single run of LC-MPI results in greedy policies for *multiple* values of β, allowing the use of cross-validation to pick the best policy. We show this is significantly more efficient than running policy iteration with either LARS-TD or LC-TD multiple times for different values of β. We discuss the details of the specific LCP solver we used in the appendix.

Both types of experiments were conducted on the 20-state chain [17] and mountain car [21] domains, the same problems tested by Kolter and Ng [1]. The chain MDP consists of two stochastic actions, left and right, a reward of one at each end of the chain, and $\gamma = 0.9$. One thousand samples were generated using 100 episodes, each consisting of 10 random steps. For features, we used 1000 Gaussian random noise features along with five equally spaced radial basis functions (RBFs) and a constant function. The goal in the mountain car MDP is to drive an underpowered car up a hill

Algorithm 1 LC-MPI

Inputs:

$\{s_i, a_i, r_i, s'_i\}_{i=1}^n$, state transition and reward samples

$\varphi : S \times A \to \mathbb{R}^k$, state-action features

$\gamma \in [0, 1)$, discount factor

$\{\beta^{(j)}\}_{j=1}^m$, where $\beta^{(1)} = \max_l \left|\sum_{i=1}^n \varphi_l(s_i, a_i) r_i\right|$, $\beta^{(j)} < \beta^{(j-1)}$ for $j \in \{2, \ldots, m\}$, and $\beta^{(m)} \geq 0$

$\epsilon \in \mathbb{R}_+$ and $T \in \mathbb{N}$, termination conditions for policy iteration

Initialization:

$\Phi \leftarrow [\varphi(s_1, a_1) \ \ldots \ \varphi(s_n, a_n)]^T$, $R \leftarrow [r_1 \ \ldots \ r_n]^T$, $w^{(1)} \leftarrow \mathbf{0}$

for $j = 2$ to m **do**

// Initialize with the previous solution

$\hat{w}^{(j)} \leftarrow w^{(j-1)}$

// Policy iteration loop

Loop:

// Select greedy actions and form Φ'

$\forall i : a'_i \leftarrow \arg\max_a \varphi(s'_i, a)^T \hat{w}^{(i)}$

$\Phi' \leftarrow [\varphi(s'_1, a'_1) \ \ldots \ \varphi(s'_n, a'_n)]^T$

// Solve the LC-TD problem using a (B)LCP solver with a warm start

$w^{(j)} \leftarrow$ LC-TD$(\Phi, \Phi', R, \gamma, \beta^{(j)})$ with warm start $supp(\hat{w}^{(j)})$

// Check for termination

if $(\|w^{(j)} - \hat{w}^{(j)}\|_2 \leq \epsilon)$ or (# iterations $\geq T$)

then break loop

else $\hat{w}^{(j)} \leftarrow w^{(j)}$

Return $\{w^{(j)}\}_{j=1}^m$

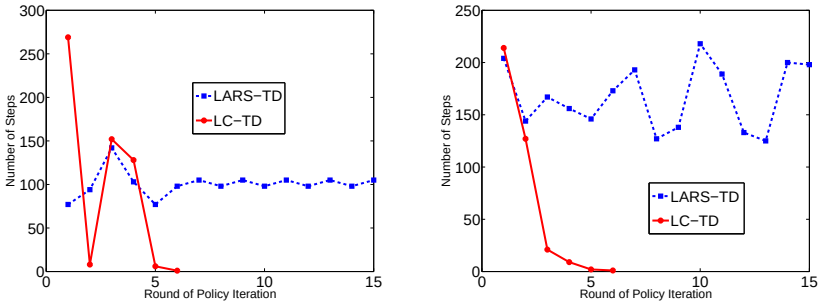
by building up momentum. The domain is continuous, two dimensional, and has three actions. We used $\gamma = 0.99$ and 155 radial basis functions (apportioned as a two dimensional grid of 1, 2, 3, 4, 5, 6, and 8 RBFs) and one constant function for features. Samples were generated using 75 episodes where each episode started in a random start state, took random actions, and lasted at most 20 steps.

6.1 Policy Iteration

To compare LARS-TD and LC-TD when employed within policy iteration, we recorded the *number of steps* used during each round of policy iteration, where a *step* corresponds to a change in the active feature set. The computational complexity per step of each algorithm is similar; therefore, we used the average number of steps per policy as a metric for comparing the algorithms. Policy iteration was run either until the solution converged or 15 rounds were exceeded. This process was repeated 10 times for 11 different values of β. We present the results from these experiments in the first two columns of Table 1. The two algorithms performed similarly for the chain MDP, but LC-TD used significantly fewer steps for the mountain car MDP. Figure 1 shows plots for the number of steps used for each round of policy iteration for a single (typical) trial. Notice the declining trend for LC-TD; this is due to the warm starts requiring fewer steps to find a solution. The plot for the chain MDP shows that LC-TD uses many more steps in the first round of policy iteration than does LARS-TD. Lastly, in the trials shown in Figure 1, policy iteration using LC-TD converged in six iterations whereas it did not converge at all when using LARS-TD. This was due to LARS-TD producing solutions that violate the L_1TD optimality conditions. We discuss this in detail in appendix A.5.

6.2 LC-MPI

When LARS-TD and LC-TD are used as subroutines within policy iteration, the process ends at a single value of the L_1 regularization parameter β. The policy iteration loop must be rerun to consider different values of β. In this section, we show how much computation can be saved by running LC-MPI once (to produce m greedy policies, each at a different value of β) versus running policy iteration m separate times. The third column in Table 1 shows the average number of algorithm steps per policy for LC-MPI. As expected, there is a significant reduction in complexity by using LC-MPI for both domains. In the appendix, we give a more detailed example of how cross-validation can be



(a) Chain (b) Mountain car

Figure 1: Number of steps used by algorithms LARS-TD and LC-TD during each round of policy iteration for a typical trial. For LC-TD, note the decrease in steps due to warm starts.

Domain	LARS-TD, PI	LC-TD, PI	LC-MPI
Chain	73 ± 13	77 ± 11	24 ± 11
Mountain car	214 ± 33	116 ± 22	21 ± 5

Table 1: Average number of algorithm steps per policy.

used to select a good value of the regularization parameter. We also offer some additional comments on the robustness of the LARS-TD algorithm.

7 Conclusions

In this paper, we proposed formulating the L_1 regularized linear fixed point problem as a linear complementarity problem. We showed the LCP formulation leads to a stronger theoretical guarantee in terms of the solution's uniqueness than was previously shown. Furthermore, we demonstrated that the "warm start" ability of LCP solvers can accelerate the computation of the L_1TD fixed point when initialized with the support set of a related problem. This was found to be particularly effective for policy iteration problems when the set of active features does not change significantly from one policy to the next.

We proposed the LARQ algorithm as an alternative to LARS-TD. The difference between these algorithms is that LARQ incorporates greedy policy improvements inside the homotopy path. The advantage of this "greedy" homotopy path is that it provides a set of action-values that are a greedy fixed point for all settings of the L_1 regularization parameter. However, this additional flexibility comes with increased computational complexity. As a compromise between LARS-TD and LARQ, we proposed the LC-MPI algorithm which only maintains the LARQ invariants at a fixed set of values. The key to making LC-MPI efficient is the use of warm starts by using an LCP algorithm.

There are several directions for future work. An interesting question is whether there is a natural way to incorporate policy improvement directly within the LCP formulation. Another concern for L_1TD algorithms is a better characterization of the conditions under which solutions exist and can be found efficiently. In previous work, Kolter and Ng [1] indicated the P-matrix property can always hold provided enough L_2 regularization is added to the problem. While this is possible, it also decreases the sparsity of the solution; therefore, it would be useful to find other techniques for guaranteeing convergence while maintaining sparsity.

Acknowledgments

This work was supported by the National Science Foundation (NSF) under Grant #0937060 to the Computing Research Association for the CIFellows Project, NSF Grant IIS-0713435, and DARPA CSSG HR0011-06-1-0027. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation or the Computing Research Association.

References

[1] J. Kolter and A. Ng. Regularization and feature selection in least-squares temporal difference learning. In *Proc. ICML*, pages 521–528, 2009.

[2] S. Bradtke and A. Barto. Linear least-squares algorithms for temporal difference learning. *Machine Learning*, 22(1-3):33–57, 1996.

[3] M. Petrik, G. Taylor, R. Parr, and S. Zilberstein. Feature selection using regularization in approximate linear programs for Markov decision processes. In *To appear in Proc. ICML*, 2010.

[4] R. Tibshirani. Regression shrinkage and selection via the Lasso. *Journal of the Royal Statistical Society. Series B (Methodological)*, 58(1):267–288, 1996.

[5] B. Efron, T. Hastie, I. Johnstone, and R. Tibshirani. Least angle regression. *The Annals of Statistics*, 32(2):407–451, 2004.

[6] J. Júdice and F. Pires. A block principal pivoting algorithm for large-scale strictly monotone linear complementarity problems. *Computers and Operations Research*, 21(5):587–596, 1994.

[7] K. Murty. *Linear Complementarity, Linear and Nonlinear Programming*. Heldermann Verlag, 1988.

[8] J. Kim and H. Park. Fast active-set-type algorithms for L_1-regularized linear regression. In *Proc. AISTAT*, pages 397–404, 2010.

[9] H. Lee, A. Battle, R. Raina, and A. Ng. Efficient sparse coding algorithms. In *Advances in Neural Information Processing Systems 19*, pages 801–808, 2007.

[10] S. Mahadevan and M. Maggioni. Proto-value functions: A Laplacian framework for learning representation and control in Markov decision processes. *JMLR*, 8:2169–2231, 2007.

[11] R. Parr, L. Li, G. Taylor, C. Painter-Wakefield, and M. Littman. An analysis of linear models, linear value-function approximation, and feature selection for reinforcement learning. In *Proc. ICML*, 2008.

[12] A. Farahmand, M. Ghavamzadeh, C. Szepesvári, and S. Mannor. Regularized fitted Q-iteration for planning in continuous-space Markovian decision problems. In *Proc. ACC*. IEEE Press, 2009.

[13] M. Loth, M. Davy, and P. Preux. Sparse temporal difference learning using LASSO. In *IEEE International Symposium on Approximate Dynamic Programming and Reinforcement Learning*, 2007.

[14] J. Johns and S. Mahadevan. Sparse approximate policy evaluation using graph-based basis functions. Technical Report UM-CS-2009-041, University of Massachusetts Amherst, Department of Computer Science, 2009.

[15] S. Mallat and Z. Zhang. Matching pursuits with time-frequency dictionaries. *IEEE Transactions on Signal Processing*, 41(12):3397–3415, 1993.

[16] Y. Pati, R. Rezaiifar, and P. Krishnaprasad. Orthogonal matching pursuit: Recursive function approximation with applications to wavelet decomposition. In *Proceedings of the 27th Annual Asilomar Conference on Signals, Systems, and Computers*, volume 1, pages 40–44, 1993.

[17] M. Lagoudakis and R. Parr. Least-squares policy iteration. *Journal of Machine Learning Research*, 4:1107–1149, 2003.

[18] S. Lee and H. Seol. A survey on the matrix completion problem. *Trends in Mathematics*, 4(1):38–43, 2001.

[19] J. Júdice and F. Pires. Basic-set algorithm for a generalized linear complementarity problem. *Journal of Optimization Theory and Applications*, 74(3):391–411, 1992.

[20] M. Puterman and M. Shin. Modified policy iteration algorithms for discounted Markov decision problems. *Management Science*, 24(11), 1978.

[21] R. Sutton and A. Barto. *Reinforcement Learning: An Introduction*. MIT Press, 1998.

A Appendix

A.1 BLCP to LCP Reformulation

Here we generalize an observation from Júdice and Pires [19] to show that any BLCP can be converted to an LCP of *twice* the original dimension. Recall that the BLCP seeks vectors w and z such that $w = q + Mz$ and each variable z_i meets one of the following conditions:

$$\begin{aligned} z_i = u_i \quad &\Longrightarrow \quad w_i \leq 0 \\ z_i = l_i \quad &\Longrightarrow \quad w_i \geq 0 \\ l_i < z_i < u_i \quad &\Longrightarrow \quad w_i = 0 \end{aligned}$$

where the vectors l and u are lower and upper bounds respectively. An LCP must have nonnegative variables, so the BLCP conversion begins by introducing a nonnegative slack variable $z^+ = z - l$. This slight change allows for writing the problem as $w = (q + Ml) + Mz^+$ where:

$$\begin{aligned} z_i^+ = u_i - l_i \quad &\Longrightarrow \quad w_i \leq 0 \\ z_i^+ = 0 \quad &\Longrightarrow \quad w_i \geq 0 \\ 0 < z_i^+ < u_i - l_i \quad &\Longrightarrow \quad w_i = 0. \end{aligned}$$

The next step is to deal with the fact that variable w can be negative. We do so by introducing two sets of nonnegative variables such that $w = w^+ - w^-$. Now, the problem can be written $w^+ = (q + Ml) + Mz^+ + w^-$ where:

$$\begin{aligned} z_i^+ = u_i - l_i \quad &\Longrightarrow \quad w_i^+ = 0 \;\; \& \;\; w_i^- \geq 0 \\ z_i^+ = 0 \quad &\Longrightarrow \quad w_i^+ \geq 0 \;\; \& \;\; w_i^- = 0 \\ 0 < z_i^+ < u_i - l_i \quad &\Longrightarrow \quad w_i^+ = w_i^- = 0. \end{aligned}$$

At this point, the conditions can be expanded into an LCP with twice the original dimension. To help in this conversion, we add another slack variable $z^- = u - z = u - l - z^+$. Now the problem becomes:

$$\begin{bmatrix} w^+ \\ z^- \end{bmatrix} = \begin{bmatrix} q + Ml \\ u - l \end{bmatrix} + \begin{bmatrix} M & I \\ -I & 0 \end{bmatrix} \begin{bmatrix} z^+ \\ w^- \end{bmatrix}$$

with variables $w^+, w^-, z^+, z^- \geq 0$ and $z_i^+ w_i^+ = z_i^- w_i^- = 0, \forall i$. If matrix M is invertible then we can pivot on M, yielding the following equivalent LCP:

$$\begin{bmatrix} z^+ \\ z^- \end{bmatrix} = \begin{bmatrix} -l - M^{-1}q \\ u + M^{-1}q \end{bmatrix} + \begin{bmatrix} M^{-1} & -M^{-1} \\ -M^{-1} & M^{-1} \end{bmatrix} \begin{bmatrix} w^+ \\ w^- \end{bmatrix}$$

with variables $w^+, w^-, z^+, z^- \geq 0$ and $z_i^+ w_i^+ = z_i^- w_i^- = 0, \forall i$. Given the solution to this LCP, the original solution to the BLCP can be computed via $w = w^+ - w^-$ and $z = z^+ + l$.

A.2 LCP formulation of the L_1 Regularized Linear Fixed Point

Here, we derive an LCP formulation of the L_1 reguarized fixed point directly. Some readers may find this presentation more intuitive than the BLCP formulation.

Begin with the LARS-TD optimality conditions. Each coefficient w_i must meet one of the following conditions:

$$\begin{aligned} -\beta < \Phi_i^T(R - (\Phi - \gamma\Phi'^{\pi})w) < \beta \quad &\Longrightarrow \quad w_i = 0 \\ \Phi_i^T(R - (\Phi - \gamma\Phi'^{\pi})w) = \beta \quad &\Longrightarrow \quad w_i \geq 0 \\ \Phi_i^T(R - (\Phi - \gamma\Phi'^{\pi})w) = -\beta \quad &\Longrightarrow \quad w_i \leq 0 \end{aligned}$$

These can be rewritten as:

$$\begin{aligned} 0 < \beta - \Phi_i^T(R - (\Phi - \gamma\Phi'^{\pi})w) < 2\beta \quad &\Longrightarrow \quad w_i = 0 \\ \beta - \Phi_i^T(R - (\Phi - \gamma\Phi'^{\pi})w) = 0 \quad &\Longrightarrow \quad w_i \geq 0 \\ \beta - \Phi_i^T(R - (\Phi - \gamma\Phi'^{\pi})w) = 2\beta \quad &\Longrightarrow \quad w_i \leq 0 \end{aligned}$$

Now define two vectors x^+ and x^- as:

$$x^+ = \beta - \Phi^T(R - (\Phi - \gamma\Phi'^\pi)w)$$
$$x^- = \beta + \Phi^T(R - (\Phi - \gamma\Phi'^\pi)w) = 2\beta - x^+$$

Notice that both x^+ and x^- are in the range $[0, 2\beta]$. Furthermore, when an element $x_i^+ = 0$, it must be that $x_i^- = 2\beta$ (and vice-versa). With these definitions in hand, we can write the optimality conditions as:

$$\begin{aligned} 0 < x_i^+ < 2\beta \;\;\&\;\; 0 < x_i^- < 2\beta \;\;&\Longrightarrow\;\; w_i = 0 \\ x_i^+ = 0 \;\;\&\;\; x_i^- = 2\beta \;\;&\Longrightarrow\;\; w_i \geq 0 \\ x_i^- = 0 \;\;\&\;\; x_i^+ = 2\beta \;\;&\Longrightarrow\;\; w_i \leq 0 \end{aligned}$$

Now decompose the weight vector w into positive (w^+) and negative (w^-) portions such that:

$$\begin{aligned} w &= w^+ - w^- \\ w^+, w^- &\geq \mathbf{0} \\ \text{if } w_i \geq 0, \text{ then } w_i^+ \geq 0 \;\;\&\;\; w_i^- &= 0 \\ \text{if } w_i \leq 0, \text{ then } w_i^- \geq 0 \;\;\&\;\; w_i^+ &= 0 \end{aligned}$$

When we convert the optimality conditions to a linear complementarity problem (LCP), we will show that a solution to the LCP must satisfy these constraints placed on w^+ and w^-. The optimality conditions can now be written as:

$$\begin{aligned} 0 < x_i^+ < 2\beta \;\;\&\;\; 0 < x_i^- < 2\beta \;\;&\Longrightarrow\;\; w_i^+ = w_i^- = 0 \\ x_i^+ = 0 \;\;\&\;\; x_i^- = 2\beta \;\;&\Longrightarrow\;\; w_i^+ \geq 0 \;\;\&\;\; w_i^- = 0 \\ x_i^- = 0 \;\;\&\;\; x_i^+ = 2\beta \;\;&\Longrightarrow\;\; w_i^- \geq 0 \;\;\&\;\; w_i^+ = 0 \end{aligned}$$

Now it is easy to show these optimality conditions define a LCP. To help see this, note that (1) all the variables (x^+, x^-, w^+, w^-) must be nonnegative and (2) the only way a weight w_i^+ (or w_j^-) can be positive is if the corresponding variable x_i^+ (or x_j^-) equals zero. Using the notation $A = \Phi^T(\Phi - \gamma\Phi'^\pi)$ and $b = \Phi^T R$, the LCP is written:

$$\begin{aligned} \begin{bmatrix} x^+ \\ x^- \end{bmatrix} &= \left(\beta - \begin{bmatrix} b \\ -b \end{bmatrix}\right) + \begin{bmatrix} A & -A \\ -A & A \end{bmatrix} \begin{bmatrix} w^+ \\ w^- \end{bmatrix} \\ \begin{bmatrix} x^+ \\ x^- \end{bmatrix}, \begin{bmatrix} w^+ \\ w^- \end{bmatrix} &\geq \mathbf{0} \\ x_i^+ w_i^+ &= 0 \;\; \forall i \\ x_i^- w_i^- &= 0 \;\; \forall i \end{aligned}$$

Notice that a solution to the LCP must meet the constraints we placed on w^+ and w^-. To see this, assume $w_i^+ > 0$. If this is true, then x_i^+ must equal 0 (by complementarity) and $x_i^- = 2\beta$ (by definition) which in turn implies $w_i^- = 0$ (by complementarity). The same logic applies if $w_i^- > 0$.

Finally, note that the solution above is equivalent to performing the transformation in section A.1 to the BLCP formulation of the L_1 regularized fixed point.

A.3 The LARQ algorithm

The LARQ algorithm (Figure 2) modifies LARS-TD to identify policy change points in addition to feature change points. These policy change points, much like the feature addition/removal points of LARS-TD, can be found analytically. Policy changes can be made at a step size given by the equation

$$\alpha = \min_{a,i} -\frac{\Phi_i'^a \mathbf{w} - \Phi_i'^\pi \mathbf{w}}{\Phi_i'^a \Delta\mathbf{w} - \Phi_i'^\pi \Delta\mathbf{w}}, \tag{6}$$

where $\Delta\mathbf{w}$ is the vector along which the coefficients are currently constrained to move according to the LARS-TD invariants.

Algorithm LARQ ($\{s_i, a_i, r_i, s'_i\}, \gamma, \{\varphi_j\}, \beta$)

parameters:

$\{s_i, a_i, r_i, s'_i\}$ state, action, reward, and nextstate samples
γ : discount factor
$\{\varphi_j\}$: basis functions
β : regularization parameter

initialization:

$\mathbf{w} \leftarrow \mathbf{0}$
$\boldsymbol{\pi} : \pi(s'_i) \leftarrow 1$
$\boldsymbol{\Phi} : \Phi_{ij} \leftarrow \varphi_j(s_i, a_i)$
$\mathbf{c} \leftarrow \boldsymbol{\Phi}^T R$
$\{\bar{\beta}, i\} \leftarrow \max_j |c_j|$
$\mathcal{I} \leftarrow \{i\}$

while $\bar{\beta} > \beta$:

Get features for next states following policy $\boldsymbol{\pi}$:
$\boldsymbol{\Phi'}^{\pi} : \Phi'^{\pi}_{ij} \leftarrow \varphi_j(s_i, \pi(s_i))$

Find update direction:
$\Delta\mathbf{w}_{\mathcal{I}} \leftarrow (\boldsymbol{\Phi}_{\mathcal{I}}^T \boldsymbol{\Phi}_{\mathcal{I}} - \gamma \boldsymbol{\Phi}_{\mathcal{I}}^T \boldsymbol{\Phi}'^{\pi}_{\mathcal{I}})^{-1} \operatorname{sgn}(\mathbf{c}_{\mathcal{I}})$

Find step size for adding a feature to the active set:
$\mathbf{d} \leftarrow (\boldsymbol{\Phi}^T (\boldsymbol{\Phi}_{\mathcal{I}} - \gamma \boldsymbol{\Phi}'^{\pi}_{\mathcal{I}}) \Delta\mathbf{w}_{\mathcal{I}}$
$\{\alpha_1, i_1\} \leftarrow \min^{+}_{j \notin \mathcal{I}} \left\{ \frac{c_j - \bar{\beta}}{d_j - 1}, \frac{c_j + \bar{\beta}}{d_j + 1} \right\}$

Find step size for removing a feature from the active set:
$\{\alpha_2, i_2\} \leftarrow \min^{+,0}_{j \in \mathcal{I}} \left\{ -\frac{w_j}{\Delta w_j} \right\}$

Find step size to first of {add a feature, remove a feature, terminate at fixed point}:
$\alpha \leftarrow \min\{\alpha_1, \alpha_2, \bar{\beta} - \beta\}$

// This next section is omitted in LARS-TD
Find step size for greedy policy update
$Q'^{\pi} \leftarrow \boldsymbol{\Phi'}^{\pi}_{\mathcal{I}} \mathbf{w}_{\mathcal{I}}$
$\Delta Q'^{\pi} \leftarrow \boldsymbol{\Phi'}^{\pi}_{\mathcal{I}} \Delta\mathbf{w}_{\mathcal{I}}$
foreach action a
$Q'^{a} : Q'^{a}_i \leftarrow \Phi(s'_i, a)_{\mathcal{I}} \mathbf{w}_{\mathcal{I}}$
$\Delta Q'^{a} : \Delta Q'^{a}_i \leftarrow \Phi(s'_i, a)_{\mathcal{I}} \Delta\mathbf{w}_{\mathcal{I}}$
$\{\alpha^a_3, i^a_3\} \leftarrow \min_i \left\{ -\frac{Q'^{a}_i - Q'^{\pi}_i}{\Delta Q'^{a}_i - \Delta Q'^{\pi}_i} \text{ such that } \Delta Q'^{a}_i > \Delta Q'^{\pi}_i \right\}$
$\{\alpha_3, \pi'_3\} \leftarrow \min_a \{\alpha^a_3\}$
$i_3 \leftarrow i^{\pi'_3}_3$
$\alpha \leftarrow \min\{\alpha, \alpha_3\}$
// End of LARQ-only section

Update weights, $\bar{\beta}$, and correlation vector:
$\mathbf{w}_{\mathcal{I}} \leftarrow \mathbf{w}_{\mathcal{I}} + \alpha \Delta\mathbf{w}_{\mathcal{I}}$
$\bar{\beta} \leftarrow \bar{\beta} - \alpha$
$\mathbf{c} \leftarrow \mathbf{c} - \alpha\mathbf{d}$

Update active set or policy:
if $(\alpha = \alpha_1), \mathcal{I} \leftarrow \mathcal{I} \bigcup \{i_1\}$
else if $(\alpha = \alpha_2), \mathcal{I} \leftarrow \mathcal{I} - \{i_2\}$
else if $(\alpha = \alpha_3), \pi(i_3) \leftarrow \pi'_3$
end while
return $\mathbf{w}$.

Figure 2: The LARQ algorithm.

To satisfy the greedy invariant of LARQ, however, we must also ensure that the new policy we switch to will be better than the old policy as we move towards the new fixed point. In general, all that is necessary to satisfy this requirement is that the gradient of the new policy, $\Phi'^{\pi'}_i \Delta\mathbf{w}$ with respect to the direction of travel is greater than the gradient of the old policy.

A.4 LCP Solver

In our experiments, we used a modified complementary pivoting algorithm to solve for the L_1TD fixed point using the LCP formulation as shown at the end of appendix section A.1. We adapted a Matlab LCP solver due to P. Fackler and M. Miranda that allows for warm starts. The file `lemke.m` can be found in the CompEcon toolbox at `http://www4.ncsu.edu/~pfackler`. The code was adapted so that the full matrix $\Phi^T(\Phi - \gamma\Phi'^{\pi})$, which can be very large depending on the size of the feature set, was never explicitly formed. The full matrix is not needed to determine which element should be pivoted in and out of the active set at any iteration of the algorithm. We only formed the principal submatrix associated with the active elements. As a further optimization, rather than forming the principal submatrix itself, it is possible to update and downdate the *inverse* of this principal submatrix incrementally to solve the necessary linear system of equations.

In section 2.3, we stated that the LCP formulation is twice as large as the BLCP formulation. We used the LCP formulation here because the code was readily available and appeared robust. However, any suitable solver could be used instead.

A.5 Robustness of LARS-TD

When the A matrix is a P-matrix, LC-TD and LARS-TD are guaranteed to produce solutions that achieve the L_1TD optimality conditions. When the P-matrix property does not hold, both algorithms have the potential to violate the optimality conditions. In practice, we found LARS-TD to be much more susceptible to this problem. We discuss here a possible rationale explaining this behavior.

There are two ways that LARS-TD can violate the invariants: (1) the sign of the correlation may disagree with the sign of the weights for features in the active set, and (2) once a feature is removed from the active set, the absolute value of its correlation may in fact increase compared to the correlation of features in the active set. Both of these problems are of course detectable while running the algorithm. Instead of terminating prematurely when one of these two problems arises (in which case the algorithm would *not* return a set of coefficients for the desired value of the regularization parameter β), our LARS-TD implementation simply ignores the deviations and continues. This decision was made for practical reasons. We found it better for LARS-TD to return a set of coefficients at the requested value of β than to terminate early and return coefficients at a larger value of β. However, these LARS-TD violations did impact performance. For example, for one setting of β in our mountain car experiments, we found that the final policy learned using LC-TD reached the goal in 19 out of 20 trials, taking 136 ± 22 steps on average. The policies learned using LARS-TD reached the goal in 17 of 20 trials, taking 161 ± 41 steps on average.

The fact that LARS-TD produced solutions violating the L_1TD optimality conditions explains the empirical behavior shown in Figure 1. In those plots, notice that policy iteration converged after six rounds when using LC-TD whereas it did not converge at all when using LARS-TD. Recall that convergence is obtained when the coefficients w do not change significantly from one round to the next. Since LARS-TD was frequently violating the optimality conditions, it was not reaching an L_1TD fixed point whereas LC-TD was doing so. Thus, LC-TD was able to produce the same set of coefficients in two subsequent rounds of policy iteration. LARS-TD would violate the optimality conditions in different ways in subsequent rounds of policy iteration and therefore never stabilized on one set of coefficients.

We offer the following explanation as to why a homotopy method like LARS-TD may be more prone to violating the optimality conditions in comparison to an LCP solver. Consider the behavior of both LARS-TD and LC-TD for a fixed Φ, Φ'^{π}, R, and γ but a varying β. For a particular non-P-matrix $\Phi^T(\Phi - \gamma\Phi'^{\pi})$, there will exist at least one R such that there is not a unique solution achieving the L_1TD optimality conditions. As demonstrated by Kolter and Ng [1] (Figure 2b), changes in the number of solutions as β varies can correspond to discontinuities in the homotopy path. If LARS-TD encounters such a point, then it will fail to preserve the invariants when it crosses the discontinuity and it is unlikely to re-establish them. This is because the algorithm does not make the sort of discontinuous changes needed to jump onto a different homotopy path. In contrast, LC-TD directly solves the problem for a particular β (or perhaps a finite set of β values). Discontinuities in the homotopy path have no direct impact on the LCP solver's ability to find a solution. Moreover, if solutions are desired at a set of β values, a failure to find a solution for any particular value will not taint the solver's search for a valid solution at other values, even if using warm starts. In summary, a

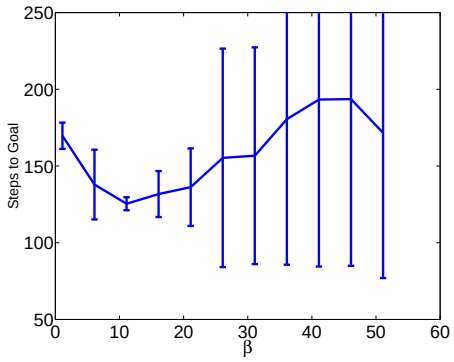


Figure 3: Average performance of final policy produced by LC-MPI for the mountain car MDP at different values of the regularization parameter β.

negative aspect of a homotopy method is that it can be derailed if a problem occurs at any point along the continuous homotopy path, while a direct method that searches for a solution at a particular value of the regularization parameter is influenced only by the properties of the solution at that particular value.

A.6 Use of Cross-Validation

Lastly, to show how one may use the results of LC-MPI, Figure 3 shows the average performance of the 11 greedy policies for mountain car. The graph shows the number of steps to reach the goal up to a maximum of 350 steps per trial. The large error bars for large values of β are due to poor policies resulting in tests that do not reach the goal within 350 steps. Notice the classic "bathtub" shape in which under- and over-regularized solutions perform poorly.