Multigrid Algorithms for Temporal Difference Reinforcement Learning

Omer Ziv OMERZ@V-TARGET.COM
Department of Electrical Engineering, Technion – Israel Institute of Technology, Haifa 32000, Israel

Nahum Shimkin SHIMKIN@EE.TECHNION.AC.IL
Department of Electrical Engineering, Technion – Israel Institute of Technology, Haifa 32000, Israel

Key Words: Multigrid, temporal difference learning, basis function hierarchies.

Abstract

We introduce a class of Multigrid based temporal difference algorithms for reinforcement learning with linear function approximation. Multigrid methods are commonly used to accelerate convergence of iterative numerical computation algorithms. The proposed Multigrid-enhanced TD(λ) algorithms allows to accelerate the convergence of the basic TD(λ) algorithm while keeping essentially the same per-sample computational cost. We propose two versions of the algorithm, a sequential and synchronous one, establish the convergence of the latter, and provide a simulation example that demonstrates the potential performance benefits.

1. Introduction

Reinforcement Learning (RL) is concerned with on-line computation of effective control policies, based on interaction with the controlled environment (Bertsekas & Tsitsiklis, 1996; Sutton & Barto, 1998). Special emphasis is placed on techniques for handling large and complex problems, in particular models with a large state space. Parameterized function approximators are commonly invoked to provide compact representations and learning generalization.

Temporal difference algorithms for evaluating the value function of a given policy are a central component in many RL schemes. The basic TD(λ) algorithm was introduced in Sutton (1988), and its convergence with linear function approximation was analyzed in Tsitsiklis and Van Roy (1997). Recently, two new classes of related algorithms were introduced, LSTD(λ) (Boyan, 2002; Xu, He & Hu, 2002) and λ-LSPE (Nedić & Bertsekas, 2003; Bertsekas, Borkar & Nedić, 2004). These algorithms are essentially of a least-squares nature, and compared to TD(λ) offer faster convergence in terms of the data sample count, while the computational complexity per sample increases from $O(K)$ to $O(K^2)$ at least, where K is the number of parameters to be learned. Consequently, when K is large these algorithms may not be feasible. In this paper we propose a Multigrid-based enhancement of the TD(λ) algorithm, which aims to improve the convergence rate while retaining the same $O(K)$ complexity per iteration.

Multigrid (Briggs, Henson and McCormick, 2000; Trottenberg, Oosterlee & Schüller, 2001) is a well-established approach to accelerate iterative solutions of large sets of linear equations, such as those arising in the numerical solution of partial differential equations. Essentially, an iterative relaxation scheme at a fine resolution level is augmented by a coarse-grid correction which reduces the so-called "smooth" error components, which are otherwise slow to converge. Applying this correction recursively over several resolution levels leads to a Multigrid scheme. When applied to value iteration or TD(λ) with linear function approximation, this approach leads to algorithms that operate with different sets of basis functions, each intended to capture a different resolution level of the problem. We shall focus in particular on the Algebraic Multigrid (AMG) variant of Multigrid, which allows the automatic construction of the coarse grid hierarchies based on the system matrices. This opens up interesting possibilities for the automatic construction of basis function hierarchies.

We shall propose two variants of a Multigrid learning algorithm. The Sequential Multigrid TD(λ) algorithm, or SeqMGTD(λ), operates on each grid level sequentially, similar to standard Multigrid, with only one level active at a time. In Simultaneous Multigrid TD(λ), or SimMGTD(λ), computations at all resolution levels are performed simultaneously as soon as a new data point is available. As will be shown, the latter algorithm

To appear (in a shorter version) in *Proceeding of the ICML'05 Workshop on Rich Representations for Reinforcement Learning,* Bonn, Germany, August 2005.

converges to the exact (fine-level) solution of the problem. The analysis of this algorithm will further point to some possible enhancements of the standard TD(λ) algorithm by modifying its eligibility trace vector.

An extensive literature exists on hierarchical and multiscale methods in Dynamic Programming and RL, pointers to which may be found in (Boutillier, Dean, & Hanks, 1999; Barto & Mahadevan, 2003). In the context of iterative policy evaluation, the aggregation-disaggregation algorithm of Schweitzer, Puterman & Kindle (1985) (see also Shweitzer, 1991) uses coarse level corrections over fixed state aggregates, but with "aggregation directions" (or inter-level operators as defined below) that are re-computed at each iteration, while Bertsekas & Castañon (1989) propose a related scheme with adaptive state aggregates. The Multigrid framework, on the other hand, essentially relies on a fixed multi-level structure, an approach which facilitates its incorporation in the TD(λ) algorithm with little overhead. An application of Multigrid methods to Q-learning for controlled diffusion processes is reported in Pareigis (1997), where the focus is on the relation between time and space discretization.

The paper is structured as follows. Sections 2 and 3 provide the necessary background on Multigrid and TD(λ), respectively. Section 4 outlines the (straightforward) application of Multigrid to value iteration for policy evaluation in a non-learning scenario. Section 5 presents the Multigrid learning algorithms are their analysis, while Section 6 presents a basic simulation experiment, followed by concluding remarks.

2. Multigrid Basics

We consider the efficient solution of the system of linear equations $\mathbf{Ax}=\mathbf{b}$, where $\mathbf{A}$ is a square matrix and typically sparse. Standard iterative methods are of the form $\mathbf{x}:=\mathbf{x}+\mathbf{Q}^{-1}(\mathbf{b}-\mathbf{Ax})$, where $\mathbf{Q}$ stand for a scaled identity matrix (Richardson iteration), the diagonal of $\mathbf{A}$ (Jacoby relaxation), or its lower-triangular part (Gauss-Seidel). When the *smoothing matrix* $(\mathbf{I}-\mathbf{Q}^{-1}\mathbf{A})$ has eigenvalues close to the unit circle, the corresponding error components are slow to converge. Such error components are referred to as "smooth", and typically correspond to "low frequency" components in a geometric context. Multigrid uses coarse-level corrections to reduce these smooth error components.

A multigrid structure comprises of: (a) A sequence of subsequent resolution levels indexed by $\ell\in\{0,1,\ldots,\ell_{\max}\}$, with $\ell=0$ the finest; (b) A corresponding set of equations $\mathbf{A}_\ell\mathbf{x}_\ell=\mathbf{b}_\ell$ of dimensions n_ℓ, where $\mathbf{A}_0,\mathbf{b}_0$ are the primary (fine-resolution) system matrices, $\mathbf{A}_\ell,\mathbf{b}_\ell$ represent the system equations at resolution level ℓ, and $n_{\ell+1}$ is several times smaller than n_ℓ (a factor of 4 is common for 2D problems); (c) *Restrictor operators* $\mathbf{I}_\ell^{\ell+1}$ which turn a solution $\mathbf{x}_\ell$ into an approximate solution $\mathbf{x}_{\ell+1}=\mathbf{I}_\ell^{\ell+1}\mathbf{x}_\ell$ of the next-coarser level; and (d) *Interpolators* $\mathbf{I}_{\ell+1}^{\ell}$ ($n_\ell\times n_{\ell+1}$ matrices), which do the opposite.

A basic two-level coarse grid correction at level $\ell<\ell_{\max}$ proceeds as follows. Starting with a vector $\mathbf{x}_\ell$, an approximate solution to the equation $\mathbf{A}_\ell\mathbf{x}_\ell=\mathbf{r}_\ell$ (with $\mathbf{r}_\ell$ to be defined shortly) is obtained as follows:

1. Presmoothing: Apply a (small) number of iterative relaxations $\mathbf{x}_\ell:=\mathbf{x}_\ell+\mathbf{Q}_\ell^{-1}(\mathbf{r}_\ell-\mathbf{A}_\ell\mathbf{x}_\ell)$
2. Compute the residual $res_\ell=\mathbf{r}_\ell-\mathbf{A}_\ell\mathbf{x}_\ell$, and restrict to the next level: $\mathbf{r}_{\ell+1}=\mathbf{I}_\ell^{\ell+1}res_\ell$
3. Approximately solve $\mathbf{A}_{\ell+1}\mathbf{x}_{\ell+1}=\mathbf{r}_{\ell+1}$
4. Apply correction: $\mathbf{x}_\ell:=\mathbf{x}_\ell+\mathbf{I}_{\ell+1}^{\ell}\mathbf{x}_{\ell+1}$
5. Postsmoothing: Similar to presmoothing.

By recursively applying this procedure at step 3 we obtain a multi-grid scheme. A standard *V-cycle* starts at level 0 with $\mathbf{r}_0=\mathbf{b}_0$ and proceeds all the way down to level $\ell_{\max}$ and back up. Note that the system vectors $\mathbf{b}_\ell$ (for $\ell\geq 1$) do not play any role here as they are replaced by the interpolated residuals $\mathbf{r}_\ell$. At the coarsest level $\ell=\ell_{\max}$ the dimension is typically chosen to be sufficiently small so that the equation $\mathbf{A}_\ell\mathbf{x}_\ell=\mathbf{r}_\ell$ may be solved exactly. We note that other, more involved cycles are often used as well. The whole scheme is usually initialized with some "coarse to fine" procedure which does utilize the system vectors $(\mathbf{b}_\ell)$.

In classical (geometric) Multigrid, the system equations at the different levels are typically obtained by discretizing the original (continuous) problem over a regular grid at different resolutions. The inter-level (restriction and interpolation) operators are then constructed. A judicious choice of these operators is critical for the efficiency of the method, especially in the presence of discontinuities and other spatial irregularities, and is highly problem dependent.

2.1 Algebraic Multigrid (AMG)

AMG (Brandt, McCormick & Ruge, 1984; Stüben, 2001) takes a different approach. Here the multigrid structure is constructed automatically in a setup phase from the initial system matrices $\mathbf{A}_0,\mathbf{b}_0$, based only on the algebraic structure of $\mathbf{A}_0$ and without any “higher level" information on the problem. This makes AMG attractive as a "black box" solver for sparse linear equations, whether of geometric origin or not.

The AMG setup phase proceeds recursively, starting at $\ell = 0$. First the inter-level operators $\mathbf{I}_{\ell+1}^{\ell}$ and $\mathbf{I}_{\ell}^{\ell+1}$ are constructed based on $\mathbf{A}_\ell$. The system matrices for the next level are then defined, typically via the *Galerkin operator* $\mathbf{A}_{\ell+1} = \mathbf{I}_{\ell}^{\ell+1}\mathbf{A}_\ell\mathbf{I}_{\ell+1}^{\ell}$, and $\mathbf{b}_{\ell+1} = \mathbf{I}_{\ell}^{\ell+1}\mathbf{b}_\ell$. This proceeds until the dimension of $\mathbf{A}_\ell$ is sufficiently small for a direct solution.

Several procedures exist for the definition of the inter-level operators at the setup phase. The guiding principle is to allow any *algebraically smooth* error vector to be well approximated over the next level, namely by some interpolated vector of that level. For concreteness we briefly describe a simple scheme (Ruge-Stüben algorithm with direct interpolation) which applies to matrices $\mathbf{A}_\ell = (a_{ij})$ with predominantly non-positive off-diagonal elements (the diagonal is always taken to be positive). Full details and generalizations can be found in (Stüben, 2001; section A.7). A variable i is said to be *strongly coupled* to a variable j if

$$-a_{ij} \geq \varepsilon \max_{k\neq i}\{-a_{ik}\}$$

where $\varepsilon \in (0,1)$ is a design parameter, with typical values around 0.25. We start by a *coarsening* process which splits the variables $(1,\dots,n_\ell)$ into two disjoint sets C and F, with the $n_{\ell+1}$ variable in C are the next-level coarse variables. The general objective is to ensure a strong connectivity of each F-variable to C-variables; in particular, a simple one-pass algorithm can ensure that each F-variable is strongly coupled to some C-variable. Given this partition, the interpolation operator is defined by

$$\left(\mathbf{I}_{\ell+1}^{\ell}\mathbf{e}\right)_i = \begin{cases} e_i & : \ i \in C \\ -\alpha_i \sum_{k\in P_i} \dfrac{a_{ik}^-}{a_{ii}} e_k & : \ i \in F \end{cases}$$

where $a^- \leq 0$ denotes the negative part of a, $P_i \subset C$ is the set of C-variables to which i is strongly coupled, and

$$\alpha_i = \frac{\sum_j a_{ij}^-}{\sum_{j\in P_i} a_{ij}^-} .$$

Finally, the restriction operator is defined as $\mathbf{I}_{\ell+1}^{\ell} = (\mathbf{I}_{\ell}^{\ell+1})^T$.

Simpler interpolators are used in a (strict) *aggregation* scheme, where each F-variable is interpolated from exactly one F-variable. The variables are thus effectively partitioned into disjoint aggregates, each represented by a single coarse variable. Here we may define

$$\left(\mathbf{I}_{\ell+1}^{\ell}\right)_{ik} = \begin{cases} 1 & \text{for a single } k \text{ for which } -a_{ik} = \max_{j\neq i}\{-a_{ij}\} \\ 0 & : \ \text{otherwise} \end{cases}$$

Aggregation schemes often result in inferior performance, but are somewhat simpler to implement.

Multigrid theory aims to establish convergence of the iterative algorithm and, more importantly, to provide bounds on the convergence rate and guidelines for algorithm improvement. A well developed theory currently exists mainly for problems in which the system matrix $\mathbf{A}$ is symmetric and positive-definite (s.p.d.), and, in particular, when $\mathbf{A}$ is also an M-matrix (namely s.p.d. with negative off-diagonal elements) and diagonally dominant. In practice, properly planned algorithms (and AMG in particular) are robust with respect to violation of these assumptions.

3. MDPs and the TD(λ) Algorithm

Consider a Markov Decision Process (MDP) with state $\mathcal{S}$ and action space $\mathcal{A}$. We assume here a finite state space (but note that the proposed learning algorithms are applicable to more general state spaces due to the use of basis functions). Given the state s_t and action a_t at time t, a reward $g_t = g(s_t, a_t)$ is obtained, and the next state s_{t+1} is determined according to the stationary transition probability $p(s_{t+1} \mid s_t, a_t)$.

A *stationary policy* π is a mapping $\pi : \mathcal{S}\times\mathcal{A} \to [0,1]$, where $\pi(s,a)$ is the probability of taking action a at state s. Fixing the policy π, the state process becomes a Markov chain with transition probabilities $p(s' \mid s) = \sum_a \pi(s,a)p(s' \mid s,a)$, and expected rewards $g(s) = \sum_a \pi(s,a)g(s,a)$. We shall assume that the induced Markov chain is *irreducible*, *a-periodic*, with a unique stationary distribution $q(s)$. For future reference we denote by $\mathbf{P}$ the transition matrix with $\mathbf{P}_{s,s'} = p(s' \mid s)$, the reward vector $\mathbf{g}$ with elements $\mathbf{g}_s = g(s)$, and the diagonal matrix $\mathbf{D}$ with $\mathbf{D}_{ss} = q(s)$. We consider the discounted cost functional with a discount factor $\gamma \in (0,1)$, namely $\mathrm{v}(s) = E(\sum_{t=0}^{\infty}\gamma^t g_t \mid s_0 = s)$. The function $\mathrm{v}(s)$ of the stationary policy π is well known to be the unique solution of the Bellman equation

$$(\mathbf{I}-\gamma\mathbf{P})\,\mathbf{v}=\mathbf{g} \tag{1}$$

where $\mathbf{I}$ denotes the identity matrix and $\mathbf{v}$ is a vector of state values, i.e. $\mathbf{v}_s = \mathrm{v}(s)$. The value function is approximated as a linear combination of K basis functions $\{\phi_k : S \to \Re^K\}$, namely

$$\mathrm{v}(s) \approx \sum\nolimits_{k=1}^{K} \phi_k(s)\,\theta_k = \phi(s)^T\,\theta$$

where $\phi = (\phi_1,\ldots,\phi_K)$, and $\theta \in \Re^K$ is the parameter vector to be tuned. The TD(λ) algorithm iteratively applies the following update rule

$$\theta_t = \theta_{t-1} + \alpha_t \mathbf{z}_t \left(g_t - \left(\phi(s_t) - \gamma\phi(s_{t+1})\right)^T \theta_{t-1} \right);$$

$$\mathbf{z}_t = \lambda\gamma\mathbf{z}_{t-1} + \phi(s_t)$$

where $\mathbf{z}_t = (z_t(s), s \in S)$ is the *eligibility trace* vector, initialized by $\mathbf{z}_0 = \mathbf{0}$. $\lambda \in [0,1]$ is the algorithm parameter, and α_t is a positive *gain* sequence.

Theorem 1 *(Tsitsiklis & Van Roy, 1997). Assume that*
(i) The gain sequence satisfies

$$\sum\nolimits_{t=0}^{\infty} \alpha_t = \infty,\ \sum\nolimits_{t=0}^{\infty} \alpha_t^2 < \infty$$

(ii) The basis functions are linearly independent.
Then TD(λ) converges with probability 1 to the unique vector θ^* that satisfies

$$\mathbf{A}\theta^* = \mathbf{b} \tag{2}$$

where $\mathbf{A}$ is a K×K matrix and $\mathbf{b}$ a K×1 vector defined as follows

$$\mathbf{A} = \mathbf{\Phi}^T (\mathbf{I}-\gamma\lambda\mathbf{P})^{-1} \mathbf{D}(\mathbf{I}-\gamma\mathbf{P})\mathbf{\Phi} \tag{3}$$
$$\mathbf{b} = \mathbf{\Phi}^T \mathbf{D}(\mathbf{I}-\gamma\lambda\mathbf{P})^{-1}\mathbf{g}.$$

$\mathbf{\Phi}$ is the N×K matrix with basis functions as its columns, namely $\mathbf{\Phi}_{sk} = \phi_k(s)$.

4. AMG for Value Iteration

In this section we briefly consider value iteration for the known model case. Here the application of AMG as a "black box" solver to (1) is straightforward, by defining $\mathbf{A} = \mathbf{I}-\gamma\mathbf{P}$ and $\mathbf{b} = \mathbf{g}$. Observe that standard value iteration, namely $\mathbf{v} := \gamma\mathbf{P}\mathbf{v} + \mathbf{g}$, is equivalent to a Richardson relaxation of the corresponding linear system. It is well known that standard value iteration is slow to converge when $\gamma\mathbf{P}$ has an eigenvalue close to the unit disk, namely γ is close to 1, and this is exactly when that we expect AMG (and Multigrid in general) to provide a significant improvement.

Similarly, we can apply AMG to solve (2), with **A** and **b** as defined in (3). This is the starting point for the multigrid learning algorithms of the next section.

In the special case when the transition probability matrix **P** is symmetric, the matrix $\mathbf{A} = \mathbf{I}-\gamma\mathbf{P}$ turns out to be an M-matrix with strictly dominant diagonal, a case to which AMG theory nicely applies. Note also that **P** is typically sparse in practical problems, hence so is **A**, a property which is important for Multigrid efficiency. However, since **P** is hardly ever symmetric, theoretical performance bounds are not readily available. Nonetheless, standard AMG algorithms can be applied to the non-symmetric case without modification, and practical experience shows that they perform well even when symmetry is violated (e.g., Stüben, 2001, p. 518). It should also be noted that convergence to the exact solution can always be enforced, simply by "turning off" the coarse grid corrections at some point, or by monitoring the error reduction as done in other hierarchical schemes (Schweitzer *et. al.*, 1985; Bertsekas & Castañon, 1989). For the problems we tested, unforced convergence was always obtained.

5. Multigrid Temporal Difference Learning

To motivate the proposed Multigrid enhancement to TD(λ), we briefly consider the convergence of the mean of the parameter vector $\mathbb{E}\{\theta_t\}$, denoted $\overline{\theta}_t$. The stochastic dynamics of the TD(λ) algorithm, as derived in (Tsitsiklis & Van Roy, 1997), may be asymptotically approximated by $\overline{\theta}_{t+1} = \overline{\theta}_t + \alpha_t(\mathbf{b} - \mathbf{A}\overline{\theta}_t)$, where $\mathbf{A}$ and $\mathbf{b}$ are defined in (3). The error $\overline{\mathbf{e}}_t = \theta^* - \overline{\theta}_t$ relative to the fixed point $\theta^* = \mathbf{A}^{-1}\mathbf{b}$ satisfies $\overline{\mathbf{e}}_{t+1} = (\mathbf{I} - \alpha_t\mathbf{A})\overline{\mathbf{e}}_t$. TD(λ) may thus be interpreted as a stochastic smoother of the error. Multigrid is therefore a natural candidate for speeding up its convergence.

5.1 The SeqMGTD(λ) algorithm

Oyr first algorithm mimics the V-cycle of the Multigrid algorithm as described in Section 2. We assume that we are given an initial (fine-level) set of K basis functions, with corresponding feature vectors $\phi_0(s) = \phi(s)$, as well as a set of interpolators $\mathbf{I}_{\ell+1}^{\ell}$ and restrictors $\mathbf{I}_{\ell}^{\ell+1}$. We then recursively define feature vectors for all levels by

$$\phi_{\ell+1}(s)^T = \phi_\ell(s)^T \mathbf{I}_{\ell+1}^{\ell} \tag{4}$$

The algorithm is started by the function call θ_0:=seqMTD$(\theta_0, \ell=0)$, where θ_0 is an initial guess. The algorithm requires some switching criterion for the pre and post iterates. In the reported experiments we used the simplest rule of switching after a fixed number of iterations. Another reasonable option would be to increase the iteration count per level as the gain parameter decreases. The algorithm is started by the function call θ_0:=seqMTD$\left(\theta_0, \ell=0\right)$, where θ_0 is an initial guess. The estimated value function at the end of a complete cycle is given by $\mathrm{v}(s)=\phi_0(s)^T\theta_0$. The value function for intermediate times while level ℓ is completed is given more accurately by $\mathrm{v}(s)=\sum_{m=0}^{\ell}\phi_m(s)^T\theta_m$.

Table 1: Sequential Multigrid TD(λ) at level ℓ

```
SeqMGTD$\left(\theta_\ell^0, \ell, \theta_0, \theta_1, \ldots, \theta_{\ell-1}\right)$
1. **Initialize level correction**:   $\theta_\ell := \theta_\ell^0$, $\mathbf{z}_\ell := \mathbf{0}$
2. **Pre-iterate at level $\ell$ with residual rewards**:
   2.1. Observe the transition $s_t \to s_{t+1}$ and the reward
        $g_t$ at time $t$.
   2.2. Update the eligibility traces $\mathbf{z}_\ell := \lambda\gamma\mathbf{z}_\ell + \phi_\ell(s_t)$
   2.3. Sample the residual
        $r_\ell := g_t - \sum_{m=0}^{\ell-1}\left(\phi_m(s_t) - \gamma\phi_m(s_{t+1})\right)^T\theta_m$
   2.4. Calculate the temporal difference
        $d_\ell = r_\ell - \left(\phi_\ell(s_t) - \gamma\phi_\ell(s_{t+1})\right)^T\theta_\ell$
   2.5. Update $\theta_\ell := \theta_\ell + \alpha_{\ell,t}\mathbf{z}_\ell d_\ell$
   2.6. If the switching criterion is met then continue,
        otherwise repeat from 2.1.
3. **Apply coarse grid correction**: If $\ell \neq \ell_{\max}$,
   3.1. Recursive call
        $\theta_{\ell+1} := \text{MG-TD}\left(\theta_{\ell+1}^0 = \mathbf{0}, \ell+1, \theta_0, \theta_1, \ldots, \theta_\ell\right)$
   3.2. Correction using the interpolated error
        $\theta_\ell := \theta_\ell + \mathbf{I}_{\ell+1}^{\ell}\theta_{\ell+1}$
4. **Post-iterate**: repeat step 2 until meeting the
   switching criterion.
5. **Return** $\theta_\ell$.
```

To understand the proposed algorithm, note that the Multigrid algorithm (as presented in Section 2) at level ℓ aims at the solution of the equation $\mathbf{A}_\ell\mathbf{x}_\ell = \mathbf{r}_\ell$, where $\mathbf{r}_\ell$ is defined recursively via $\mathbf{r}_\ell = \mathbf{I}_{\ell-1}^{\ell}(\mathbf{r}_{\ell-1} - \mathbf{A}_{\ell-1}\mathbf{x}_{\ell-1})$. In the RL context, this equation cannot be represented explicitly since the $\mathbf{A}_\ell$ and $\mathbf{r}_\ell$ are unavailable. We resolve this problem by using an appropriate TD(λ) type iteration that serves as the iterative smoother for that level. First, the interpolated residual $\mathbf{r}_\ell$ is *sampled* as step 2.3, and serves as the driving reward signal for that stage. The TD(λ) algorithm then proceeds with the level-ℓ basis functions. A transition to level $\ell+1$ then takes place for the purpose of coarse grid correction, intended to accelerate the convergence of the smoothed error at level ℓ.

At the coarsest level ($\ell = \ell_{\max}$) step 3 and 4 should be skipped. Alternatively, the TD(λ) iteration at that stage may be replaced by another learning algorithm such as LSTD(λ) or λ-LSPE.

The algorithm is sequential in nature, as the different levels operate on non-overlapping time intervals. This implies, in particular, that to make full use of data points at each level requires data reuse, or *experience replay*. Resetting of the eligibility traces at the beginning of each stage is not necessary if temporal continuity of the data samples is maintained in subsequent activations of the same level.

The following claim demonstrates that each level *in isolation* approaches the solution of the desired equation at that level. I

Proposition 1. *Assume that the conditions of Theorem 1 are satisfied with the level-0 basis functions* $\phi_0(s)$*, and that all interpolator operators (*$\mathbf{I}_{\ell+1}^{\ell}$*) are full rank. If at some point the SeqMGTD(λ) algorithm is kept indefinitely at level* ℓ*, then* θ_ℓ *converges (w.p. 1) to the solution of the equation* $\mathbf{A}_\ell\theta_\ell = \mathbf{r}_\ell$*. Here* $\mathbf{A}_\ell$ *is defined as in (3) with* $\mathbf{\Phi}$ *replaced by* $\mathbf{\Phi}_\ell$*, and satisfies the recursion* $\mathbf{A}_\ell = \mathbf{I}_{\ell-1}^{\ell}\mathbf{A}_{\ell-1}\mathbf{I}_{\ell}^{\ell-1}$*; while* $\mathbf{r}_\ell$ *is defined recursively via* $\mathbf{r}_\ell = \mathbf{I}_{\ell-1}^{\ell}(\mathbf{r}_{\ell-1} - \mathbf{A}_{\ell-1}\theta_{\ell-1})$*, with* $\mathbf{r}_0$ *defined as* $\mathbf{b}$ *in (3) with* $\mathbf{\Phi}$ *replaced by* $\mathbf{\Phi}_0$.

The proof (details of which we omit here) follows from Theorem 1 by substituting r_ℓ for the reward signal, computing its expected value with respect to the invariant distribution, and some algebra.

Convergence of the overall algorithm cannot be established in general, as even the basic AMG algorithm is not guaranteed to converge without further restrictions on the system matrices or inter-level operators. However, with a bounded number of smoothing iterates per level and diminishing gain, more complete results may yet be obtained. We demonstrate that for the following variant of the algorithm.

5.2 The SimMGTD(λ) algorithm

We next consider a variant of the last algorithm which proceeds simultaneously at all levels, thereby eliminating the requirement for data reuse. Moreover, for this variant a convergence analysis of overall algorithm is provided.

The algorithm is shown in the next table. To highlight the similarity with the previous algorithm is has been written in recursive from, but can easily be written more explicitly as will be seen shortly. The SimMGTD(λ) algorithm has the following distinctive features:

1. All parameters except θ_0 are reset to 0 before each TD(λ) iteration.
2. The same temporal difference signal d_0 is used at all levels.

Thus, the temporal difference updates at all levels level are carried out simultaneously and instantaneously, and no coarse-level parameters need to be retained for the next iteration. As before, the value function estimate is $\mathrm{v}(s) = \phi_0(s)^T \theta_0$.

Table 2: Simultaneous Multigrid TD(λ)

```
A. Basic loop:
    1. Initialize $\theta_0$, $\mathbf{z}_\ell := 0$ for all $\ell$
    2. Observe the transition $s_t \to s_{t+1}$ and the reward $g_t$ at time $t$.
    3. Calculate the level-0 temporal difference:
           $d_0 := g_t - \left(\phi_0(s_t) - \gamma\phi_0(s_{t+1})\right)^T \theta_0$
    4. Update $\theta_0$: $\theta_0 := \mathrm{MGTD}(\theta_0, \ell = 0)$
    5. $t := t+1$; goto A2.

B. MGTD($\theta_\ell^0, \ell$)   (recursive function)
    1. Update eligibility trace:   $\mathbf{z}_\ell := \lambda\gamma\mathbf{z}_\ell + \phi_\ell(s_t)$
    2. TD(λ) iteration:   $\theta_\ell := \theta_\ell + \alpha_{\ell,t}\mathbf{z}_\ell d_0$
    3. Coarse grid correction:
        3.1. Recursive call:
                 $\theta_{\ell+1} := \mathrm{MGTD}\left(\theta_{\ell+1}^0 = \mathbf{0}, \ell+1\right)$
        3.2. Correction:   $\theta_\ell := \theta_\ell + \mathbf{I}_{\ell+1}^\ell \theta_{\ell+1}$
    4. Return $\theta_\ell$.
```

An equivalent form of the algorithm is given next.

Lemma 1. *The SimMGTD(λ) algorithm is equivalent to the following iteration*

$$\theta_0 := \theta_0 + \mathbf{\Lambda}_t \mathbf{z}_0 d_0$$

where $\mathbf{z}_0, d_0$ *are as defined in Table 2, and*

$$\mathbf{\Lambda}_t = \alpha_{0,t}\mathbf{I}_0 + \sum\nolimits_{\ell=1}^{\ell_{\max}} \alpha_{\ell,t}\mathbf{I}_\ell^0(\mathbf{I}_\ell^0)^T .$$

Here $\mathbf{I}_0$ *is the* $n_0 \times n_0$ *unit matrix, and* $\mathbf{I}_\ell^0 = \mathbf{I}_1^0\mathbf{I}_2^1 \cdots \mathbf{I}_\ell^{\ell-1}$.

This claim is easily verified by direct algebra, after noting that the definition of $\phi_\ell(s)$ in (4) together with the definition of $\mathbf{z}_\ell$ imply that $\mathbf{z}_{\ell+1} = (\mathbf{I}_{\ell+1}^\ell)^T \mathbf{z}_\ell$.

We note that this equivalent form not efficient for the purpose of implementation, as $\mathbf{\Lambda}_t$ is an $n_0 \times n_0$ matrix. However, its does shed some light on the effect of the coarse grid corrections in this algorithm, which is equivalent to modulating the eligibility trace $\mathbf{z}_0$ by the (positive definite) matrix $\mathbf{\Lambda}_t$. Moreover, this form will leads us to the main convergence result of this section.

Theorem 2. *Assume that the SimMGTD(λ) algorithm is implemented with proportional gains, namely*

$$\alpha_{\ell,t} = \beta_\ell \alpha_t$$

for some non-negative constant β_ℓ, *with* $\beta_0 > 0$. *Assume further that the conditions of Theorem 1 hold with respect to the level-0 basis functions* $\phi(s) = \phi_0(s)$ *and the above gain factors* α_t. *Then* θ_ℓ *converges (w.p. 1) to the same limit point as that the standard TD(λ) algorithm at the finest level, namely to the solution of equation (3).*

Proof (outline). For the assumed gains, it follows from Lemma 1 that the algorithm is described by $\theta_0 := \theta_0 + \alpha_t\mathbf{C}\mathbf{z}_0 d_0$, where $\mathbf{C} = \beta_0\mathbf{I}_0 + \sum\nolimits_{\ell=1}^{\ell_{\max}} \beta_\ell\mathbf{I}_\ell^0(\mathbf{I}_\ell^0)^T$, a symmetric positive definite matrix. Following the analysis of Tsitsiklis & Van Roy (1997), it may be established that the algorithm asymptotically follows the trajectories of the ODE $\dot\theta = -\mathbf{C}\mathbf{A}_0\theta + \mathbf{C}\mathbf{b}_0$, where $\mathbf{A}_0$ is known to be a Hurwitz-stable matrix. It is now easily established (using a Lyapunov equation argument) that stability of $\mathbf{A}_0$ together with positive-definiteness of the symmetric matrix $\mathbf{C}$ imply stability of $\mathbf{C}\mathbf{A}_0$, which implies stability of the ODE and its convergence to the its equilibrium point $\theta^* = (\mathbf{C}\mathbf{A}_0)^{-1}\mathbf{C}\mathbf{b}_0 = (\mathbf{A}_0)^{-1}\mathbf{b}_0$. This, in turn, implies the convergence (w.p. 1) of the learning algorithm to the same point. □

6. An Illustrative Simulation Experiment

We next present some simulation experiments, which are meant to illustrate in an idealized problem setting the potential benefits of the proposed algorithms. The test-bed problem we consider is a 1-D random walk described in Figure 1. This Markov chain has N states, ordered on a 1-D line. Transition probabilities and rewards from inner states and edge states are defined in the figure, and a discount factor of $\gamma = \sqrt[N-1]{0.5}$ is chosen, so that the effective discount factor for a complete sweep of the state is space 0.5. This problem is similar to the hop-world problem in Xu et. al. (2002), and should provide favorable conditions for performance improvement by multigrid methods, due to the local and linear nature of the transition structure.

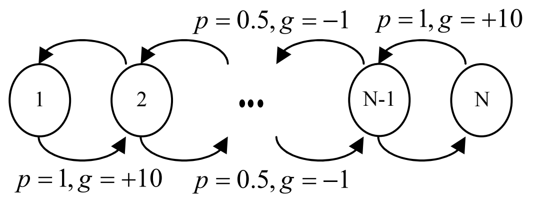

Figure 1 . The 1-D random walk problem

In the setup phase of AMG we used two interpolation methods, as described in Section 2: the Ruge-Stüben method and state aggregation. In Figure 2 we show results for the (non-learning) value iteration scheme of Section 4. One computational unit equals the number of mathematical operations required for a single sweep of standard value iteration. The computational effort required to reach a residual error norm of 10^{-10} was 38203 for standard value iteration, 19103 for the Gauss Seidel variant, 1174 for AMG with state aggregation, and 23 for AMG with Ruge-Stüben interpolation.

We next consider the Multigrid learning algorithms for the same problem, this time with 256 states. The trivial basis functions were used in the first level (namely $\Phi_0 = \mathbf{I}$). For the purpose of the setup phase the full model was made available, Ruge-Stüben interpolation was employed. We used TD(0) at all levels (including the coarsest one) and a constant gain of $\alpha = 0.1$ throughout. In SeqMGTD, we switched levels every 5000 samples (which accounts for the periodic ripple of the corresponding graph). The norm of the error in the parameter vector (relative to its target value) is plotted in Figure 3 as a function of the number of iterations. The number of iterations (in thousands) required for reducing the error norm by half is 654 for standard TD, 76 for SeqMGTD, and 62 for simMGTD.

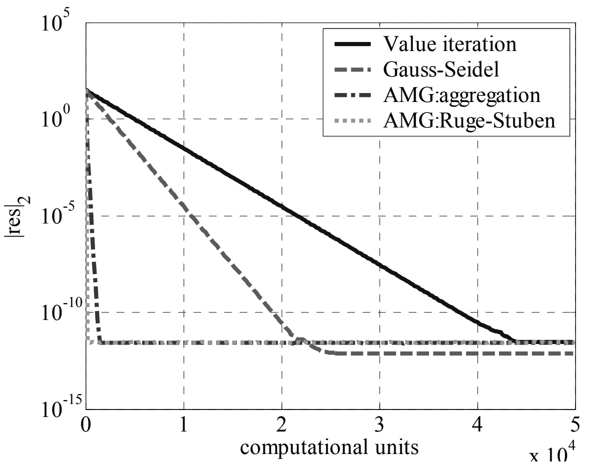

Figure 2. Convergence curves for the 1-D random walk problem with 1000 states. The AMG methods use 6 grid levels, with one pre- and post-smoothing iteration for seqMGTD.

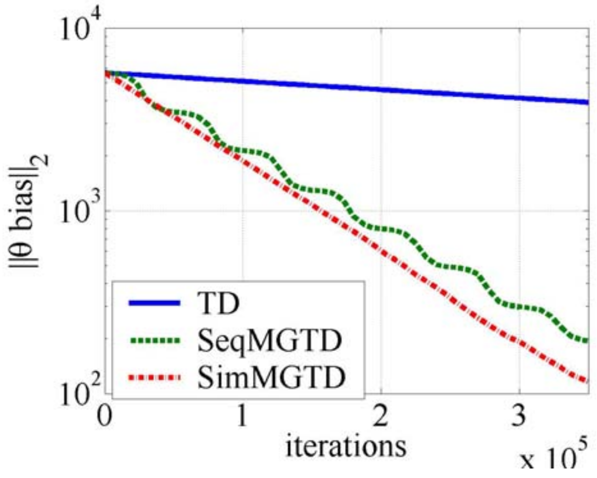

Figure 3. Learning curves for the random walk problem with 256 states. AMG methods use 6 grid levels. Each curve is an average of 5 Monte-Carlo runs..

In both cases, AMG shows at least order of magnitude improvement relative to standard iteration methods.

7. Concluding Remarks

Multigrid is currently a major tools major tool in computational mathematics for speeding up the convergence of iterative methods. As such, its interaction with dynamic programming, and with RL in particular, seems natural. In this paper we have outlined some specific ways in which Multigrid might be combined with temporal difference learning, in order to speed up its convergence.

Several issues remain concerning the possible application of the proposed algorithms. A central question is how to set up an effective multigrid hierarchy, namely the coarse level equations and inter-level operators, especially when the model is unknown. In many cases the state space possesses enough structure, for example geometric, to guide a reasonable selection of state aggregates. A notable feature of AMG is the automatic creation of the multigrid hierarchy at the setup phase, even when no such structure is available. In the known model case, the implementation of a setup phase of AMG is straightforward, with many methods available in AMG literature. In the learning case, the required data on the relevant system matrix may not be available beforehand. One way to obtain the required information is to compute an estimate of the matrix $\mathbf{A}_0$, as carried out in the LSTD algorithm (Boyan, 2002). Clearly this might be an expensive process when this matrix is large; however, a crude or even qualitative estimate should suffice for the setup phase. An interesting alternative would be to develop methods that form effective aggregates directly from the observed state process, based on the observed temporal proximity between states, and relying on the AMG guidelines for relating states that have strong connections in the (effective) transition matrix (see Kretchmar & Anderson, 1999, for some related ideas).

Other issues concern the optimization of various parts the algorithms themselves, such as the choice of setup method, relative gains at the different levels, and switching rules between levels. Clearly, additional experimental work is required to evaluate these issues, as well as the overall efficacy of these algorithms. Further theoretical results concerning convergence and performance bounds are equally of interest.

From another perspective, one should note that AMG is a bottom up approach which builds coarse basis from finer ones. Applying this process to the full state space, for example, may lead to a scheme for constructing effective basis functions, based on the considerable theoretical and practical insight of AMG research.

Finally, we point out the result in Lemma 1, which implies that the coarse grid corrections in the SimMGTD algorithm are equivalent to a certain modification of the eligibility traces at the basic (fine) level. This might point to other possibilities for accelerating TD(λ) (and related algorithms) by modifying the eligibility trace, an issue which seems worthy of further study.

Acknowledgments

We thank Irad Yavne for his invaluable guidance and advice on Multigrid methods.

References

Barto, A.G., and Mahadevan, S. (2003). Recent advances in hierarchical reinforcement learning. *Discrete Event Dynamic Systems: Theory and Applications*, *13,* 41-77.

Bertsekas, D. P., Borkar, V., & Nedić, A. (2004). Improved temporal difference methods with linear function approximation. In J. Si, A. G. Barto, W. B. Powell and D. Wunsch (Eds.), *Learning and Approximate Dynamic Programming*, Wiley-IEEE Press, 2004.

Bertsekas, D. P., & Castañon, D.A. (1989). Adaptive aggregation methods for infinite horizon dynamic programming. *IEEE Trans. Automat. Contr.*, *34,* 589-598.

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

Boutillier, C., Dean, T., & Hanks, S. (1999). Decision-theoretic planning: structural assumptions and computational leverage. *Journal of Artificial Intelligence Research*, *11,* 1-94.

Boyan, J.A. (2002). Technical update: least squares temporal difference learning. *Machine Learning*, *49,* 233-246.

Briggs, W. L., Henson V. E., & McCormick, S. F. (2000), A Multigrid Tutorial, 2nd Edition. Philadelphia, MA: Siam.

Brandt A, McCormick, S. F., & Ruge, J. (1984). Algebraic multigrid (AMG) for sparse matrix equations. In: D. J. Evans (Ed.), *Sparsity and it Applications,* Cambridge University Press, pp. 257-284.

Kretchmar, R.M., and Anderson, C.W. (1999). Using temporal neighborhoods to adapt function approximators in Reinforcement Learning. *Proceedings of the International Work Conference on Artificial and Natural Neural Networks* (IWANN), pp. 488-496, Alicante, Spain.

Nedić, A., & Bertsekas, D.P. (2003). Least squares policy evaluation algorithms with linear function approximation. *Discrete Event Dynamic Systems: Theory and Applications*, *13,* 79-110.

Pareigis, S. (1997). Multi-grid methods for reinforcement learning in controlled diffusion processes. *Proceedings of NIPS'96: Advances in Neural Information Processing Systems 9* (pp. 1033-1039). MIT Press.

Shweitzer, P. J. (1991). A survey of aggregation-disaggregation in large Markov chains. In: W. Stewart (Ed.), *Numerical Solution of Markov Chains*, New York: Marcel Dekker, New York.

Schweitzer, P. J., Puterman, M. L., & Kindle, K.W. (1985). Iterative aggregation-disaggregation procedures for discounted semi-Markov reward processes. *Operations Research*, *33(3),* 589-605.

Stüben, K. (2001). An Introduction to Algebraic Multigrid. Appendix A in Trottenberg et. al. (2001).

Sutton, R.S. (1988). Learning to predict by the methods of temporal differences. *Machine Learning*, *3,* 9-44.

Sutton, R.S., & Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA: MIT Press.

Trottenberg U., Oosterlee C., and Schüller A. (2001). *Multigrid*. San Diego: Academic Press.

Tsitsiklis, J. N., & Van Roy, B.(1997). An analysis of temporal-difference learning with function approximation. *IEEE Transactions on Automatic Control, 42(5),* 674-690.

Xu, X., He, H., & Hu, D. (2002). Efficient reinforcement learning using recursive least-squares methods. *Journal of Artificial Intelligence Research*, *16,* 259-292.