A Multigrid Form of Value Iteration Applied to a Markov Decision Problem

Robert B. Heckendorn
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
heckendo@cs.colostate.edu

Charles W. Anderson
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
anderson@cs.colostate.edu

November 12, 1998

Technical Report CS-98-113

A Multigrid Form of Value Iteration Applied to a Markov Decision Problem

Robert B. Heckendorn
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
heckendo@cs.colostate.edu

Charles W. Anderson
Department of Computer Science
Colorado State University
Fort Collins, CO 80523
anderson@cs.colostate.edu

November 12, 1998

Category: Control, Navigation, and Planning: reinforcement learning.
No part of this work has been submitted or appeared in other scientific conferences.

Abstract

We explore the use of multigrid techniques to speed the convergence of value iteration for Markov Decision Problems by applying it to the mountain car problem. We look at some of the fundamental differences between these kinds of problems and those traditionally treated with multigrid methods. We demonstrate that significant performance improvements can be achieved by using these techniques.

1 INTRODUCTION

Chow and Tsitsiklis (1991) define a multigrid version of the value iteration, or successive approximation, algorithm for discounted, infinite horizon Markov Decision Problems (MDP). The multigrid version is performed by discretizing the continuous state and action spaces of an MDP at different granularities. First, the solution of the MDP is approximated using value iteration at a coarse level. The resulting approximation is used to initialize the value iteration procedure at the next finer level. This coarse-to-fine progression is repeated until the solution at a fine granularity forms a satisfactory approximation to the solution of the MDP. Chow and Tsitsiklis show that the computational complexity of their multigrid algorithm is within a factor of $\mathrm{O}\left(\frac{1}{1-\gamma}\right)$ of the complexity of finding the optimal solution of an MDP (Chow and Tsitsiklis, 1989), where γ is the discount factor. They also show that when an ergodicity condition holds, their algorithm is within a constant factor of optimal.

Multigrid methods (Briggs, 1987; McCormick, 1992; Rüde 1993) have been applied to a variety of physical boundary value problems where the problem solution can be formulated as an iteration to convergence over a lattice. These problems include linear systems, finite element analysis networks, algebraic and differential eigenvalue problems, and image processing, to name a few (Briggs, 1987). To speed up the transfer of information and convergence, the problem is first solved to some degree of error on a coarse grid. This creates a matrix with far fewer points and much smaller diameter. Then the grid size is doubled and the values interpolated from the coarse grid to the fine grid. Multigrid, in its most elementary form, is simply a loop of relaxations (iteration) and interpolations to finer and finer grid sizes[1]. Although this is not always guaranteed to converge faster, it often can be tuned to do so.

A large class of reinforcement learning (RL) algorithms use one or more matrices of values. For example: Q learning, SARSA, and TD(λ) and Value Iteration. These algorithms all have some form of iteration over a matrix to converge to a solution. They also exhibit the same problems as with the relaxation scheme above in that the information must flow across the matrix from the boundaries (termination states) and that the matrix must be given time to stabilize. Although the multigrid approach appears very similar to matrix iteration used in reinforcement learning algorithms, there are some important distinctions.

A nice feature of PDE-like problems is that the size and shape of a point remains the same regardless of grid size. For example, at a particular point in a grid for a heat transfer problem, the point refers to the temperature at that exact point regardless of the grid in which it is embedded. In a reinforcement learning matrix the value refers to the average across an area where the area changes with grid size. For example, each cell may be divided into four subcells occupying different regions of the parameter space. The area and scaling properties imply that the interpolation operator should be constructed differently for RL applications. These properties also imply the values in a converged solution for a coarse grain matrix in a RL application may not be the same as values for the fine grain solution. Another surprising difference between RL methods and multigrid relaxation methods is locality. Classical relaxation methods generally assume each point is connected to its neighbors in an orderly orthogonal grid[2]. In RL matrices, the value in one cell is related to the cells that contain neighboring states and are not necessarily neighboring cells numerically.

Here we implement a multigrid, value iteration algorithm and apply it to the mountain car problem (Sutton, 1996). Our implementation differs from Chow and Tsitsiklis's formulation in a number of ways, some of which are described in the following section. Our implementation is similar to their formulation in that

[1]This is a gross simplification of the vast set of multigrid algorithms which often include relaxing on error terms as well as the original problem and may include both refining and coarsening steps.

[2]Again, multigrid methods allow for a wide variety of grid patterns which may include the composition of grids of varying granularity, however strict assumptions are usually made about the distance metrics for the space.

we discretize the continuous state space in a one-way, coarse-to-fine, schedule. Our results show that the multigrid approach to value iteration converges to the correct value function much faster than the conventional, single-grid approach.

In Section 2, we define our multigrid, value iteration algorithm. Section 3 describes experiments with the mountain car problem and summarizes our current results. Section 4 states our conclusions and outlines on-going work.

2 MULTIGRID VALUE ITERATION

Assume that a continuous state space is discretized equally along each dimension resulting in a multi-dimensional grid. Let δ be the spacing of grid, in each dimension, relative to the finest grid; for the finest grid, $\delta = 1$, for the next coarser grid $\delta = 2$, etc. We define $S^{\delta+1}$, the discretization of the state space for level $\delta + 1$, as a function, f, of the discretization at level δ:

$$f : S^\delta \rightarrow S^{\delta+1}, \qquad f(s^\delta) = s^{\delta+1}.$$

The discretization of the action space is defined similarly:

$$g : A^\delta \rightarrow A^{\delta+1}, \qquad g(a^\delta) = a^{\delta+1}.$$

The probability of visiting state t^δ after taking action a^δ while in state s^δ is represented by $P^\delta_{s^\delta a^\delta t^\delta}$. The state transition probabilities for one level depend on the probabilities at the next finer level in the following way. Let $S^\delta_{\delta+1}$ be the set of state, action, next state triples at level δ that correspond to a given triple at level $\delta + 1$. Specifically,

$$S^\delta_{\delta+1} = \left\{(s^\delta, a^\delta, t^\delta) | f(s^\delta) = s^{\delta+1}, g(a^\delta) = a^{\delta+1}, f(t^\delta) = t^{\delta+1}\right\}.$$

The transition probabilities at a coarser resolution are defined in terms of the probabilities at the finer resolution by

$$P^{\delta+1}_{s^{\delta+1} a^{\delta+1} t^{\delta+1}} = \frac{1}{|S^\delta_{\delta+1}|} \sum_{(s^\delta, a^\delta, t^\delta) \in S^\delta_{\delta+1}} P^\delta_{s^\delta a^\delta t^\delta}.$$

To complete the definition of an MDP discretized at level δ, we define the expected reward given an action a^δ in state s^δ as $R^\delta_{s^\delta a^\delta}$. Now we can define the multigrid version of value iteration in which states are updated asynchronously by the following update procedure:

$$V^\delta_{s^\delta}(k+1) = \max_{a^\delta \in A^\delta} \{R^\delta_{s^\delta a^\delta} + \gamma^\delta \sum_{t^\delta \in S^\delta} P^\delta_{s^\delta a^\delta t^\delta} V^\delta_{t^\delta}(k)\},$$

where $0 < \gamma^\delta \leq 1$ is a discount factor.

The value iteration update procedure is first performed at the coarsest level. After some criterion is met, such as number of iterations or minimum amount of change in the value function, the current estimate of the value function at the coarse level is used to initialize the value function at the next finer level and value iteration is performed at that level. The way in which the finer-level value function is initialized depends on the problem being solved.

3 APPLICATION TO THE MOUNTAIN CAR PROBLEM

In this section the general multigrid form of value iteration is specialized to the mountain car problem (Sutton, 1996) in which a car must be driven out of a two-dimensional valley by rocking it back and forth. The state space is two-dimensional, consisting of the car's position and velocity, and the available actions are a fixed magnitude acceleration forward and backward and a zero acceleration coasting action. Rewards are deterministic—a reward of -1 is given for all actions except those that cause the car to leave the valley at its most positive position, for which a reward of 0 is given.

We applied conventional value iteration to the mountain car problem after discretizing its state space into a grid of 32 x 32 states. For each iteration, all states were updated, starting with states for which the car's position and velocity are most positive and ending with states for which the car's position and velocity are most negative. Value iteration continued until the maximum value change over all states during one iteration was less than 10^{-6}.

To apply multigrid value iteration to the mountain car problem, we specialized the algorithm in the following ways. The same action space, consisting of three actions, is used at all levels. The discount factor, γ, is set to 1 for all levels. Finally, we must specify how the value function at one level is used to initialize the value function at the next finer level. We implemented a simple, linear interpolation scheme to do this. Let the discrete state at level δ be indexed by i^δ and j^δ whose values range from zero to one less than the number of intervals along the respective dimensions at level δ. The interpolation is given by

$$\begin{aligned}
a &= \text{floor}(i^{\delta+1}/2), \\
b &= \text{floor}(j^{\delta+1}/2), \\
c &= 2i^{\delta+1} - 3a - 1, \\
d &= 2j^{\delta+1} - 3b - 1, \\
V^{\delta}_{i^\delta j^\delta} &= \frac{1}{16}(9V^{\delta+1}_{ab} + 3V^{\delta+1}_{ad} + 3V^{\delta+1}_{cb} + V^{\delta+1}_{cd}),
\end{aligned}$$

for states other than those along the border of the δ-level grid. Those in the corners are determined by one value at the coarser level, while those along the edges are determined by a linear interpolation between two values at the coarser level.

Value iteration at a given level was terminated after a fixed number of iterations. Several iteration limits were tried. In the next section, we report results from limiting iterations at each level to 4, 8, 16, 32, and 64 at each level. The multigrid procedure started at a granularity of 2 x 2 and at each level the size of the grid was doubled, resulting in four times as many discrete states. So, with an iteration limit of 4, 4 iterations of value iteration were performed at a granularity of 2 x 2, then 4 were performed at 4 x 4, then 4 at 8 x 8, then 4 at 16 x 16, and, finally, iterations were performed at 32 x 32 until the maximum value change was less than 10^{-6}.

Several measures of performance were used to compare the results of multigrid value iteration at the different iteration limits and conventional value iteration. Note that some measures are taken at different granularities and plotted on the same axes. Strictly speaking one cannot directly compare value iteration at different granularities. To have truly valid comparisons, the reader should only look at the data when algorithms are working on the finest grid size. However, we believe that the values for coarse grids in the multigrid algorithm are indicative of the degree of convergence. The relative smoothness of the measures across interpolation steps supports this.

One performance measure is the *average steps to goal* that measures the quality of the greedy policy derived from the current estimate of the value function. The mountain car state was initialized to the center of each grid cell at a given granularity level and then driven with actions at each step chosen by a greedy policy using the current value function. Each driving sequence was terminated when the car escaped the valley or a maximum number of steps occurred. The number of steps from each initial state was averaged to obtain the average number of steps to goal. The smaller the average, the better the policy. For coarse grids this measure is not being applied to a grid of the final grid size, hence this measure is only a predictor of the steps to goal in the final grid size. The number of steps to goal is not dependent on the granularity of the grid, because the same action granularity is used at all levels.

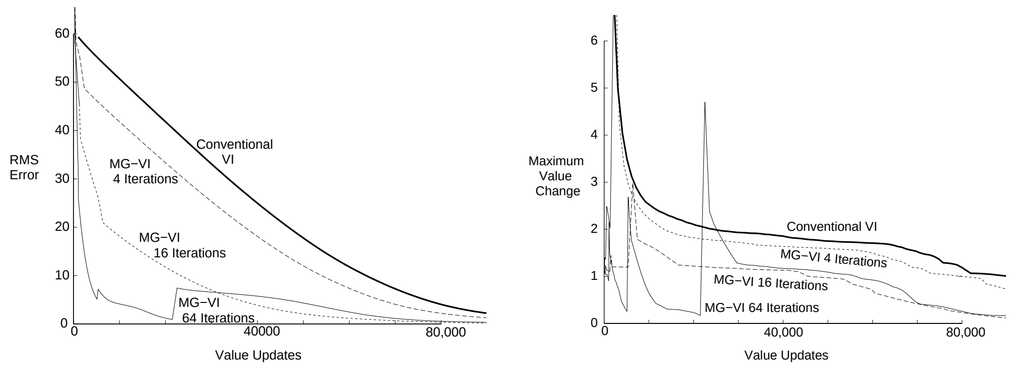
The second performance measure is the *RMS error* in the value function with respect to the optimum value function. We obtained an estimate of the optimal value function by applying conventional value iteration until the maximal change of any one cell in an iteration was less than 10^{-6}. The smaller the RMS error, the closer the current value function is to the optimal one. The RMS value can only be computed for grids of the same size. The value function at coarser grids was compared against the ideal value function only at the coarser granularity.

The third performance measure is the *maximum change* to any value during an iteration. Many values change much less than the maximum value in any given iteration.

All three performance measures were recorded with respect to the number of *value updates* as a measure of execution time. For example, one iteration of a 10 x 10 grid results in 100 value updates.

4 RESULTS

The plots in Figure 1a of RMS error between estimated and optimal value functions show that the performance of multigrid value iteration is superior to conventional value iteration for all tested iteration limits. The gains made and preserved in the first 2,500 value updates are striking. The large positive jumps in RMS error for the multigrid curves coincide with the change from one granularity level to the next finer one. These jumps are more clear in Figure 1b, where the logarithm of the maximum value change in an iteration is plotted versus the number of value updates. The interpolation points at which the multigrid algorithm switches from one granularity level to the next finer one cause spikes in the maximum change measure. After each spike, the maximum change value quickly recovers to be lower than that of conventional value iteration and remains so down to an error of 10^{-6}.



a. RMS error versus value updates. b. Maximum change versus value updates.

Figure 1: RMS error and maximum change versus value updates for conventional value iteration (VI) and our multigrid value iteration algorithm (MG-VI) using 4, 16, and 64 iterations per level.

The average steps to goal is the performance measure most relevant to the behavior of the mountain car. In Figure 2, the average steps to goal is plotted with respect to value updates. An iteration limit of 16 or more results in a quick, large reduction in steps to goal. All runs achieved nearly equivalent policies, ending with an average number of steps to goal of approximately 60. However, multigrid value interation achieved this performance level about 8 times faster than did conventional value iteration.

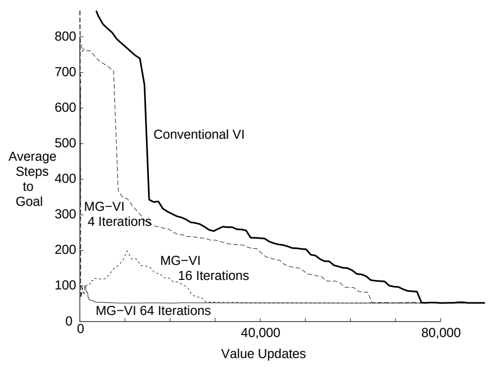


Figure 2: Steps to goal versus value updates.

5 CONCLUSIONS

The multigrid value iteration algorithm studied here follows a one-way, coarse-to-fine, schedule of discretization levels. Multi-way schedules for shifting between granularities can yield faster convergence when solving systems of partial differential equations with boundary conditions. We plan to investigate such multi-way schedules for multigrid value iteration.

Future research will also be focused on tuning the interpolation points and automating when to interpolate. Various interpolation schemes should be investigated including one that would anticipate changes in policy.

The practicality of value iteration is strongly limited by its requirement of known state transition probabilities. Since reinforcement learning methods embody a Monte Carlo approach to dynamic programming that do not require transition probabilities, the integration of multigrid techniques and reinforcement learning has the potential of being a very general and efficient approach to solving MDPs. We plan to continue the research into multigrid, Monte Carlo approaches initiated by Anderson and Crawford-Hines (1994).

In the long term, the vast literature of multigrid methods should be explored and adapted to the reinforcement learning field. If these preliminary results are any indication, the merging of these two fields should be quite fruitful.

References

Anderson, C. W., and Crawford-Hines, S. G. (1994) Multigrid Q-learning. Technical Report CS-94-121, *Colorado State University*, Fort Collins, CO 80523.

Briggs, W. (1987) A multigrid tutorial. *Society for Industrial and Applied Mathematics*, Lancaster Press, 1987.

Chow, C.-S. and Tsitsiklis, J. N. (1991) An optimal one-way multigrid algorithm for discrete time stochastic control, *IEEE Transactions on Automatic Control*, August 1991, vol. 36, no. 8, pp. 898–914.

Chow, C.-S. and Tsitsiklis, J. N. (1989) The complexity of dynamic programming, *Journal of Complexity*, vol. 5, pp. 466–488.

Dayan, P. and Hinton, G. E. (1993) Feudal reinforcement learning, *Neural Information Processing Systems 5*, Morgan Kaufmann Publishers, pp. 271–278.

Moore, A.W. (1993) The parti-game algorithm for variable resolution reinforcement learning in multidimensional state-spaces. In *Advances in Neural Information Processing Systems 6.* Morgan Kaufmann, pp. 711–718.

Moore, A.W. (1991) Variable resolution dynamic programming: Efficiently learning action maps in multivariable real-valued state-spaces. In *Proceedings of the Eighth International Workshop on Machine Learning*, Morgan Kaufmann, pp. 333-337.

McCormick, S. F. (1992) Multilevel Projection Methods for Partial Differential Equations, *Society for Industrial and Applied Mathematics*, Capital City Press.

Rüde, U. (1993) Mathematical and computational techniques for multilevel adaptive methods, *Society for Industrial and Applied Mathematics.*

Sutton, R.S. (1996) Generalization in reinforcement learning: Successful examples using sparse coarse coding. In *Advances in Neural Information Processing Systems 8*, Cambridge, MA: MIT Press.