Fast Direct Policy Evaluation using Multiscale Analysis of Markov Diffusion Processes

Mauro Maggioni
Program in Applied Mathematics
Department of Mathematics
Yale University
New Haven, CT 06511
mauro.maggioni@yale.edu

Sridhar Mahadevan
Department of Computer Science
University of Massachusetts
Amherst, MA 01003
mahadeva@cs.umass.edu

June 9, 2005

Technical Report 2005-39
Department of Computer Science
140 Governors Drive
University of Massachusetts
Amherst, Massachusetts 01003-9624

Abstract

Policy evaluation is a critical step in the approximate solution of large Markov decision processes (MDPs), typically requiring $O(|S|^3)$ to directly solve the Bellman system of $|S|$ linear equations (where $|S|$ is the state space size). In this paper we apply a recently introduced multiscale framework for analysis on graphs to design a faster algorithm for policy evaluation. For a fixed policy π, this framework efficiently constructs a multiscale decomposition of the random walk P^π associated with the policy π. This enables efficiently computing medium and long term state distributions, approximation of value functions, and the *direct* computation of the potential operator $(I - \gamma P^\pi)^{-1}$ needed to solve Bellman's equation. We show that even a preliminary non-optimized version of the solver competes with highly optimized iterative techniques, and can be computed in time $O(|S| \log^2 |S|)$.

Keywords: Markov Decision Processes, Reinforcement learning, Spectral Graph Theory, Harmonic Analysis, Riemannian Manifolds.

1 Introduction

In this paper we apply a novel framework for multiscale analysis of Markov diffusion processes on graphs to efficiently solving certain classes of Markov Decision Processes (MDPs). The approach is based on learning a multiscale tree of *wavelet*-type basis functions on the state space of a MDP, which allows efficient hierarchical representation of value functions, and yields a fast algorithm for the direct solution of the Bellman equation for policy evaluation. The paper focuses on policy evaluation primarily, and the resulting method can be easily incorporated in approximate or exact policy iteration (PI) algorithms, such as LSPI [6] and RPI [7].

Bellman's equation usually involves the solution of a sparse linear system of size $|S|$, where S is the state space. A classical direct solution of the system is infeasible for large problem sizes, since it requires $\mathcal{O}(|S|^3)$ steps. One common technique is to use an iterative method, such as value iteration, which has worst case complexity $\mathcal{O}(|S|^2)$ for sparse transition matrices, $\mathcal{O}(|S| \log |S|)$ when the problem is well-conditioned and only low-precision is required. The approach in this paper is fundamentally different, and yields a *direct* solution with the per-step efficiency of value iteration in time $\mathcal{O}(|S| \log^2 |S|)$. It consists of two parts:

(i) a *pre-computation* step, which depends on the structure of the state space and on the policy. The result of this step is a multiscale hierarchical decomposition of the set of all value functions over the state space, and a multiscale compression of powers of the transition matrix (random walk operator) over the state space. This computation, for many classes of problems of interest in applications, has complexity $\mathcal{O}(|S| \log^2 |S|)$.

(ii) an *inversion* step, which uses the multiscale structure resulting from the "pre-computation" step to efficiently compute the solution of Bellman's equations for a given reward function. This phase of the computation has complexity $\mathcal{O}(|S| \log |S|)$ for many problems of practical importance where the transition matrix is *diffusion*-like (defined precisely below). The constants in front of this asymptotic complexity are much smaller than those in the pre-computation step.

We will define the class of problems for which the complexity of our method is linear up to logarithmic factors. Qualitatively, this class includes the case of state spaces that can be represented by a finite undirected weighted graph, with all the vertices of "small" degree in which transitions are allowed only among neighboring points, and the spectrum of the transition matrix decays fast enough. The direct method we present offers several advantages.

(i) The multiscale construction allows efficient approximation of reward and value functions, which is an important task *per se* [6, 7].

(ii) It is well-known that the number of iterations necessary for an iterative method to converge can be very large, depending on the condition number of the problem (which in general depends on the number of points), and on the precision required. Increasing precision in the direct inversion technique we propose can be done more efficiently. In this context we will see that even a simple, non-optimized implementation of our scheme outperforms standard iterative solvers.

(iii) When the state space and the policy are fixed, and many value functions corresponding to different rewards (tasks) need to be computed, iteration schemes do not take advantage of the common structure between the problems. In this case, the number of iterations for finding each solution is multiplied by the number of solutions sought. Our direct inversion technique efficiently encodes the common structure of the state space in the pre-computation step, and then takes advantage of this in the solution of multiple problems.

A key advantage of the proposed approach is that direct inversion reveals interesting structure in the underlying problem. The multiresolution construction has interesting connections to work on hierarchical reinforcement learning [1].

2 Policy Evaluation

A finite Markov decision process (MDP) $M = (S, A, P^a_{ss'}, R^a_{ss'})$ is defined by a finite set of states S, a finite set of actions A, a transition model $P^a_{ss'}$ specifying the distribution over future states s' when an action a is performed in state s, and a corresponding reward model $R^a_{ss'}$ specifying a scalar cost or reward [9]. A value function is a mapping $S \to \mathcal{R}$ or equivalently a vector in $\mathcal{R}^{|S|}$. Given a policy $\pi : S \to A$ mapping states to actions, its corresponding value function V^π specifies the expected long-term discounted sum of rewards received by the agent in any given state s when actions are chosen using the policy. This paper focuses on policy evaluation since this is usually the most expensive step in policy iteration requiring the inversion of the transition matrix. The second phase of policy improvement requires computing the greedy policy, and is of lower complexity. Policy evaluation consists of solving the (Bellman) linear system of equations

$$V^\pi(s) = \sum_{s'} P^{\pi(s)}_{ss'} \left(R^a_{ss'} + \gamma V^\pi(s') \right).$$

Bellman's equation can be written in matrix form

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

where V^π and R are vectors of length $|\mathcal{S}|$, and P^π is a stochastic matrix of size $|\mathcal{S}| \times |\mathcal{S}|$, and $\gamma \in (0, 1]$ is the discount factor. The solution is given by

$$V^\pi = (I - \gamma P^\pi)^{-1} R.$$

The matrix

$$(I - \gamma P^\pi)^{-1}$$

is called the fundamental matrix of the Markov chain P^π (usually when $\gamma = 1$ (non discounted reward)) and also the *Green's function*, for its interpretation in potential theory and physics [5].

3 Multiscale analysis using diffusion wavelets

Diffusion wavelets, introduced in [4, 2], provide a natural hierarchical multiscale analysis of Markov chains and graphs. More specifically, this framework provides a multiscale analysis for functions on a graph, and their efficient analysis, representation, and compression. In the companion paper [8], we apply this approach to value function approximation. The approach provides a hierarchical compression and organization of the graph itself, including compression of large powers of P for efficient and accurate computation of large time behavior of the random walk, and efficient computation of several functions of P and its powers, among which notably the Green's function $(I-P)^{-1}$. By "efficient" we mean that the number of operations required is asymptotically (i.e. for fixed precision, and large $|S|$), of order $\mathcal{O}(|S|\log^2|S|)$.

We describe how to apply the diffusion wavelet framework to MDPs. We assume the state space can be modeled as a finite undirected weighted graph (S, E, W) (our approach generalizes to Riemannian manifolds, which we do not have space to discuss here). If any policy π is executed, it will traverse some subset of the state space $S^\pi \subseteq S$. For simplicity, assume that the MDP is ergodic, meaning that all states are visited infinitely often under every policy. We will write $x \sim y$ when there is an edge between x and y, we let the degree of x to be $d(x) = \sum_{x\sim y} w(x,y)$. We will denote by D the diagonal matrix defined by $D(x,x) = d(x)$, and W the matrix defined by $W(x,y) = w(x,y)$. Let P be the natural random walk defined by $P = D^{-1}W$.

3.1 Setup

The hypotheses on P are that $P^t, t \geq 0$ should be a Markov diffusion process, in a technical sense made precise in [4]. Qualitatively, this means that P should be *local*, i.e. from every point a random walk takes the agent to only a few nearby points; *smoothing*, i.e. $P^t\delta_x$, for any initial condition δ_x, should be a smooth probability cloud centered about x; *contractive*, i.e. $||P||_2 \leq 1$. For our algorithm to have complexity at most $\mathcal{O}(|S|\log^2|S|)$, we need two further assumptions: first, the matrix P should be sparse, in the sense that only about $c|S|$ entries are non-zero, where $c > 0$ is a small constant. This is the case for example for the natural random walk on a graph where the vertices have degree bounded by c. Second, the eigenvalues $\{\lambda_j\}$ of P should decay rapidly, for example

$$\#\{j : \lambda_j \geq \epsilon^{2^{-j}}\} \leq c2^{-j\alpha}\log_2^\alpha(1/\epsilon)$$

for some $\alpha > 0$, and some fixed small $\epsilon > 0$. As an example, it is shown in [4] as a direct application of Weyl's Theorem on the distribution of the eigenvalues of the Laplace-Beltrami operator on a Riemannian manifold, that this condition is satisfied when P is a discretization of the natural random walk on a smooth compact Riemannian manifold of dimension d, in which case one can choose $\alpha = d/2$.

3.2 Qualitative description

Space constraints allow us only a brief description of the construction: we refer the interested reader to [4, 2]. The input to the algorithm is a weighted graph (S, E, W) and a "precision" parameter $\epsilon > 0$.

As above, the natural random walk on the graph is represented $P = D^{-1}W$, and it is necessarily reversible. We symmetrize it by conjugation, and take powers to obtain

$$T^t = D^{\frac{1}{2}} P^t D^{-\frac{1}{2}} = (D^{-\frac{1}{2}} W D^{-\frac{1}{2}})^t = (I - \mathcal{L})^t = \sum_{i \geq 0} (1 - \lambda_i)^t \xi_i(\cdot) \xi_i(\cdot)$$

where $\mathcal{L}$ is the normalized Laplacian [3], $\{\lambda_i\}$ and $\{\xi_i\}$ are its eigenvalues and eigenvectors:

$$\mathcal{L}\xi_i = \lambda_i \xi_i.$$

Hence the eigenfunctions of T_t are again ξ_i and the i^{th} eigenvalue is $(1 - \lambda_i)^t$. We want to construct a multiresolution decomposition of the functions on the graph. This is a family of nested subspaces

$$V_0 \supseteq V_1 \supseteq \ldots \supseteq V_j \supseteq \ldots$$

spanned by orthogonal bases of diffusion scaling functions Φ_j. If we interpret T^t as an operator on functions on the graph, then V_j is defined as the numerical range, up to precision ϵ, of $T^{2^{j+1}-1}$, and the scaling functions are smooth bump functions with some oscillations, at scale roughly 2^{j+1} (measured with respect to geodesic distance). The orthogonal complement of V_{j+1} into V_j is called W_j, and is spanned by a family of orthogonal diffusion wavelets Ψ_j, which are smooth localized oscillatory functions at the same scale.

3.3 A simple example

We consider the Markov chain on 4 states

$$\{a, b, c, d\} : T = \begin{pmatrix} 0.8 & 0.2 & 0 & 0 \\ 0.2 & 0.75 & 0.05 & 0 \\ 0 & 0.05 & 0.75 & 0.2 \\ 0 & 0 & 0.2 & 0.8 \end{pmatrix}.$$

This chain has a "bottleneck" between states $\{a, b\}$ and states $\{c, d\}$. We fix a precision $\epsilon = 10^{-10}$. See Figure 1 for the discussion that follows. The scaling functions Φ_0 are simply $\{\delta_a, \delta_b, \delta_c, \delta_d\}$. We apply T to Φ_0 and orthonormalize to get Φ_1 (Figure 1). Each function in Φ_1 is an "abstract-state", i.e. a linear combination of the original states.

We represent T^2 on Φ_1, to get a matrix T_2, apply to Φ_1 and orthonormalize, and so on. At scale 5 we have the basis Φ_5 and the operator T_5, representing T^{2^5} on Φ_5. At the next level, we obtain Φ_7, which is only two dimensional, because $T_5\Phi_5$ has ϵ-rank 2 instead of 4: of the 4 "abstract-states" $T_5\Phi_5$, only two of them are at least ϵ-independent.

Observe the two scaling functions in Φ_6 are approximately the asymptotic distribution and the function which distinguishes between the two clusters $\{a, b\}$ and $\{c, d\}$. Then T_6 represents T^{2^6} on Φ_7 and is a 2 by 2 matrix. At scale 10, Φ_{10} is one-dimensional, and is simply the top eigenvector of T (represented in compressed form, on the basis Φ_8), and the matrix T_9 is 1 by 1 and is the top eigenvalue, 1, of T.

3.4 Construction of the scaling functions and wavelets

In more detail, we define the finest scale scaling space V_0 as the subspace spanned by $\Phi_0 := \{\delta_x\}_{x \in X}$, where δ_x is the Dirac delta at the point x. We then consider the family

$$\tilde{\Phi}_0 := \{T^{2^0} \delta_x\}_{x \in X}$$

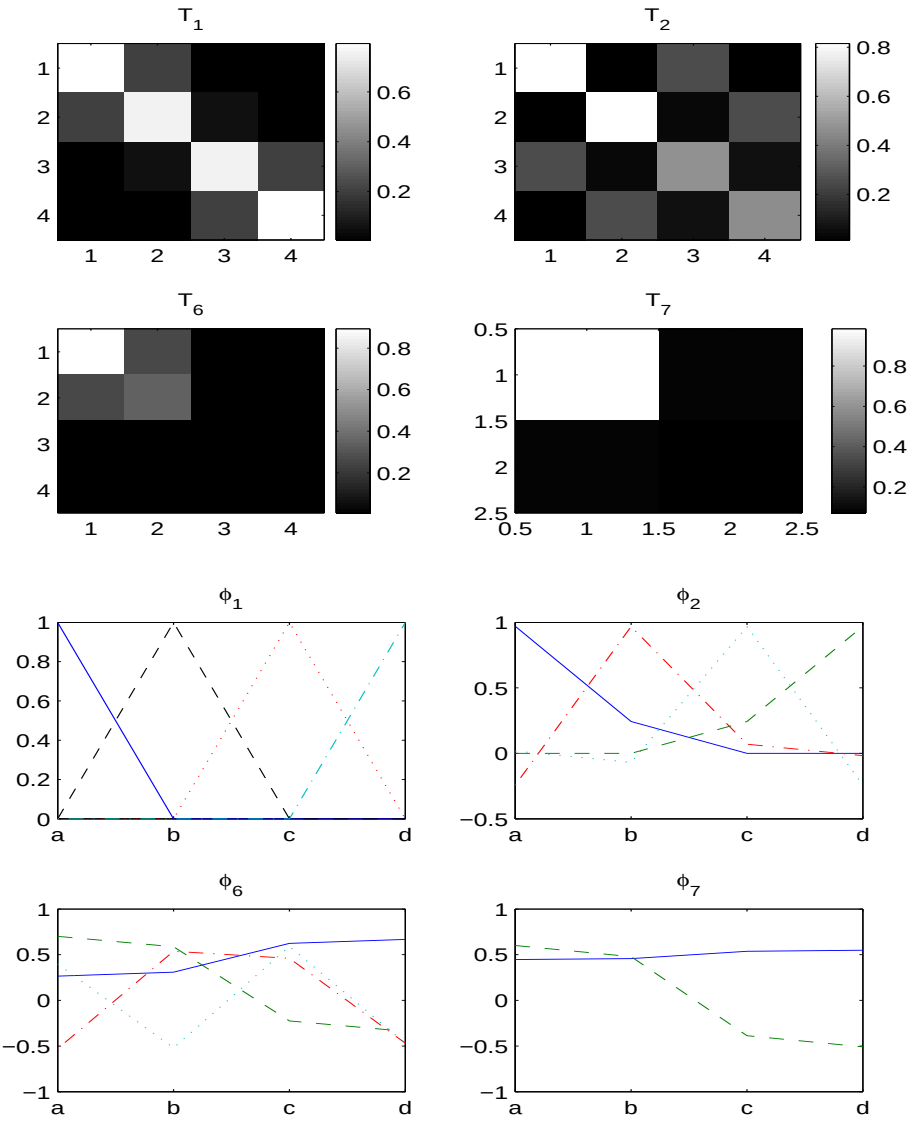

Figure 1: Top: Matrices representing some compressed dyadic powers of T, with gray level representing entry values. Bottom: some scaling function bases on the 4-state Markov chain.

and let $\tilde{V}_1$ be the span of $\tilde{\Phi}_0$. A careful Gram-Schmidt procedure applied to $\tilde{\Phi}_0$ produces an orthonormal basis of well-localized scaling functions Φ_1, spanning a subspace V_1 close to $\tilde{V}_1$ up to the pre-specified precision ϵ. V_1 is also the subspace spanned by

$$\{\xi_i : \lambda_i \geq \epsilon\}.$$

Hence in general $\dim V_1 < \dim V_0$: the orthonormalization will produce only $\dim V_1$ basis functions in Φ_1.

We now represent T^2 on the basis Φ_1 via a matrix T_1, and repeat the construction. At scale j we have constructed an orthonormal basis of localized scaling functions Φ_j, and represented T^{2^j} on this basis via a matrix T_j. Observe that the matrix representing T^{2^j} on this basis has size $|\Phi_j| \times |\Phi_j|$. The assumptions on the decay of the spectrum of T imply that $|\Phi_j| << |\Phi_0|$, and we say that we have represented T^{2^j} in compressed form.

For the next stage we let

$$\tilde{\Phi}_j := T^{2^j} \Phi_j$$

and V_{j+1} be the span of these vectors. After orthonormalization to get a (smaller) orthonormal set Φ_{j+1} spanning $V_{j+1} \subseteq V_j$. A set of oscillatory functions Ψ_j (the wavelets) spanning W_j, the orthogonal complement of V_{j+1} into V_j, can be constructed similarly. They capture the detail lost from going from V_j to V_{j+1}, and act as high-pass filters in the sense that their expansion in terms of eigenfunctions of the Laplacian ξ_i essentially only involves eigenfunctions corresponding to eigenvalues

$$\lambda_i \in [\epsilon^{-2^j-1}, \epsilon^{-2^{j+1}-1}].$$

In particular their Sobolev norm, or smoothness, is controlled. In Figure 2 this scheme is written in pseudo-code, where the operations above are described in terms of matrices: the Gram-Schmidt procedure at each level orthogonalizes the columns of T_j into the product of an orthogonal matrix Q_j, which forms the basis of scaling functions at the next scale, and a matrix R_j, which represents T_j on Φ_j in the domain and Φ_{j+1} in the range. The assumptions of the process $\{P^t\}$ guarantee that Q_j and R_j can be constructed so that they contains only $\mathcal{O}(|R_j| \log |R_j|)$ entries above precision.

3.5 Applications

Diffusion wavelets and wavelet packets are an efficient tool for representation and approximation of functions on manifolds and graphs [4, 2], generalizing to these general spaces the wavelets that have so successfully employed for similar tasks in Euclidean spaces. They are being applied to the analysis of networks, graphs, learning tasks, document corpora, and to value function approximation in the companion paper [8].

4 Direct solution of Bellman's equation

In this section we show that the multiscale construction we discussed allows a direct solution of Bellman's equation (2). The starting point are the identities

$$V^\pi = (I - \gamma P^\pi)^{-1} R = \sum_{k \geq 0} (\gamma \Pi^{-\frac{1}{2}} T^\pi \Pi^{\frac{1}{2}})^k R = \prod_{k \geq 0} (I + \gamma^{2^k} \Pi^{-\frac{1}{2}} (T^\pi)^{2^k} \Pi^{\frac{1}{2}}) R \,,$$

`DiffusionWaveletTree` $(T_0, \Phi_0, J, \epsilon)$:

// T_0: symmetric conjugate to random walk matrix, represented on the basis Φ_0
// Φ_0 : initial basis (usually Dirac's δ-function basis), one function per column
// J : number of levels to compute
// ϵ: precision

for j **from** 0 **to** J **do**,

1. Compute sparse factorization $T_j \sim_\epsilon Q_j R_j$, with Q_j orthogonal.
2. $\Phi_{j+1} \leftarrow Q_j = H_j R_j^{-1}$ and $[T_0^{2^j}]_{\Phi_{j+1}}^{\Phi_{j+1}} \sim_{j\epsilon} T_{j+1} \leftarrow R_j R_j^*$.
3. Compute sparse factorization $I - \Phi_{j+1}\Phi_{j+1}^* = Q'_j R'_j$, with Q'_j orthogonal.
4. $\Psi_{j+1} \leftarrow Q'_j$.

end

Figure 2: Pseudo-code for construction of a Diffusion Wavelet Tree

where

$$P^\pi = \Pi^{-\frac{1}{2}} T^\pi \Pi^{\frac{1}{2}}.$$

Π is the matrix whose diagonal is the asymptotic distribution of P, and R is the reward vector. The first identity follows by the definition of T^π, the second is the usual Neumann series expansion for the inverse, and the last identity is called the Schultz formula, which is true because each term of the Neumann series appears once and only once in the product (reordering the terms of the summation is allowed because both the sum and the product are absolutely convergent). The formulas hold for $\gamma \leq 1$ and f not in the kernel of $(I - \gamma P^\pi)$. The sums and products involved are of course finite once the precision is fixed.

A key component in the construction of diffusion wavelets was the compression of the (quasi-)dyadic powers of the operator T^π. In particular $(T^\pi)^{2^j-1} f$, for any function f, is just equal to the product

$$R_j R_{j-1} \cdot \ldots \cdot R_0 f.$$

In fact R_i represents the operator $(T^\pi)^{2^i}$ on the basis Φ_i in the domain and Φ_{i+1} in the range, and hence the product above is

$$T^{1+2+2^2+\ldots+2^{j-1}} f = T^{2^j-1} f$$

represented on Φ_{j+1}, i.e. "in compressed form". The matrices $[\Phi_{j+1}]_{\Phi_j}^*$ "un-pack" this representation back onto the basis Φ_0. To obtain $T^{2^j} f$ we only need to apply T once more. In this way the computation of $T^{2^j} f$ takes only $\mathcal{O}(j|S|\log|S|)$ operations, since R_j contains about $\mathcal{O}(|S|\log|S|)$ entries. This cost should be compared to that of computing directly the matrix T^{2^j}, which is $\mathcal{O}(2^j|S|)$ since this matrix contains about $\mathcal{O}(2^j|S|)$ nonzero entries; this is also the cost of applying about 2^j times the matrix T to f.

In iterative methods such as value iteration, up to $|S|$ iterations are necessary, and the cost is thus $\mathcal{O}(|S|^2)$. Our technique has cost only $\mathcal{O}(|S|\log^2|S|)$. In some cases many less

than $|S|$ iterations are needed, especially when the problem is well-conditioned (e.g. γ far from 1), and of low precision. Even in this case our method offers several advantages, as discussed above, in terms of understanding the structure of the problem, of creating useful basis functions, and is competitive in terms of speed, as we show in the experiments.

5 Experiments

We constructed the multiscale analysis on several MDPs, on discrete and continuous spaces of different topologies. The examples and code for reproducing them will be made available on the web [1]. Here, because of space constraints, we consider only one example. It simulates a continuous two-room environment, where the two rooms have an elongated shape and are connected by a corridor. The agent has randomly explored the space, so S consists of $|S|$ randomly scattered points in the rooms. These points were generated by three Gaussian distributions, centered at the centers of the two rooms and in the corridor, with different covariances, which were then nonlinearly warped to emphasize the flexibility of the approach (see Figure 3). We point out that we could have chosen rooms of arbitrary shapes, in arbitrary dimension, as the only input to the algorithm is the set of sampled points (vertices) and the local distances between close-by points (edge weights). We construct a natural diffusion associated with the random walk in the two rooms, restricted to the states S actually explored, by letting $W(i,j) = e^{-2||x_i - x_j||^2}$. We then construct the corresponding multiscale analysis, with precision set to 10^{-15}. In Figure 3 we also represent some of the scaling functions we obtain. We then pick a random reward R on S (a vector of white Gaussian noise), and compute the corresponding value function, with $\gamma = 1$ with both the multiscale direct inverse Green function, and Matlab Conjugate Squared Method implementation.

We repeat the above for $|S| = 72, 144, 288, 432, 576, 675, 900, 1125$ and, for each S, for 20 randomly generated rewards R. We plot in Figure 5 the means and variances of the running time and the $\mathcal{L}^2$-norm of the Bellman residual ($(I - P^\pi)\tilde{V}^\pi - R$, where $\tilde{V}^\pi$ is the estimated value function), achieved by the two methods.

Figure 6 shows that the complexity of the DWT direct method grows slowly and linearly, compared to the CGS method. Of course the comparison is not completely fair, in the sense that that the larger pre-computation time for DWT was not taken into account. We ran other experiments, varying the precision and γ. When the requested precision, or the discount γ, decreases, both CGS and DWT have been running times, because of the geometric term γ^k, and the instabilities in CGS disappear For precision less than 10^{-8} the running time of CGS is a constant factor smaller than that of DWT, as predicted with asymptotic estimates for both algorithms in this regime, and given that the code for DWT is not optimized.

6 Conclusions and Future Work

We applied a novel framework based on multiscale analysis of graphs and Markov diffusion processes to designing a new fast policy evaluation algorithm. The approach constructs a

[1] at www.math.yale.edu/ mmm82/MDP.html

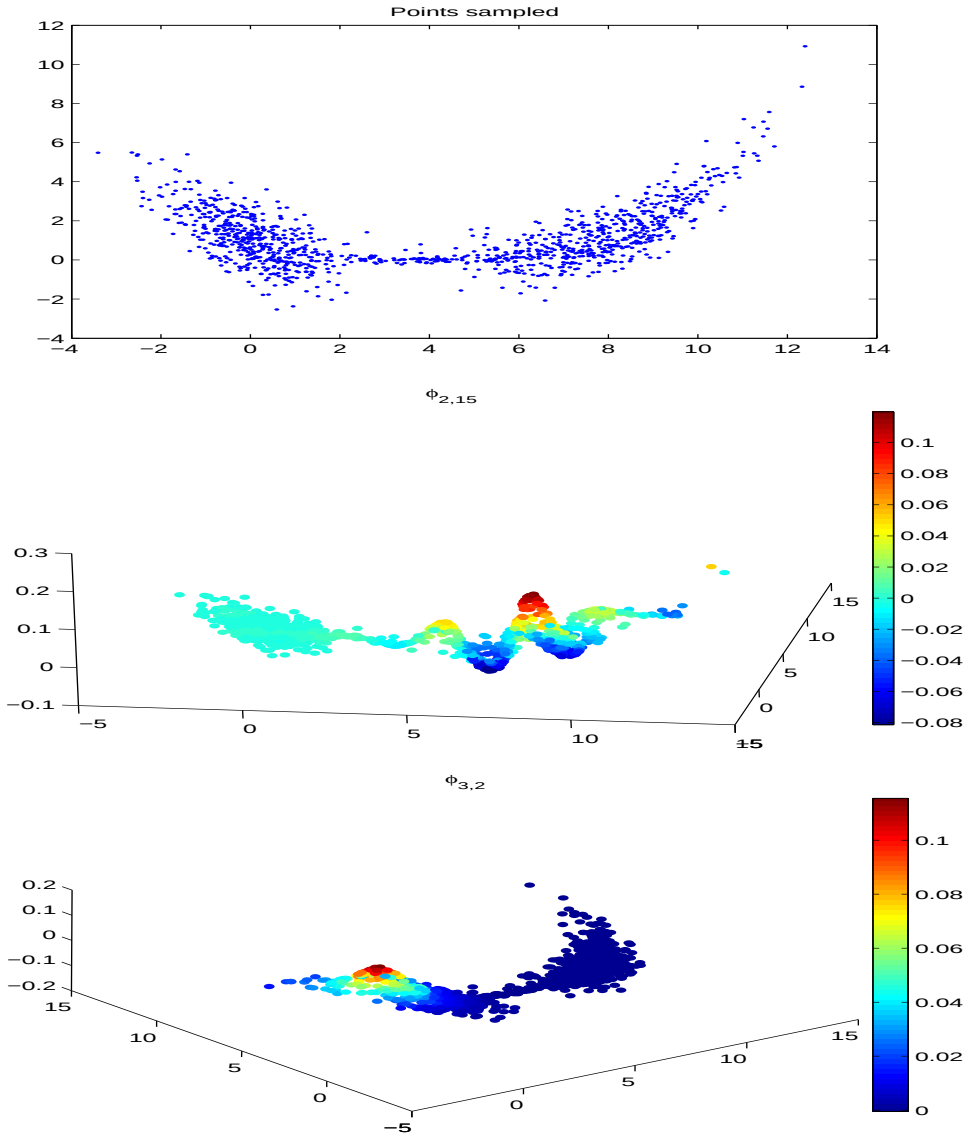

Figure 3: On top: set of samples in a two-room environment. Middle and bottom: two diffusion scaling functions built on the set, at scale 2 (center) and 3 (right). Some scaling functions are smooth "bumps", while others have oscillations, crucial for efficient approximation of smooth functions.

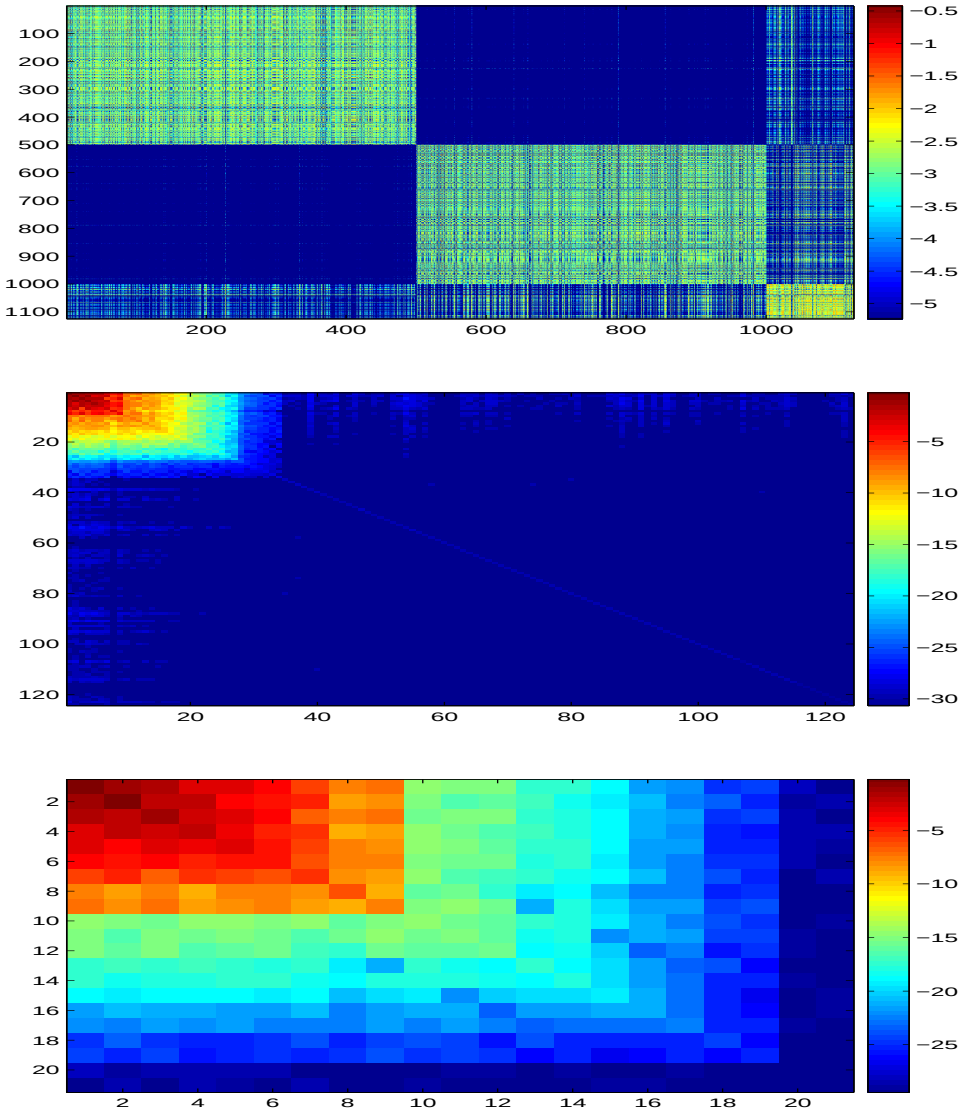

Figure 4: Compression of the powers of the symmetrized random walk T in the two-room environment: T_0 (top), T_4 (middle) and T_5. T_0 is sorted only to show the two-room and corridor structures: the two large blocks represents transitions within each room, and the bottom-right block are transitions in the corridor, with bands at the bottom and at the right indicating the transitions from the corridor to the rooms. Notice the decreasing size of the matrices: T_4 is about 10 times smaller than T_0, and T_5 about 5 times smaller than T_4. Colors represent values on a logarithmic scale.

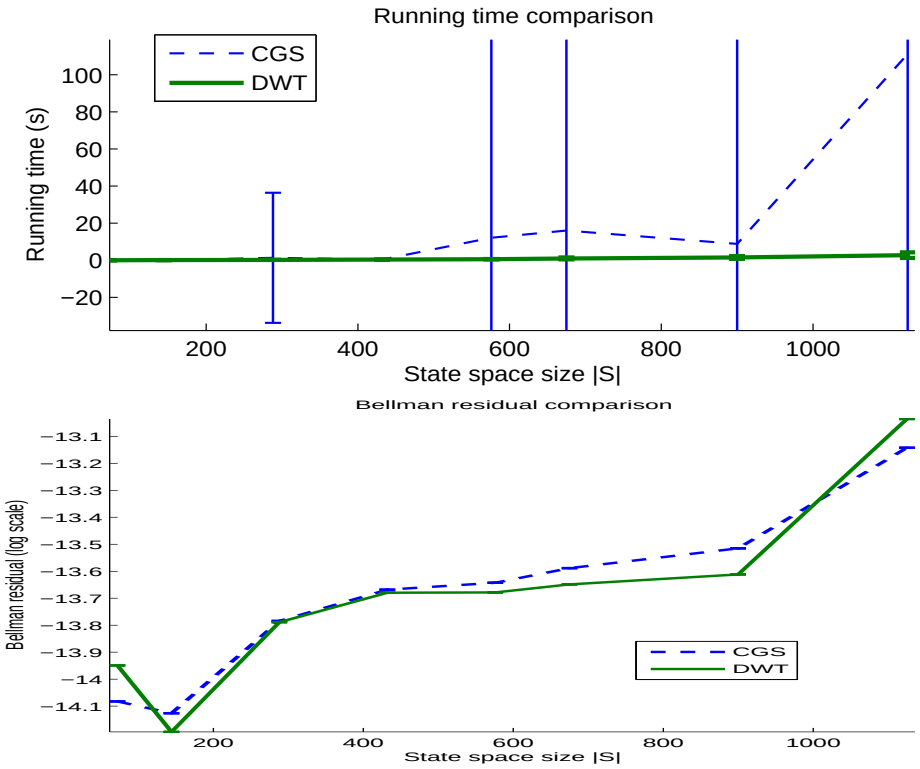

Figure 5: Top: mean and variance of running time for solving a Bellman equation on a random walk in the two-room environment, for different number of states explored (x-axis), comparing direct DWT inversion, and iterative Conjugate Gradient Squared method (Matlab implementation). Error bars for CGS are very large, showing instabilities and even lack of convergence in some runs. Bottom: The precision requested for the solution was 15 digits, we plot here is the precision (defined as log Bellman residual error) achieved.

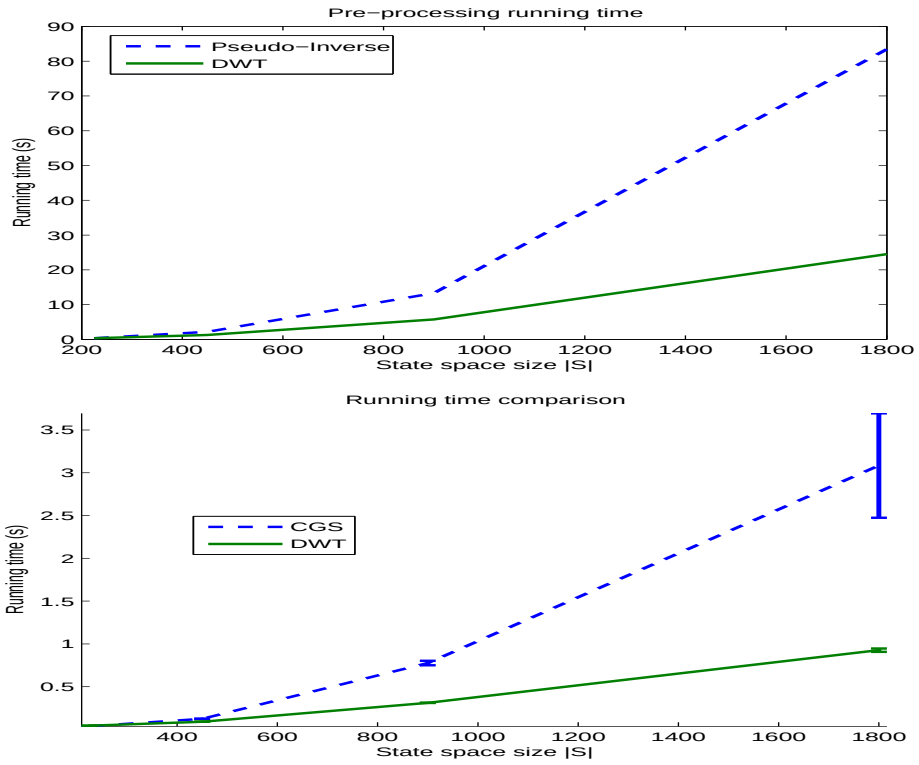

Figure 6: A run with low-precision 10^{-3}: pre-processing time for pseudo-inverse (top) and DWT (bottom) for the direct solution of Bellman's equation, using CGS and DWT.

hierarchical set of diffusion wavelet basis functions for efficiently representing powers of the transition matrix. Many directions for extending this approach are being studied, including applications to policy iteration and reinforcement learning. For large or continuous state spaces, where graphs represent a sample of the underlying state space, Nyström approximations can be exploited to interpolate basis functions to novel points. Extensions to factored state spaces are also being investigated.

References

[1] A. Barto and S. Mahadevan. Recent advances in hierarchical reinforcement learning. *Discrete Event Systems Journal*, 13:41–77, 2003.

[2] JC Bremer, RR Coifman, M Maggioni, and AD Szlam. Diffusion wavelet packets. *Tech. Rep. YALE/DCS/TR-1304, Yale Univ., to appear in Appl. Comp. Harm. Anal.*, Sep. 2004.

[3] F. Chung. *Spectral Graph Theory*. American Mathematical Society, 1997.

[4] RR Coifman and M Maggioni. Diffusion wavelets. *Tech. Rep. YALE/DCS/TR-1303, Yale Univ., to appear in Appl. Comp. Harm. Anal.*, Sep. 2004. to appear.

[5] J.G. Kemeny, J.L. Snell, and A.W. Knapp. *Denumerable Markov Chains*. Springer-Verlag, 1976.

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

[7] S. Mahadevan. Representation policy iteration. In *Proceedings of the* 21^{st} *International Conference on Uncertainty in Artificial Intelligence*, 2005.

[8] S. Mahadevan and M. Maggioni. Value function approximation with diffusion wavelets and laplacian eigenfunctions. In *NIPS, submitted*, 2005.

[9] M. L. Puterman. *Markov decision processes*. Wiley Interscience, New York, USA, 1994.