Aggregation Methods for Lineary-solvable Markov Decision Process ⋆

Mingyuan Zhong * **Emanuel Todorov** **

* *Department of Applied Mathematics, University of Washington, Seattle WA98195 USA(e-mail: zhongmy@u.washington.edu).*
** *Department of Applied Mathematics and Computer Science, University of Washington, Seattle WA98195 USA (e-mail: todorov@cs.washington.edu)*

Abstract: A general class of stochastic optimal control problems has recently been reduced to computing the principle eigenfunction of a linear operator. Here we present an approximation framework for solving such problems by using soft state aggregation over a continuous space. This approach enables us to avoid matrix factorization and take advantage of sparsity by using efficient iterative solvers. Adaptive schemes for basis placement are developed so as to provide higher resolution at the regions of state space that are visited more often. Numerical results on test problems are provided.

Keywords: Optimal control; Stochastic control; Markov desicion processs; Robotics.

1. INTRODUCTION

Nonlinear stochastic optimal control problems are fundamental in control theory, yet they remain difficult to solve globally. This motivates exploring more restricted formulations leading to more efficient algorithms. Previous work Kappen et al. [2005], Todorov [2006, 2009a] has identified a class of nonlinear stochastic optimal control problems which reduce to solving a linear problem in terms of the exponentiated optimal cost-to-go function. In infinite-horizon average-cost settings (which are the focus of the present paper) the problem comes down to computing the principal eigenfunction of a linear operator. Despite this linearity, when dealing with physical systems such as vehicles or robots, the state space become continuous and the curse of dimensionality emerges. Carefully designed approximation schemes are needed for such problems.

Some numerical methods applicable to this problem class have previously been developed, in particular direct MDP discretization in Todorov [2009a] and function approximation using Gaussian bases in Todorov [2009b]. Discretization is useful in terms of obtaining "ground-truth" solutions in low-dimensional problems and comparing to the results of more advanced algorithms that need fine-tuning, but is not applicable to higher-dimensional problems. Gaussian bases are promising, however they have some disadvantages. First the resulting eigen-problem is weighted (it is in the form $\lambda F\mathbf{w} = G\mathbf{w}$ instead of $\lambda\mathbf{w} = G\mathbf{w}$) which slows down the solver. Second, positivity of the solution (which is needed since we are solving for the exponent of a function) is hard to enforce without introducing inequality constraints. Third, when λ is also unknown, the methods might converge to the wrong eigenvector.

The new method proposed here is different from the above methods in concept. Previous methods are function-approximation schemes. They approximate the desirability function $z(x) = \exp(-v(x))$ defined on the continuous state space and solve the corresponding linear equation; v is the optimal cost-to-go function. Here, we are proposing a Problem-Approximation Scheme. Using aggregation methods, see Bertsekas [2010], Singh et al. [1995], we approximate the optimal control problem as one defined on discrete states, which we call clusters, and then solve the resulting linearly-solvable MDP (or LMDP). The new approximation scheme leads to simple eigen-problems in the form $\lambda\mathbf{w} = QG\mathbf{w}$, so it always converges to the principal eigenvalue and guarantees positivity of the solution. This scheme also provides extra flexibility in terms of constructing the approximation in first-exit problems (see below). On the other hand, it requires more dynamics evaluations compared to function approximation with Gaussian bases – by a factor of $O(m)$ where m is the state dimensionality – because certain integrals cannot be computed analytically and instead require numerical approximations via Gaussian cubature.

⋆ This work was supported by the US National Science Foundation.

2. LINEARLY-SOLVABLE OPTIMAL CONTROL PROBLEMS

In this section, based on Todorov [2009a]Todorov [2009b], we summarize the Linear Markov Decision Process (LMDP) problem class in continuous space and time. Our work mostly focuses on solving such continuous problems with the average-cost setting, but we will also address first-exit settings in later parts.

2.1 Linearly-solvable MDPs

Consider an MDP with state $\mathbf{x} \in X \subseteq \mathbf{R}^n$. Let $p(\mathbf{x}'|\mathbf{x}, u) = u(\mathbf{x}'|\mathbf{x})$ denote the transition probability given a certain control signal u, and $p(\mathbf{x}'|\mathbf{x})$ denote the transition probability without any control, also known as

passive dynamics. The optimal cost-to-go function is given by the Bellman equation

$$v(\mathbf{x}) = \min_u \{l(\mathbf{x}, u) + E_{\mathbf{x}' \sim p(\cdot|\mathbf{x},u)}[v(\mathbf{x}')]\}. \quad (1)$$

For this problem class the immediate cost function is defined as

$$\tilde{l}(\mathbf{x}, u) = \tilde{q}(x) + KL(u(\cdot|\mathbf{x})||p(\cdot|\mathbf{x})), \quad (2)$$

where KL denotes the Kullback-Leibler divergence between two probability distributions. Define the *desirability* function $z(\mathbf{x}) = \exp(-v(\mathbf{x}))$, where $v(\mathbf{x})$ is the optimal cost-to-go function. Then the optimal control law is

$$u^*(\mathbf{x}'|\mathbf{x}) = \frac{p(\mathbf{x}'|\mathbf{x})z(\mathbf{x}')}{\mathcal{G}[z](\mathbf{x})}, \quad (3)$$

where the operator $\mathcal{G}$ is defined as

$$\mathcal{G}[z](\mathbf{x}) = \int_{\mathbf{x}' \in X} p(\mathbf{x}'|\mathbf{x})z(\mathbf{x}')d\mathbf{x}'. \quad (4)$$

The Bellman equation for infinite-horizon average-cost problems can be written as

$$\exp\left(\tilde{q}(\mathbf{x}) - \tilde{c}\right) z(\mathbf{x}) = \mathcal{G}[z](\mathbf{x}). \quad (5)$$

The desirability function is the principal eigenfunction and it is guaranteed to be positive. The corresponding eigenvalue is $\lambda = \exp(-\tilde{c})$ where $\tilde{c}$ is the average cost per step. For the first-exit formulation we have $\tilde{c} = 0$ and $z(\mathbf{x}) = \exp(-q_T(\mathbf{x}))$ at terminal states. This makes the problem a linear equation rather than an eigenfunction problem (see Todorov [2009a]).

2.2 Linearly-solvable controlled diffusions

Here we consider a class of continuous-time optimal control problems with the following stochastic dynamics:

$$d\mathbf{x} = \mathbf{a}(\mathbf{x})dt + B(\mathbf{x})(\mathbf{u}dt + \sigma d\omega), \quad (6)$$

where $\omega(t)$ represents Brownian motion, and σ is the noise magnitude. The cost function is in the form

$$l(\mathbf{x}, \mathbf{u}) = q(\mathbf{x}) + \frac{1}{2\sigma^2}\|\mathbf{u}\|^2. \quad (7)$$

Note that the noise is assumed to lie in the same subspace as the control. The fact that the noise amplitude also appears in the cost function is unusual; however, $l(\mathbf{x}, \mathbf{u})$ can be scaled by σ^2 without changing the optimal control law, and this scaling factor can be absorbed in the state cost $q(\mathbf{x})$, so this is not a restriction. Now we can discretize this dynamical system. The one-step transition probability under the passive dynamics is approximated as a Gaussian distribution

$$p(\mathbf{x}'|\mathbf{x}) = \mathcal{N}(\mathbf{x} + h\mathbf{a}(\mathbf{x}), h\Sigma(\mathbf{x})), \quad (8)$$

where $h\Sigma(\mathbf{x}) = h\sigma^2 B(\mathbf{x})B(\mathbf{x})^T$ is the covariance of noise. The transition probability with control $\mathbf{u}$ is

$$p(\mathbf{x}'|\mathbf{x}, \mathbf{u}) = \mathcal{N}(\mathbf{x} + h\mathbf{a}(\mathbf{x}) + hB(\mathbf{x})\mathbf{u}, h\Sigma(\mathbf{x})). \quad (9)$$

The formula for KL divergence between Gaussians gives

$$KL(p(\mathbf{x}'|\mathbf{x}, \mathbf{u})||p(\mathbf{x}'|\mathbf{x})) = \frac{h}{2\sigma^2}\|\mathbf{u}\|^2. \quad (10)$$

Thus, the familiar quadratic energy cost is a special case of the KL divergence cost defined earlier. It can be shown that in the limit $h \to 0$, the solution to the above discrete-time problem converges to the solution of the underlying continuous-time problem. Thus, if we define

$$\tilde{q}(\mathbf{x}) = hq(\mathbf{x}), \quad (11)$$

the continuous problem is approximated as a continuous-space discrete-time LMDP. For example, for the infinite horizon average-cost setting, (5) becomes

$$\exp\left(hq(\mathbf{x}) - hc\right) z(\mathbf{x}) = \mathcal{G}[z](\mathbf{x}). \quad (12)$$

3. SOFT STATE AGGREGATION SCHEME

In this section we first summarize the idea of soft state aggregation Bertsekas [2010], Singh et al. [1995]. Then we introduce our choice of aggregation and deaggregation probabilities. After that, we show how to recover the solution of the original problem using clusters obtained from aggregation. Issues regarding setting parameters of those clusters are discussed in the end.

3.1 Soft state aggregation for LMDPs

In order to aggregate the entire space into a finite number of clusters, we need to construct the clusters $\mathcal{S}$ and define two choices of probabilities relating the clusters to the original system states.

Definition 1. (Aggregation and deaggregation probabilities)

(1) *Aggregation probability* $p(i|x) = \phi_i(x)$, $x \in \mathcal{X}, i \in \mathcal{S}$ is a probability-like quantity that is interpreted as the "degree of membership of x in the aggregate state i".
(2) *Deaggregation probability*, $p(x|i) = d_i(x)$, $x \in \mathcal{X}, i \in \mathcal{S}$ is a probability-like quantity that is interpreted as the "degree to which i is represented by x".

Aggregation and deaggregation probabilities naturally satisfy the following conditions

Fact 2. (Aggregation and deaggregation probabilities)

(1) nonnegativity
$$\phi_j(x) \geq 0, d_i(x) \geq 0. \quad (13)$$
(2) normalization conditions
$$\sum_{j \in \mathcal{S}} \phi_j(x) = 1, \int_{x \in \mathcal{X}} d_i(x)dx = 1. \quad (14)$$
(3) Bayes rule
$$\phi_i(x) = p(i|x) = \frac{p(x|i)p(i)}{p(x)} = \frac{d_i(x)p(i)}{p(x)}, i \in \mathcal{S}, x \in \mathcal{X}, \quad (15)$$
where $p(i)$ and $p(x)$ are some probability densities over $\mathcal{S}$ and $\mathcal{X}$.

When the aggregation probabilities $\phi_i(x)$ are not all 0 or 1, this method is called *Soft Aggregation*, which implies that each state of the original system may belong to multiple clusters.

Suppose we have a continuous-state LMDP with passive dynamics $p(x'|x)$, $x, x' \in \mathcal{X}$. We would like to approximate the desirability function $z(x), x \in \mathcal{X}$. Based on the aggregation framework, the transition probabilities among the clusters can be expressed as

$$\begin{aligned} \hat{p}(j|i) &= \iint_{x,x' \in \mathcal{X}} p(j|x')p(x'|x)p(x|i)dxdx' \\ &= \iint_{x',x \in \mathcal{X}} \phi_j(x')d_i(x)p(x'|x)dxdx'. \end{aligned} \quad (16)$$

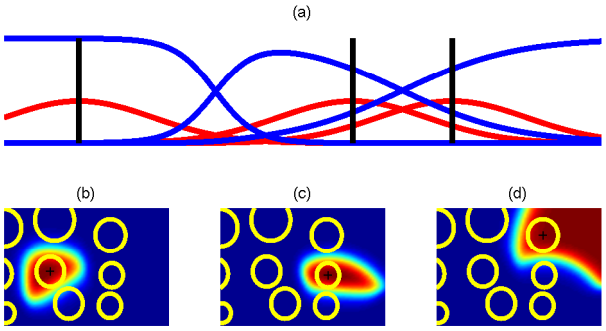

Fig. 1. (a) Aggregation and deaggregation functions in 1D. Blue lines show aggregation, while red lines show deaggregation functions. Black lines show the means of the clusters. (b-d) shows several aggregation functions in 2D. The ellipses correspond to the contours of the Gaussian deaggregation functions.

The state cost function over clusters is

$$\exp(-\tilde{q}_i) = \int_{x\in\mathcal{X}} \exp(-hq(x))d_i(x)dx. \qquad (17)$$

This completes the construction of the approximating LMDP on clusters, which is defined on a finite and discrete state space.

3.2 Choice of aggregation and deaggregation probabilities

In this paper we choose the deaggregation probabilities as multi-dimensional normal distributions:

$$d_i(x) = c_i^{-1}\exp(-(x-m_i)^T S_i(x-m_i)/2), \qquad (18)$$

where c_i^{-1} denotes the normalization factor. Then $p(x)$ and $p(i)$ in (15) are chosen as the following mixtures(19):

$$p(i) = c_i, p(x) = \Sigma_{i\in\mathcal{S}} p(i)p(x|i), \qquad (19)$$

then based on (15), the aggregation functions are

$$\phi_i(x) = \frac{\exp(-(x-m_i)^T S_i(x-m_i)/2)}{\sum_{i\in\mathcal{S}}\exp(-(x-m_i)^T S_i(x-m_i)/2)}. \qquad (20)$$

The shape of the aggregation and deaggregation functions is illustrated in Fig 1. In order to improve computational efficiency and generate a sparse transition probability matrix, the summation in (20) will be limited to nearby clusters, and $\phi_i(x)$ will be set to 0 if the m_i is not among the k nearest neighbors to x. Note that we only choose the shape of the deaggregation functions (Gaussian). The aggregation functions are then constructed automatically via normalization. The result of the normalization is to reshape the Gaussians in a way that fills the gaps and covers the space more uniformly.

3.3 Finding the desirability function $z(\mathbf{x})$ and optimal control law $\mathbf{u}^*(\mathbf{x})$

The double integral (16) is not trivial to calculate numerically (see Supplement). Currently we are employing a cubature formula Lu et al. [2005], Stroud [1971] which requires evaluating the aggregation function and the $a(x)$ function $O(m)$ times, where m is the state space dimensionality. Once the discrete-space LMDP is constructed, we can solve it by finding the principal eigen-pair for a problem in the form

$$\lambda\mathbf{z} = QP\mathbf{z}. \qquad (21)$$

Here, $P_{ij} = \hat{p}(j|i)$, $\lambda = \exp(-hc)$, Q is a diagonal matrix with $Q_{ii} = \exp(-\tilde{q}_i)$ in (17). One way to solve this problem is the power iteration method(in Trefethen et al. [1997]): $\lambda\mathbf{z}^{k+1} = QP\mathbf{z}^k$ with $\|\mathbf{z}\|$ fixed. The matrix QP is guaranteed to have a principal eigenvalue less than 1, and the corresponding eigenvector is guaranteed to have positive elements– because all elements of the matrix are non-negative.

For the first-exit formulation Todorov [2006], the corresponding linear equation is

$$\mathrm{diag}(\exp(\mathbf{q}_\mathcal{N})) - P_{\mathcal{N}\mathcal{N}}\mathbf{z}_\mathcal{N} = P_{\mathcal{N}\mathcal{T}}\exp(-\mathbf{q}_\mathcal{T}). \qquad (22)$$

Here $\mathcal{N}$ stands for non-terminal states and $\mathcal{T}$ stands for terminal states, and $P_{\mathcal{N}\mathcal{N}}, P_{\mathcal{N}\mathcal{T}}$ are the corresponding blocks of P in (21). We can approximate the continuous-state problem if we allocate several clusters as terminal states. Note that this is not equivalent to the continuous function approximation approach which requires a boundary condition on $z(x)$. Let $T \subset \mathcal{R}^m$ denote the set of terminal states in the continuous state space. The continuous function-approximation scheme would involve integrals like $\int_{x'\in\mathcal{R}^m\setminus T} p(x'|x)f(x')dx'$ which would be difficult to evaluate in high-dimensional problems.

Once the discrete-space LMDP is solved, the continuous-space desirability function is

$$z(x) = \sum_{j\in\mathcal{S}} \phi_j(x)z_j, \qquad (23)$$

and then the optimal control law $u^*(x)$ can be computed from (3).

With regard to computational complexity, estimating (16) tends to be the bottleneck. Finding the K nearest neighbors can be done efficiently using tree decompositions, while evaluating (16) requires $O(NKm^3)$ computations where N is the total number of clusters (see Supplement).

3.4 Determining the position and covariances of clusters

The clusters (defined by the Gaussian deaggregation functions) are characterized by their means and covariances. Obviously the choice of means and covariances will have a large effect on the quality of the approximation. Here we present a method for choosing these parameters in a suitable way. First we address the choice of means, and then the choice of covariances.

The approximation is more accurate near the clusters. One approach is to attempt to cover the entire state space with a grid of Gaussians, but this will not scale to high dimensional systems. Instead we use the following method. We only add clusters (nodes), restricted to be sufficiently far away from the already included clusters. If the clusters are further restricted to lie within a finite region of state space, the method is guaranteed to terminate. After solving for the optimal control law for current selection of nodes, we generate prospective nodes based on a stochastic dynamical simulation starting from the current nodes (or given initial states). Then we introduce random perturbations. A prospective node is added if it is sufficiently far from all current nodes. Nodes with lower $z(x)$, or equivalently higher cost-to-go, usually lie far away from target/limiting trajectory. They effectively surround

the region of good states and thus prevent the number of nodes from expanding indefinitely.

We also implement a mechanism to avoid generating prospective nodes at low $z(x)$ regions or regions that have already been filled with nodes. This effectively restricts sampling to low $z(x)$ regions, which is where the controlled system spends most of its time. The quality of the approximation is of course better if the initial sampling covers the region where good states are found. The distances between nodes should approximately match the characteristic distance of the system determined by $ha(x)$ and $\sqrt{h}\sigma B(x)$, which are the magnitudes of the passive dynamics and transition probabilities. This means the time step should be big enough if one wants to solve the problem with fewer clusters.

Once the means of the clusters are chosen, we need to choose the covariances. One possibility is to use gradient descent, however this is likely to be slow because the covariances have a large number of parameters. Instead we use a heuristic method to ensure that the clusters have some overlap. Since the solution $z(x)$ tends to be more spread out in the directions where the noise acts, the covariances need to be more elongated in these dimensions.

Our current results involve setting covariances in the following ways. (1) Manually set them, but this involves some insight into the specific system. (2) Assume that they have the same shape, i.e. the inverse covariance matrix S_i for all clusters is the same up to multiplication by a scalar; this scalar is determined by the distance to other clusters and how much overlap between clusters we need. Keeping S_i diagonal tends to give good enough results in most of our test problems.

4. NUMERICAL RESULTS

In this section we present numerical results. First we will illustrate that the results given by the aggregation framework are meaningful. Second we will illustrate that the method scales to high-dimensional system.

4.1 Test problems, solution and dynamical simulation

Here we will focus on the desirability function $z(\mathbf{x})$, cost-to-go function $v(\mathbf{x})$, optimal control law $u^*(\mathbf{x})$, and dynamical simulations based on $u^*(\mathbf{x})$. When possible, we will compare these results to the "ground truth" obtained from an MDP discretization Todorov [2009b] on a dense grid.

Example 1: Car-on-the-hill This test problem is adapted from Todorov [2009a]. It has 2D state space (position and horizontal velocity) and 1D control space. This dynamical system simulates a point mass (the "car") moving along a curved road (an inverted Gaussian). The control signal is a force, which can only affect the tangential velocity directly. One interesting property of this model is that the dynamics are augmented with the following rule. When the car hits the "walls" at x_{min} or x_{max} , its speed drops to 0, at which point we have $B = [0; 0]$ (i.e. the apparent inertia goes to infinity at contact). This discontinuity cannot be captured by the continuous-time diffusion model (6), yet it is easily captured by our discrete-time LMDP.

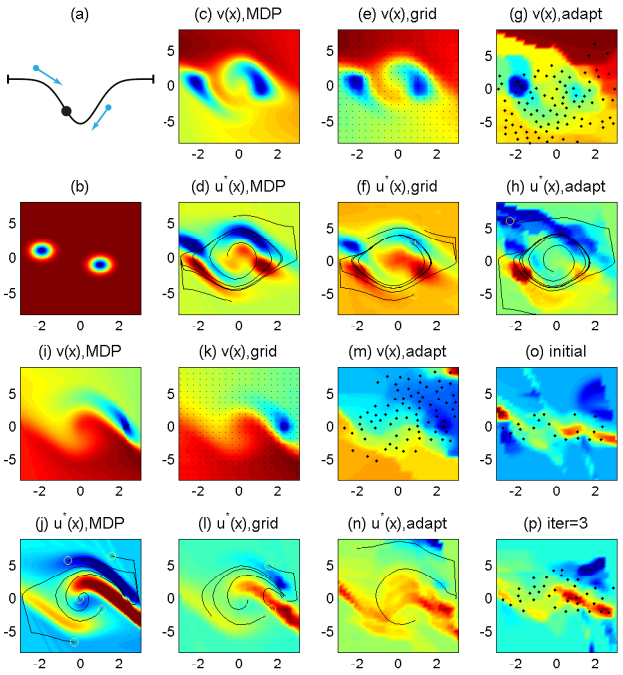

Fig. 2. Results for car-on-a-hill. (a)demonstrates the problem. Curve represents the "hill" and arrows represents via points in average-cost formulation. (b)demonstrates the state cost function $q(\mathbf{x})$ of the average cost formulation. (c-h) show results given by average-cost formulation. (i-p) show results given by the first-exit problem. Except for (a,b,o,p), the top and third rows show the cost-to-go function$(x) = -\log(z(x))$, with blue represents low cost-to-go or high desirability, dots represents where clusters are. The second and fourth rows show the corresponding optimal control law. Solid lines show the trajectories given by a simulation with several random initial states (indicated by yellow circles). (c-d,i-j) show the results given by MDP with 151 x 151 states sampled on bigger region than shown. (e-f,k-l) show the results given by putting clusters on a 21x21 grid. (g-h,m-n) shows the results given by putting clusters with adaptive scheme. The average-cost one (g-f) uses 94 clusters, (m-n) the first-exit one uses 78 clusters. For those results by adaptive clusters, their covariances are adaptive to match their distance with the nearest neighbor. (o) shows initial clusters and (p) shows clusters after 3 iterations. The result given by (m-n) is obtained after the 14th iteration. Except (a), horizontal axis represents horizontal position and vertial axis represents velocity.

The reason for constructing a model with collisions is that we hope that our methods will work for locomotion and hand manipulation where contact phenomena and discontinuities are essential.

For the average-cost setting, we design a cost-function $q(\mathbf{x})$ (Fig.2(b)) which encourates the car to pass through two targets that have non-zero desired velocities; thus the optimal behavior involves a limit cycle. As shown in Fig.2(c-h), the cost-go function and control law given by our approximations are consistent with the dense MDP solution, even though here we use a small number of bases.

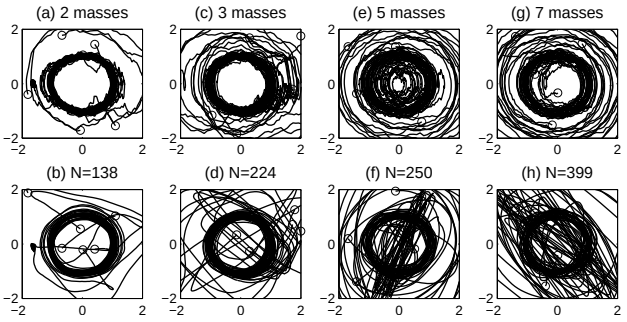

Fig. 3. Results for N masses model. The first row represents simulated simulated trajectories projected to x_1 vs v_1 plane, while the second row represents those projected to x_1 vs x_2 plane. (a-b) for 2 masses, (c-d) for 3 masses, (e-f) for 5 masses, (g-h) for 7 masses. N stands for number of clusters. Circles represents initial states. All of them are calculated with hundreds of clusters. We can see that most trajectories converge to the unit circle on planes, which are projections of the theoritial attracting trajectories.

We can also use this dynamical model to create a first-exit problem. The goal of this model is to park the car (i.e. achieve zero velocity) at horizontal position 2.35. The running state cost is constant; thus it effectively acts as a cost for time. Again, in Fig.2(i-p), the results are consistent with the MDP method. Subplots (o-p) give the initial clusters and the intermediate clusters during iterations. We can see that there are fewer clusters at the right-bottom corner, where the car visits less frequently.

For the car-on-a-hill model, one interesting result is that our framework could capture collisions properly, by obtaining control laws take advantage of collissions. This behavior is difficult to capture using trajectory-based methods.

Example 2: Coupling a series of masses on ideal springs This model simulates several identical 1D masses attached on ideal frictionless springs, which are dynamically independent (the only coupling is due to the cost function). Each mass can oscillate with any amplitude. The cost function encodes two concurrent objectives: (1) fix the energy of each mass-spring to a constant; (2) make mass number i oscillate with phase $\pi/2$ ahead of mass number $i+1$, for all i. Here we use an average-cost setting.

This model is rather simple, but it has advantages when tuning the algorithm. First, it has similar behavior when scaling to higher dimensionality (simply changing the number of masses). Second, the limiting behavior of this system can be easily computed theoretically: it is in the form of cosine functions $\cos(Ct+\phi)$ with appropriate phase changes. Note that the solver does not "know" about this analytical form and instead is applying a generic numerical method.

Results in Fig 3 show that we can control systems with 2,3,5,7 masses using only hundreds of clusters. Since this model is better used for fine-tuning this method, we will discuss more details of this model in the following section about scalability.

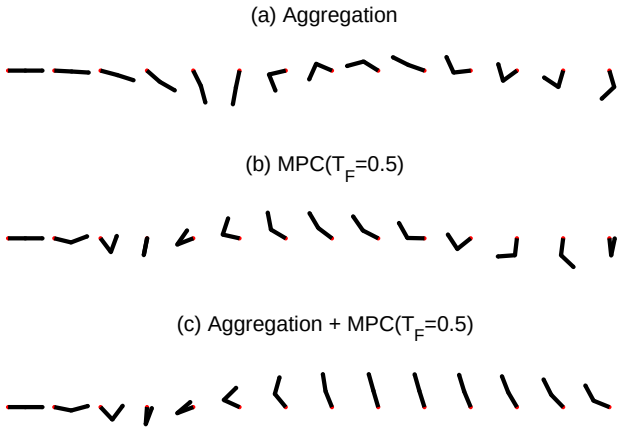

Fig. 4. Results for acrobot. All figures shows acrobot starting from the same position to $t = 2$.

Example 3: Acrobot The acrobot is a 2-link arm with control torque only applied to the joint between the two parts. The details of this model is well-explained in Coulom [2002], and we used exactly the same model and parameters. Our test focus on swing up the acrobot to the upright position and balancing it, thus the state cost function is set to be $q(\mathbf{x}) = -CH(\mathbf{x}), C > 0$, where C is a constant and $H(\mathbf{x})$ is the height of the acrobot's tip.

We explored two approaches to solving this problem: the aggregation method developed in this paper, and model-predictive control (MPC) (Tassa [2011]). We found that the two approaches have complementary stengths, which can be combined to yield improved performance. MPC is difficult here because the system is underactuated and the optimal trajectory often has long duration and compicated shape – thus trajectory optimization without aid from a global function approximator can easily fail. The aggregation method provides a more global approximation, however it tends to smooth out the value function, often resulting is control that are not sufficiently large. This can be overcome by adding more bases, at the expense of computational complexity.

Now we show how to combine them. At time i, the MPC minimize the cost on trajectory, i.e., solving $\min_{\mathbf{u}_{i,\dots,i+N}} \left[\Sigma_{k=i}^{i+N-1} l(\mathbf{x}_i, \mathbf{u}_i) + v_F(\mathbf{x}_N)\right]$, where the states $\mathbf{x}_k$ and controls $\mathbf{x}_k$ are constrained by the dynamics . MPC apply only $\mathbf{u}_i$ to the system and then resolve the problem at next time step. To combine MPC and aggregation methods, we set $v_F(\mathbf{x})$ as the cost-to-go function solved from aggregation method under average-cost setting.

Numerical results confirm above arguments as in Fig. 4. With higher receding horizon ($t_F = 1$), the combined method would swing up the acrobot and also balance it. Imposing noise on simulation would give similar results. More work is needed to be done on the MPC-aggregation combination.

4.2 Scalability to High dimensional system

Scalability to high dimensional systems is the primary challenge of solving Bellman equations numerically. A successful method needs to satisfy several conditions. First, the method should give a controller not far away from

the true solution. Second, the computational complexity of the method should scale well with the number of samples used. Third, the number of samples needed to achieve good accuracy should scale well with dimensionality. The first and second conditions normally hold(e.g. MDP), and the real challenge is the third one.

At least in the example 2 shown, our framework can control a high-dimensional system semi-globally with a surprisingly small number of clusters given the dimensionality. In general, the number of samples needed is likely to be problem specific, and might blow up due to the complexity of a system. Nevertheless our numerical results up to date are encouraging.

We also note that the CPU time scales well with dimensionality. Our code is written in C and is multithreaded. It runs on a quad-core 3GHz Intel i7 processor. Going from 2 masses to 7 masses (4 to 14 state dimensions), the total CPU time increases from 0.5 seconds to 4 seconds. The CPU time for estimating the probability transition matrix increases from around 15 milliseconds to 80 milliseconds. When using 7 masses and 20000 clusters, computing the transition matrix takes 56.5 seconds with 6 threads, as opposed to 243.7 seconds with 1 thread. Thus our algorithm successfully takes advantage of multithreading.

5. SUMMARY

Here we presented a new problem-approximation scheme, which transforms a linearly solvable optimal control problem with continuous state space into a problem with discrete state space. This provides extra flexibility in the problem formulation and also avoids numerical issues with function approximation methods. Preliminary results show that it can generate acceptable control laws and is scalable to higher dimensional systems. We also developed an adaptive scheme for aggregation/deaggregation functions that increases the approximation accuracy in the regions of state space which are visited most often.

In the future, we hope to broaden our selection of aggregation and deaggregation functions, explore different ways to adapt them, explore combining MPC with aggregation, and apply the method to more complex problems. Systematic comparison with other existing methods is also needed.

ACKNOWLEDGEMENTS

This work was supported by the US National Science Foundation.

REFERENCES

D. Bertsekas. *Dynamic programming and optimal control* , 3rd Edition. . Volume II, Chap 6, revision at May 4, 2010

R. Coulom. *Reinforcement learning using neural networks, with applications to motor control*. Ph.D. dissertation, Institut National Polytechnique de Grenoble, 2002.

H. Kappen. *Linear theory for control of nonlinear stochastic systems*. Phys Rev :ett 95: 200201 ,2005.

J Lu, DL Darmofal. *Higher-dimensional integration with Gaussian weight for applications in probabilistic design* .SIAM J. SCI. COMPUT, Vol. 26, No. 2, pp. 613, 2005

S. Singh, T. Jaakkola and M.Jordan. *Reinforcement learning with soft state aggregation*. Advances in Neural Information Processing Systems, 1995.

A. H. Stroud. *Approximate Calculation of Multiple Integrals, Prentice-Hall*. Englewood Cliffs,NJ, 1971.

Y. Tassa, T. Erez, and E. Todorov. *Fast model predictive control for complex robotic behavior*. Manuscript under review, 2011.

E. Todorov. *Linearly-solvable Markov decision problems*. Advances in Neural Information Processing Systems, 2006.

E. Todorov. *Efficient computation of optimal actions*. PNAS, July 14, 2009 vol. 106 no. 28 11478-11483 .

E. Todorov. *Eigen-function approximation methods for linearly-solvable optimal control problems*. IEEE International Symposium on Adaptive Dynamic Programming and Reinforcemenet Learning, 2009.

LN Trefethen, D Bau . *Numerical Linear Algebra*. SIAM1997

Appendix A. NUMERICAL ISSUES FOR ESTIMATING PROBABILITY TRANSITION MATRIX OF CLUSTERS

$$\hat{p}(j|i) = \iint_{x',x\in\mathcal{X}} \phi_j(x') d_i(x) p(x'|x) dx dx' \tag{A.1}$$

is not trivial to be calculated numerically. With $x' = \mathbf{x} + ha(\mathbf{x}) + \sqrt{h}\sigma B(\mathbf{x})\xi$, the integral becomes

$$\hat{p}(j|i) = \iint_{x\in\mathcal{X},\xi\in\mathcal{R}^{m_c}} \phi_j(\mathbf{x}+ha(\mathbf{x})+\sqrt{h}\sigma B\xi) d_i(x)\mathcal{N}(\mathbf{0},I) d\xi. \tag{A.2}$$

Here $\mathcal{N}(\mathbf{0},I)$ represent the probability density function of a unit Gaussian distribution. $d_i(x)\mathcal{N}(\mathbf{0},I)$ is a Gaussian function. If problem defined in entire space or clusters are small, the above integral can be approximately calculated by the cubature formula. Those formulas approximate the integral with a weighted summation of the function evaluated at carefully selected sampling points.

$$I[\Phi] = \int \Phi(\mathbf{y}) \exp(-\mathbf{y}^T\mathbf{y}) d\mathbf{y} \approx Q[\Phi] = \Sigma_{j=1}^{n_{sample}} w_j \Phi(\mathbf{y}^j) \tag{A.3}$$

We currently are using one with $n_{sample} = 2(m+m_c)+1$ points, a variation from Stroud [1971], where m is the dimensionality of the state space and m_c is that of the control space.

$$\begin{aligned} Q[\Phi] &= w_1\pi^{n/2}\Phi(\mathbf{0}) + w_2\Sigma_{full\ sym.}\Phi(\sqrt{\lambda},0,..,0), \\ w_1 &= \frac{2\lambda-1}{2\lambda}, w_2 = \frac{4}{n}\lambda, \end{aligned} \tag{A.4}$$

λ is an arbitrary parameter. Here, "full sym." means all possible indexes, permutations and reflections. Therefore, for a m dimensional system with N clusters, the time-complexity for computing the entire matrix is $O(NKm^3)$, where K corresponds to how many neighboring points you have chosen to evaluate. That is due to :(1) calculating a Gaussian needs $O(m^2)$ time, (2) calculating a row of the probability transition matrix needs evaluating Gaussians $O(Km)$ times. In our trials, computing the transition matrix seems to be a bottleneck.