Efficient computation of optimal actions

Emanuel Todorov[1]

Departments of Applied Mathematics and Computer Science & Engineering, University of Washington, Box 352420, Seattle, WA 98195

Edited by James L. McClelland, Stanford University, Stanford, CA, and approved April 28, 2009 (received for review November 16, 2007)

Optimal choice of actions is a fundamental problem relevant to fields as diverse as neuroscience, psychology, economics, computer science, and control engineering. Despite this broad relevance the abstract setting is similar: we have an agent choosing actions over time, an uncertain dynamical system whose state is affected by those actions, and a performance criterion that the agent seeks to optimize. Solving problems of this kind remains hard, in part, because of overly generic formulations. Here, we propose a more structured formulation that greatly simplifies the construction of optimal control laws in both discrete and continuous domains. An exhaustive search over actions is avoided and the problem becomes linear. This yields algorithms that outperform Dynamic Programming and Reinforcement Learning, and thereby solve traditional problems more efficiently. Our framework also enables computations that were not possible before: composing optimal control laws by mixing primitives, applying deterministic methods to stochastic systems, quantifying the benefits of error tolerance, and inferring goals from behavioral data via convex optimization. Development of a general class of easily solvable problems tends to accelerate progress—as linear systems theory has done, for example. Our framework may have similar impact in fields where optimal choice of actions is relevant.

action selection | cost function | linear Bellman equation | stochastic optimal control

If you are going to act, you might as well act in the best way possible. But which way is best? This is the general problem we consider here. Examples include a nervous system generating muscle activations to maximize movement performance (1), a foraging animal deciding which way to turn to maximize food (2), an internet router directing packets to minimize delays (3), an onboard computer controlling a jet engine to minimize fuel consumption (4), and an investor choosing transactions to maximize wealth (5). Such problems are often formalized as Markov decision processes (MDPs), with stochastic dynamics $p(x'|x,u)$ specifying the transition probability from state x to state x' under action u, and immediate cost $\ell(x,u)$ for being in state x and choosing action u. The performance criterion that the agent seeks to optimize is some cumulative cost that can be formulated in multiple ways. Throughout the article we focus on one formulation (total cost with terminal/goal states) and summarize results for other formulations.

Optimal actions cannot be found by greedy optimization of the immediate cost, but instead must take into account all future costs. This is a daunting task because the number of possible futures grows exponentially with time. What makes the task doable is the optimal cost-to-go function $v(x)$ defined as the expected cumulative cost for starting at state x and acting optimally thereafter. It compresses all relevant information about the future and thus enables greedy computation of optimal actions. $v(x)$ equals the minimum (over actions u) of the immediate cost $\ell(x,u)$ plus the expected cost-to-go $\mathrm{E}[v(x')]$ at the next state x':

$$v(x) = \min_u \{\ell(x,u) + \mathrm{E}_{x'\sim p(\cdot|x,u)}[v(x')]\}. \qquad [1]$$

The subscript indicates that the expectation is taken with respect to the transition probability distribution $p(\cdot|x,u)$ induced by action u. Eq. **1** is fundamental to optimal control theory and is called the Bellman equation. It gives rise to Dynamic Programming (3) and Reinforcement Learning (2) methods that are very general but can be inefficient. Indeed, Eq. **1** characterizes $v(x)$ only implicitly, as the solution to an unsolved optimization problem, impeding both analytical and numerical approaches.

Here, we show how the Bellman equation can be greatly simplified. We find an analytical solution for the optimal u given v, and then transform Eq. **1** into a linear equation. Short of solving the entire problem analytically, reducing optimal control to a linear equation is the best one can hope for. This simplification comes at a modest price: although we impose certain structure on the problem formulation, most control problems of practical interest can still be handled. In discrete domains our work has no precursors. In continuous domains there exists related prior work (6–8) that we build on here. Additional results can be found in our recent conference articles (9–11), online preprints (12–14), and supplementary notes [supporting information (SI) *Appendix*].

Results

Reducing Optimal Control to a Linear Problem. We aim to construct a general class of MDPs where the exhaustive search over actions is replaced with an analytical solution. Discrete optimization problems rarely have analytical solutions, thus our agenda calls for continuous actions. This may seem counterintuitive if one thinks of actions as symbols ("go left," "go right"). However, what gives meaning to such symbols are the underlying transition probabilities—which are continuous. The latter observation is key to the framework developed here. Instead of asking the agent to specify symbolic actions, which are then replaced with transition probabilities, we allow the agent to specify transition probabilities $u(x'|x)$ directly. Formally, we have $p(x'|x,u) = u(x'|x)$.

Thus, our agent has the power to reshape the dynamics in any way it wishes. However, it pays a price for too much reshaping, as follows. Let $p(x'|x)$ denote the passive dynamics characterizing the behavior of the system in the absence of controls. The latter will usually be defined as a random walk in discrete domains and as a diffusion process in continuous domains. Note that the notion of passive dynamics is common in continuous domains but is rarely used in discrete domains. We can now quantify how "large" an action is by measuring the difference between $u(\cdot|x)$ and $p(\cdot|x)$. Differences between probability distributions are usually measured via Kullback–Leibler (KL) divergence, suggesting an immediate cost of the form

$$\ell(x,u) = q(x) + \mathrm{KL}(u(\cdot|x)||p(\cdot|x)) = q(x) + \mathrm{E}_{x'\sim u(\cdot|x)}\left[\log \frac{u(x'|x)}{p(x'|x)}\right]. \qquad [2]$$

The state cost $q(x)$ can be an arbitrary function encoding how (un)desirable different states are. The passive dynamics $p(x'|x)$ and controlled dynamics $u(x'|x)$ can also be arbitrary, except that

Author contributions: E.T. designed research, performed research, analyzed data, and wrote the paper.

The author declares no conflict of interest.

This article is a PNAS Direct Submission.

Freely available online through the PNAS open access option.

See Commentary on Page 11429.

[1]E-mail: todorov@cs.washington.edu.

This article contains supporting information online at www.pnas.org/cgi/content/full/0710743106/DCSupplemental.

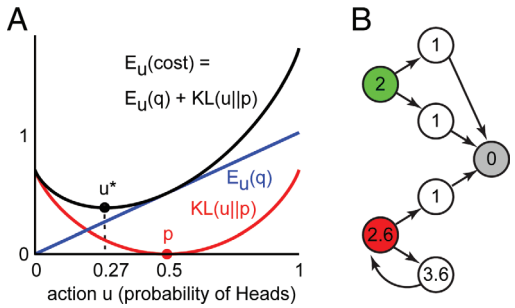

Fig. 1. New problem formulation. (*A*) A coin-toss example where the action corresponds to biasing the coin. The optimal bias is obtained by minimizing the sum (black) of the KL divergence cost (red) and the expected state cost (blue). Note that the tosses are independent and thus the temporal dynamics are irrelevant in this example. (*B*) A stochastic shortest-path problem illustrating how our framework can capture the benefits of error tolerance. See main text.

we require $u(x'|x) = 0$ whenever $p(x'|x) = 0$. This constraint is needed to make KL divergence well-defined. It has the added benefit of preventing the agent from jumping directly to goal states, and more generally from making state transitions that are physically impossible.

Fig. 1 illustrates the construction above with two simple examples. Fig. 1*A* is a coin-toss problem where $q(\text{Tails}) = 0, q(\text{Heads}) = 1$ and the passive dynamics correspond to an unbiased coin. The action u has the effect of biasing the coin. The optimal bias, which turns out to be $u^*(\text{Heads}) = 0.27$, achieves a trade-off between keeping the action cost and the expected state cost small. Note that the controller could have made the coin deterministic by setting $u(\text{Heads}) = 0$, but this is suboptimal because the associated action cost is too large. In general, the optimal actions resulting from our framework are stochastic. Fig. 1*B* is a shortest-path problem where $q = 0$ for the goal state (gray) and $q = 1$ for all other states. The passive dynamics correspond to the random walk on the graph. At the green state it does not matter which path is taken, so the optimal action equals the passive dynamics, the action cost is 0, and the cost-to-go (shown inside the circle) equals the length of the deterministic shortest path. At the red state, however, the optimal action deviates from the passive dynamics to cause a transition up, incurring an action cost of 0.6 and making the red state worse than the green state. In general, $v(x)$ is smaller when the task can be accomplished in multiple ways starting from x. This reflects a preference for error tolerance that is inherent in our framework.

We now return to the theoretical development. The results take on a simpler form when expressed in terms of the *desirability* function

$$z(x) = \exp(-v(x)). \qquad [3]$$

This terminology reflects the fact that z is large at states where the cost-to-go v is small. Substituting Eq. **1** in Eq. **2**, the Bellman equation can be written in terms of z as

$$-\log(z(x)) = q(x) + \min_u \left\{ \mathrm{E}_{x'\sim u(\cdot|x)} \left[\log \frac{u(x'|x)}{p(x'|x)z(x')} \right] \right\}. \qquad [4]$$

The expression being minimized resembles KL divergence between u and pz, except that pz is not normalized to sum to 1. Thus, to obtain proper KL divergence (*SI Appendix*), we have to multiply and divide by the normalization term

$$\mathcal{G}[z](x) = \sum_{x'} p(x'|x)z(x') = \mathrm{E}_{x'\sim p(\cdot|x)}[z(x')]. \qquad [5]$$

Recall that KL divergence achieves its global minimum of zero when the 2 distributions are equal. Therefore, the optimal action u^* is proportional to pz:

$$u^*(x'|x) = \frac{p(x'|x)z(x')}{\mathcal{G}[z](x)}. \qquad [6]$$

This represents the first general class of MDPs where the optimal actions can be found analytically given the optimal costs-to-go. Previously, such results were available only in continuous domains.

The Bellman equation can now be simplified (*SI Appendix*) by substituting the optimal action, taking into account the normalization term and exponentiating. The result is

$$z(x) = \exp(-q(x))\mathcal{G}[z](x). \qquad [7]$$

The expectation $\mathcal{G}[z]$ is a linear operator; thus, Eq. **7** is linear in z. It can be written more compactly in vector notation. Enumerate the states from 1 to n, represent $z(x)$ and $q(x)$ with the n-dimensional column vectors $\mathbf{z}$ and $\mathbf{q}$, and $p(x'|x)$ with the n-by-n matrix P, where the row-index corresponds to x and the column-index to x'. Then Eq. **7** becomes $\mathbf{z} = M\mathbf{z}$, where $M = \mathrm{diag}(\exp(-\mathbf{q}))P$, exp is applied element-wise and diag transforms vectors into diagonal matrices. The latter equation looks like an eigenvector problem, and indeed it can be solved (9) by using the power iteration method $\mathbf{z} \leftarrow M\mathbf{z}$ (which we call Z iteration). However, the problem here is actually simpler because the eigenvalue is 1 and $v(x) = q(x)$ at terminal states. If we define the index sets $\mathcal{T}$ and $\mathcal{N}$ of terminal and nonterminal states and partition $\mathbf{z}$, $\mathbf{q}$, and P accordingly, Eq. **7** becomes

$$(\mathrm{diag}(\exp(\mathbf{q}_\mathcal{N})) - P_{\mathcal{N}\mathcal{N}})\mathbf{z}_\mathcal{N} = P_{\mathcal{N}\mathcal{T}} \exp(-\mathbf{q}_\mathcal{T}). \qquad [8]$$

The unknown $\mathbf{z}_\mathcal{N}$ is the vector of desirabilities at the nonterminal states. It can be computed via matrix factorization or by using an iterative linear solver.

Let us now compare our result (Eq. **8**) with policy iteration (3). We have to solve an equation of the form $A\mathbf{z} = \mathbf{b}$ just once. In policy iteration one has to solve an equation of the form $A_{(\pi)}\mathbf{v} = \mathbf{b}_{(\pi)}$ to evaluate the current policy π; then, the policy has to be improved and the process repeated. Therefore, solving an optimal control problem in our formulation is computationally equivalent to half a step of policy iteration.

Thus far, we have studied MDPs with discrete state spaces. There exists a family of continuous (in space and time) problems related to our MDPs. These problems have stochastic dynamics

$$d\mathbf{x} = \mathbf{a}(\mathbf{x})dt + B(\mathbf{x})(\mathbf{u}dt + \sigma d\boldsymbol{\omega}). \qquad [9]$$

$\boldsymbol{\omega}(t)$ is Brownian motion and σ is the noise amplitude. The cost rate is of the form

$$\ell(\mathbf{x}, \mathbf{u}) = q(\mathbf{x}) + \frac{1}{2\sigma^2}\|\mathbf{u}\|^2. \qquad [10]$$

The functions $q(\mathbf{x})$, $\mathbf{a}(\mathbf{x})$, and $B(\mathbf{x})$ can be arbitrary. This problem formulation is fairly general and standard (but see *Discussion*). Consider, for example, a one-dimensional point with mass m, position x_p, and velocity x_v. Then, $\mathbf{x} = [x_p, x_v]^\mathsf{T}$, $\mathbf{a}(\mathbf{x}) = [x_v, 0]^\mathsf{T}$, and $B = [0, m^{-1}]^\mathsf{T}$. The noise and control signals correspond to external forces applied to the point mass.

Unlike the discrete case where the agent could specify the transition probability distribution directly, here, $\mathbf{u}$ is a vector that can shift the distribution given by the passive dynamics but cannot reshape it. Specifically, if we discretize the time axis in Eq. **9** with step h, the passive dynamics are Gaussian with mean $\mathbf{x} + h\mathbf{a}(\mathbf{x})$ and covariance $h\sigma^2 B(\mathbf{x})B(\mathbf{x})^\mathsf{T}$, whereas the controlled dynamics are Gaussian with mean $\mathbf{x} + h\mathbf{a}(\mathbf{x}) + hB(\mathbf{x})\mathbf{u}$ and the same covariance. Thus, the agent in the continuous setting has less freedom compared with the discrete setting. Yet the two settings share many similarities, as follows. First, the KL divergence between the above Gaussians can be shown to be $\frac{h}{2\sigma^2}\|\mathbf{u}\|^2$, which is just the quadratic energy cost accumulated over time interval h. Second, given the

Table 1. Summary of results for all performance criteria

	Discrete	Continuous
Finite	$\exp(q)z_t = \mathcal{G}[z_{t+1}]$	$qz = \mathcal{L}[z] + \frac{\partial}{\partial t}z$
Total	$\exp(q)z = \mathcal{G}[z]$	$qz = \mathcal{L}[z]$
Average	$\exp(q-c)\widetilde{z} = \mathcal{G}[\widetilde{z}]$	$(q-c)\widetilde{z} = \mathcal{L}[\widetilde{z}]$
Discounted	$\exp(q)z = \mathcal{G}[z^\alpha]$	$qz = \mathcal{L}[z] - z\log(z^\alpha)$

optimal cost-to-go $v(\mathbf{x})$, the optimal control law can be computed analytically (4):

$$\mathbf{u}^*(\mathbf{x}) = -\sigma^2 B(\mathbf{x})^{\mathsf{T}} v_{\mathbf{x}}(\mathbf{x}). \qquad [11]$$

Here, subscripts denote partial derivatives. Third, the Hamilton–Jacobi–Bellman equation characterizing $v(\mathbf{x})$ becomes linear (6, 7) (*SI Appendix*) when written in terms of the desirability function $z(\mathbf{x})$:

$$q(\mathbf{x})z(\mathbf{x}) = \mathcal{L}[z](\mathbf{x}). \qquad [12]$$

The second-order linear differential operator $\mathcal{L}$ is defined as

$$\mathcal{L}[z](\mathbf{x}) = \mathbf{a}(\mathbf{x})^{\mathsf{T}} z_{\mathbf{x}}(\mathbf{x}) + \frac{\sigma^2}{2}\mathrm{trace}(B(\mathbf{x})B(\mathbf{x})^{\mathsf{T}} z_{\mathbf{xx}}(\mathbf{x})). \qquad [13]$$

Additional similarities between the discrete and continuous settings will be described below. They reflect the fact that the continuous problem is a special case of the discrete problem. Indeed, we can show (*SI Appendix*) that, for certain MDPs in our class, Eq. **7** reduces to Eq. **12** in the limit $h \to 0$.

The linear equations **7** and **12** were derived by minimizing total cost in the presence of terminal/goal states. Similar results can be obtained (*SI Appendix*) for all other performance criteria used in practice, in both discrete and continuous settings. They are summarized in Table 1. The constant c is the (unknown) average cost computed as the principal eigenvalue of the corresponding equation. $\widetilde{z}$ is the exponent of the differential cost-to-go function (3) (*SI Appendix*). The constant α is the exponential discount factor. Note that all equations are linear except in the discounted case. The finite-horizon and total-cost results in continuous settings were previously known (6, 7); the remaining six equations were derived in our work.

Applications. The first application is an algorithm for finding shortest paths in graphs (recall Fig. 1*B*). Let $s(x)$ denote the length of the shortest path from state x to the nearest terminal state. The passive dynamics p correspond to the random walk on the graph. The state costs are $q = \rho > 0$ for non-terminal states and $q = 0$ for terminal states. Let $v_\rho(x)$ denote the optimal cost-to-go function for given ρ. If the action costs were 0, the shortest path lengths would be $s(x) = \frac{1}{\rho}v_\rho(x)$ for all ρ. Here, the action costs are not 0, but nevertheless they are bounded. Because we are free to choose ρ arbitrarily large, we can make the state costs dominate the cost-to-go function, and so

$$s(x) = \lim_{\rho\to\infty} \frac{v_\rho(x)}{\rho}. \qquad [14]$$

The construction above involves a limit and does not yield the shortest paths directly. However, we can obtain arbitrarily accurate (modulo numerical errors) approximations by setting ρ large enough. The method is illustrated in Fig. 2 on the graph of internet routers. There is a range of values of ρ for which all shortest path lengths are exactly recovered. Although a thorough comparison with dedicated shortest-path algorithms remains to be done, we suspect that they will not be as efficient as linear solvers. Problems with weighted edges can be handled with the general embedding method presented next.

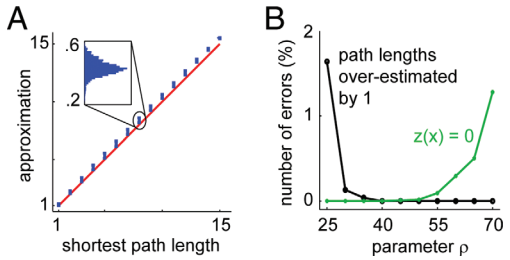

Fig. 2. Approximating shortest paths. The computation of shortest path lengths is illustrated here using the graph of internet routers and their connectivity as of 2003. This dataset is publicly available at www.caida.org. The graph has 190,914 nodes and 609,066 undirected edges. The shortest path length from each node to a specified destination node was computed exactly by using dynamic programming adapted to this problem, and also approximated by using our algorithm with $\rho = 40$. Our algorithm was ≈5 times faster, although both implementations were equally optimized. The exact shortest path lengths were integers between 1 and 15. (*A*) The approximate values were binned in 15 bins according to the corresponding correct value. The range of approximate values in each bin is shown with the blue symbols. The diagonal red line is the exact solution. (*Inset*) Histogram of all values in one bin, with the correct value subtracted. Note that all errors are between 0 and 1, thus rounding down to the nearest integer recovers the exact solution. This was the case for all bins. (*B*) To assess the effects of the free parameter ρ, we solved the above problem 500 times for each of 10 values of ρ between 25 and 70. In each instance of the problem, the set of destination nodes was generated randomly and had between 1 and 5 elements. The approximate shortest path lengths found by our algorithm were rounded down to the nearest integer and compared with the exact solution. The number of mismatches, expressed as a percent of the number of nodes and averaged over the 500 repetitions, is plotted in black. For large values of ρ the approximation becomes exact, as expected from Eq. **14**. However, ρ cannot be set too large, because our algorithm multiplies by $\exp(-\rho)$, thus some elements of z may become numerically zero. The percentage of such numerical errors is plotted in green. There is a comfortable range of ρ where neither type of error is observed.

The second application is a method for continuous embedding of traditional MDPs with symbolic actions. It is reminiscent of linear programming relaxation in integer programming. Denote the symbolic actions in the traditional MDP with a, the transition probabilities with $\widetilde{p}(x'|x,a)$, and the immediate costs with $\widetilde{\ell}(x,a)$. We seek an MDP in our class such that for each (x,a) the action $\widetilde{p}(\cdot|x,a)$ has cost $\widetilde{\ell}(x,a)$:

$$q(x) + \mathrm{KL}(\widetilde{p}(\cdot|x,a)||p(\cdot|x)) = \widetilde{\ell}(x,a). \qquad [15]$$

For each x this yields a system of linear equations in q and $\log p$, which has a unique solution under a mild nondegeneracy condition (*SI Appendix*). If we keep the number of symbolic actions per state fixed while increasing the number of states (for example, by making a grid world larger and larger), the amount of computation needed to construct the embedding scales linearly with the number of states. Once the embedding is constructed, the optimal actions are computed and replaced with the nearest (in the sense of transition probabilities) symbolic actions. This yields an approximately optimal policy for the traditional MDP. We do not yet have theoretical error bounds but have found the approximation to be very accurate in practice. This method is illustrated in Fig. 3 on a machine repair problem adapted from ref. 3. The R^2 between the correct and approximate cost-to-go is 0.993.

The third application is a Monte Carlo method for learning z from experience in the absence of a model of the passive dynamics p and state costs q. Unlike the previous two applications, where we started with traditional MDPs and approximated them with MDPs in our class, here we assume that the problem is already in our class. The linear Bellman Eq. **7** can be unfolded recursively by replacing $z(x')$ with the expectation over the state following x'

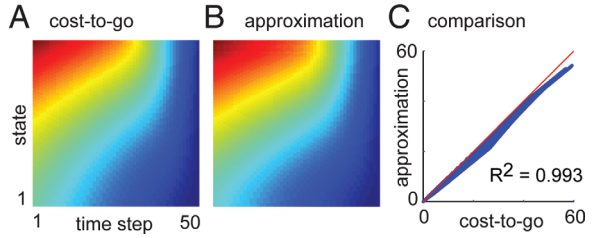

Fig. 3. Embedding traditional MDPs. The continuous embedding of traditional MDPs is illustrated here on a machine repair problem adapted from ref. 3. We have a machine whose state x_t at time t is an integer between 1 and 100. Larger x_t means that the machine is more broken and more costly to operate: we incur an operation cost of $0.02x_t$. There are 50 time steps (this is a finite-horizon problem). The state of the machine has a tendency to deteriorate over time: if we do nothing, x_{t+1} has a probability 0.9 of being one of $x_t \cdots x_t + 8$ (chosen uniformly) and probability 0.1 of being one of $x_t - 9 \cdots x_t - 1$. When x_t is near the edges we clip this distribution and renormalize it to sum to 1. The symbolic actions u_t are integers between 0 and 9 corresponding to how much we invest in repairing the machine in each time step. The repair cost is $0.1u_t$. The effect of the repairs is to circularly left-shift the above transition probability distribution by u_t positions. Thus, $u_t = 0$ corresponds to doing nothing; larger u_t causes larger expected improvement in the state of the machine. (*A*) The traditional MDP described above was embedded within our class, as outlined in the main text and described in detail in (*SI Appendix*). Then, $z_t(x)$ was computed and the approximation $-\log(z)$ to the optimal cost-to-go was plotted. Blue is small, red is large. (*B*) The traditional MDP was also solved by using dynamic programming and the optimal cost-to-go $v_t(x)$ was plotted in the same format as in *A*. (*C*) Scatter plot of the optimal versus approximate costs-to-go at 5,000 space-time points (blue). The R^2 between the two is 0.993, that is, the optimal values account for 99.3% of the variance in the approximate values. The red diagonal line corresponds to the ideal solution. We also computed (via extensive sampling) the performance of the optimal policy found by dynamic programming, the approximately optimal policy derived from our embedding, and a random policy. The performance of our approximation was 0.9% worse than optimal, whereas the performance of the random policy was 64% worse than optimal.

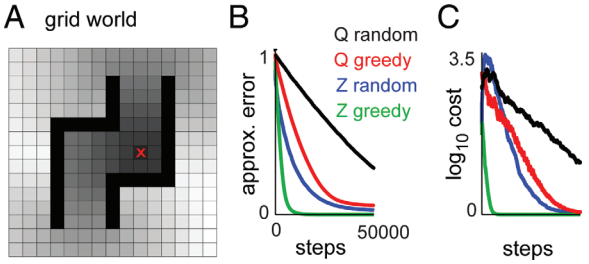

Fig. 4. Z learning and Q learning. Z learning and Q learning are compared in the context of a grid-world MDP. The goal state is marked "X." State transitions are allowed to all neighbors (including diagonal neighbors) and to the current state. Thus, there are at most 9 possible next states, less when the current state is adjacent to the obstacles shown in black or to the edges of the grid. p corresponds to a random walk. All state costs are $q = 1$ except for the goal state where $q = 0$. This MDP is within our class so Z learning can be applied directly. To apply Q learning we first need to construct a corresponding traditional MDP with symbolic actions. This is done as follows. For each state x we define a symbolic action with transition probability distribution matching the optimal $u^*(\cdot|x)$. We also define $N(x) - 1$ other symbolic actions, where $N(x)$ is the number of possible next states following x. Their transition probability distributions are obtained from $u^*(\cdot|x)$ by circular shifting; thus, they have the same shape as u^* but peak at different next states. All these symbolic actions incur cost $q(x) + \mathrm{KL}(u^*(\cdot|x)||p(\cdot|x))$ matching the cost in our MDP. The resulting traditional MDP is guaranteed to have the same optimal cost-to-go as our MDP. (*A*) The grayscale image shows the optimal cost-to-go $v = -\log z$ where z is computed with our model-based method. Darker colors correspond to smaller values. Both Z learning and Q learning aim to approximate this function. (*B*) Error is defined as the average absolute difference between each approximation and the optimal costs-to-go, normalized by the average optimal cost-to-go. All approximations to z are initialized at 0. The learning rate decays as $\eta_t = c/(c+t)$, where $c = 5,000$ for Z learning and $c = 40,000$ for Q learning. Each simulation is repeated 10 times. The state is reset randomly whenever the goal state is reached. Each learning algorithm is tested by using a random policy corresponding to p, as well as a greedy policy. Q learning requires exploration; thus, we use an ε-greedy policy with $\varepsilon = 0.1$. The values of c and ε are optimized manually for each algorithm. Z learning implicitly contains exploration so we can directly use the greedy policy, i.e., the policy $\widehat{u}(x'|x)$ which appears optimal given the current approximation $\widehat{z}$. Greedy Z learning requires importance sampling: the last term in Eq. **17** must be weighted by $p(x_{t+1}|x_t)/\widehat{u}(x_{t+1}|x_t)$. Such weighting requires access to p. (*C*) Empirical performance of the policies resulting from each approximation method at each iteration. Z learning outperforms Q learning, and greedy methods outperform random sampling.

and so on, and then pushing all state costs inside the expectation operators. This yields the path-integral representation

$$z(x) = \mathrm{E}_{x_{t+1} \sim p(\cdot|x_t)} \left[\exp\left(-\sum_{t=0}^{t_f} q(x_t) \right) \right]. \qquad [16]$$

$(x_0, x_1 \cdots x_{t_f})$ are trajectories initialized at $x_0 = x$ and sampled from the passive dynamics, and t_f is the time when a goal state is first reached. A similar result holds in continuous settings, where the Feynman–Kac theorem states that the unique positive solution to Eq. **12** has a path-integral representation. The use of Monte Carlo methods for solving continuous optimal control problems was pioneered in ref. 7. Our result (Eq. **16**) makes it possible to apply such methods to discrete problems with non-Gaussian noise.

The fourth application is related to the path-integral approach above but aims to achieve faster convergence. It is motivated by Reinforcement Learning (2) where faster convergence is often observed by using Temporal Difference (TD) methods. A TD-like method for our MDPs can be obtained from Eq. **7**. It constructs an approximation $\widehat{z}$ using triplets (x_t, q_t, x_{t+1}). Note that measuring the action costs is not necessary. $\widehat{z}$ is updated online as follows:

$$\widehat{z}(x_t) \leftarrow (1 - \eta_t)\,\widehat{z}(x_t) + \eta_t \exp(-q_t)\,\widehat{z}(x_{t+1}). \qquad [17]$$

η_t is a learning rate which decreases over time. We call this algorithm Z learning. Despite the resemblance to TD learning, there is an important difference. Our method learns the optimal cost-to-go directly, whereas TD methods are limited to learning the cost-to-go of a specific policy—which then needs to be improved, the cost-to-go relearned, and so on. Indeed Z learning is an off-policy method, meaning that it learns the cost-to-go for the optimal policy while sampling according to the passive dynamics. The only Reinforcement Learning method capable of doing this is Q learning (15). However, Q learning has the disadvantage of operating in the product space of states and actions, and is therefore less efficient. This is illustrated in Fig. 4 on a navigation problem.

The fifth application accelerates MDP approximations to continuous problems (16). Such MDPs are obtained via discretization and tend to be very large, calling for efficient numerical methods. Because continuous problems of the form (Eqs. **9** and **10**) are limits of MDPs in our class, they can be approximated with MDPs in our class, which, in turn, are reduced to linear equations and solved efficiently. The same continuous problems can also be approximated with traditional MDPs and solved via dynamic programming. Both approximations converge to the same solution in the limit of infinitely fine discretization, and turn out to be equally accurate away from the limit, but our approximation is faster to compute. This is shown in Fig. 5 on a car-on-a-hill problem, where our method is $\approx$10 times faster than policy iteration and 100 times faster than value iteration.

The remaining applications have been developed recently. Below we summarize the key results and refer the reader to online preprints (12–14). The sixth application is a deterministic method for computing the most likely trajectory of the optimaly controlled stochastic system. Combining Eqs. **6** and **7**, the optimal control law can be written as $u^*(x'|x) = \exp(-q(x))p(x'|x)\frac{z(x')}{z(x)}$. Given a fixed

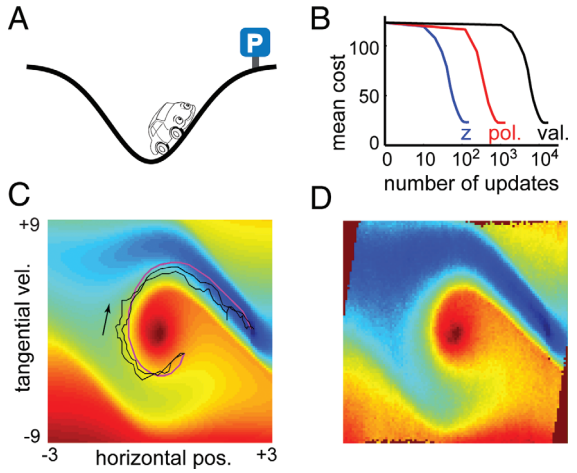

Fig. 5. Continuous problems. Comparison of our MDP approximation and a traditional MDP approximation on a continuous car-on-a-hill problem. (*A*) The car moves along a curved road in the presence of gravity. The control signal is tangential acceleration. The goal is to reach the parking lot with small velocity. (*B*) Z iteration (blue), policy iteration (red), and value iteration (black) converge to control laws with identical performance; however, Z iteration is 10 times faster than policy iteration and 100 times faster than value iteration. Note the log-scale on the horizontal axis. (*C*) The optimal cost-to-go for our approximation. Blue is small, red is large. The two black curves are stochastic trajectories resulting from the optimal control law. The thick magenta curve is the most likely trajectory of the optimally controlled stochastic system. It is computed by solving the deterministic optimal control problem described in the main text. (*D*) The optimal cost-to-go is inferred from observed state transitions by using our algorithm for inverse optimal control. Brown pixels correspond to states where we did not have data (i.e., no state transitions landed there); thus, the cost-to-go could not be inferred. The details are given in (*SI Appendix*).

initial state x_0 and final time T, the probability that the optimal control law u^* generates trajectory $x_1, x_2, \cdots x_T$ is

$$\prod_{t=0}^{T-1} u^*(x_{t+1}|x_t) = \frac{z(x_T)}{z(x_0)} \prod_{t=0}^{T-1} \exp(-q(x_t))p(x_{t+1}|x_t). \qquad [18]$$

Omitting $z(x_0)$, which is fixed, and noting that $z(x_T) = \exp(-q(x_T))$, the negative log of Eq. **18** can be interpreted as the cumulative cost for a deterministic optimal control problem with immediate cost $q(x) - \log(p(x'|x))$. A related result is obtained in continuous settings when B is constant: the corresponding deterministic problem has dynamics $\dot{\mathbf{x}} = \mathbf{a}(\mathbf{x}) + B\mathbf{u}$ and cost $\ell(\mathbf{x}, \mathbf{u}) + \frac{1}{2}\mathrm{div}(\mathbf{a}(\mathbf{x}))$. These results are important because optimal trajectories for deterministic problems can be found with efficient numerical methods (17) that avoid the curse of dimensionality. Our framework makes it possible to apply deterministic methods to stochastic control for the first time. The most likely trajectory in the car-on-a-hill problem is shown in Fig. 5*C* in magenta. See ref. 12 for details.

The seventh application exploits the duality between Bayesian estimation and stochastic optimal control. This duality is well-known for linear systems (4) and was recently generalized (8, 10) to nonlinear continuous systems of the form Eqs. **9** and **10**. Duality also holds for our MDPs. Indeed, Eq. **18** can be interpreted as the posterior probability of a trajectory in a Bayesian smoothing problem, where $p(x_{t+1}|x_t)$ are the prior probabilities of the state transitions and $\exp(-q(x_t))$ are the likelihoods of some unspecified measurements. The desirability function in the control problem can be shown to be proportional to the backward filtering density in the dual estimation problem. Thus, stochastic optimal control problems can be solved by using Bayesian inference. See refs. 10 and 12 for details.

The eighth application concerns inverse optimal control, where the goal is to infer $q(x)$ given a dataset $\{x_k, x'_k\}_{k=1\cdots K}$ of state transitions generated by the optimal controller. Existing methods rely on guessing the cost function, solving the forward problem, comparing the solution with data, and improving the guess and iterating (18, 19). This indirect procedure can be inefficient when the forward problem is hard to solve. For our MDPs the inverse problem can be solved directly, by minimizing the convex function

$$L(\mathbf{v}) = \mathbf{d}^{\mathsf{T}}\mathbf{v} + \mathbf{c}^{\mathsf{T}} \log(P \exp(-\mathbf{v})), \qquad [19]$$

where $\mathbf{v}$ is the vector of (unknown) optimal costs-to-go, P is the passive dynamics matrix defined earlier, and $\mathbf{d}$ and $\mathbf{c}$ are the histograms of x'_k and x_k (i.e., d_i is the number of times that $x'_k = i$). It can be shown that $L(\mathbf{v})$ is the negative log-likelihood of the dataset. We have found empirically that its Hessian tends to be diagonally dominant, which motivates an efficient quasi-Newton method by using a diagonal approximation to the Hessian. Once the minimum is found, we can compute $\mathbf{z}$ and then find $\mathbf{q}$ from the linear Bellman equation. Fig. 5*D* illustrates this inverse optimal control method on the car-on-a-hill problem. See ref. 14 for details.

The ninth application is a method for constructing optimal control laws as combinations of certain primitives. This can be done in finite-horizon as well as total-cost problems with terminal/goal states, where the final cost plays the role of a boundary condition. Suppose we have a collection of component final costs $g_i(\mathbf{x})$ for which we have somehow computed the desirability functions $z_i(\mathbf{x})$. Linearity implies that, if the composite final cost $g(\mathbf{x})$ satisfies

$$\exp(-g(\mathbf{x})) = \sum_i w_i \exp(-g_i(\mathbf{x})) \qquad [20]$$

for some set of w_i, then the composite desirability function is $z(\mathbf{x}) = \sum_i w_i z_i(\mathbf{x})$. This approach is particularly useful when the component problems can be solved analytically, as in the linear-quadratic-Gaussian (LQG) framework (4). In that case the component desirabilities $z_i(\mathbf{x})$ are Gaussians. By mixing them linearly, we can obtain analytical solutions to finite-horizon problems of the form Eqs. **9** and **10** where $\mathbf{a}(\mathbf{x})$ is linear, B is constant, $q(\mathbf{x})$ is quadratic; however, the final cost $g(\mathbf{x})$ is not constrained to be quadratic. Instead $g(\mathbf{x})$ can equal the negative log of any Gaussian mixture. This is a nontrivial extension to the widely used LQG framework. See ref. 13 for details.

Discussion

We formulated the problem of stochastic optimal control in a way which is rather general and yet affords substantial simplifications. Exhaustive search over actions was replaced with an analytical solution and the computation of the optimal cost-to-go function was reduced to a linear problem. This gave rise to efficient new algorithms speeding up the construction of optimal control laws. Furthermore, our framework enabled a number of computations that were not possible previously: solving problems with nonquadratic final costs by mixing LQG primitives, finding the most likely trajectories of optimally controlled stochastic systems via deterministic methods, solving inverse optimal control problems via convex optimization, quantifying the benefits of error tolerance, and applying off-policy learning in the state space as opposed to the state-action space.

These advances were made possible by imposing a certain structure on the problem formulation. First, the control cost must be a KL divergence—which reduces to the familiar quadratic energy cost in continuous settings. This is a sensible way to measure control energy and is not particularly restrictive. Second, the controls and the noise must be able to cause the same state transitions; the analog in continuous settings is that both the controls and the

noise act in the subspace spanned by the columns of $B(\mathbf{x})$. This is a more significant limitation. It prevents us from modeling systems subject to disturbances outside the actuation space. We are now pursuing an extension that aims to relax this limitation while preserving many of the appealing properties of our framework. Third, the noise amplitude and the control costs are coupled, and, in particular, the control costs are large when the noise amplitude is small. This can be compensated to some extent by increasing the state costs while ensuring that $\exp(-q(x))$ does not become numerically zero. Fourth, with regard to Z learning, following a policy other than the passive dynamics p requires importance-sampling correction based on a model of p. Such a model could presumably be learned online; however, this extension remains to be developed.

This framework has many potential applications and we hope that the list of examples will grow as other researchers begin to use it. Our current focus is on high-dimensional continuous problems such as those arising in biomechanics and robotics, where the discrete-time continuous-state MDP approximation is particularly promising. It leads to linear integral equations rather than differential equations, resulting in robust numerical methods. Furthermore it is well suited to handle discontinuities due to rigid-body collisions. Initial results using mixture-of-Gaussian approximations to the desirability function are encouraging (11), yet a lot more work remains.

ACKNOWLEDGMENTS. We thank Surya Ganguli, Javier Movellan, and Yuval Tassa for discussions and comments on the manuscript. This work was supported by the National Science Foundation.

1. Todorov E (2004) Optimality principles in sensorimotor control. *Nat Neurosci* 7(9):907–915.
2. Sutton R, Barto A (1998) *Reinforcement Learning: An Introduction* (MIT Press, Cambridge MA).
3. Bertsekas D (2001) *Dynamic Programming and Optimal Control* (Athena Scientific, Bellmont, MA), 2nd Ed.
4. Stengel R (1994) *Optimal Control and Estimation* (Dover, New York).
5. Korn R (1997) *Optimal Portfolios: Stochastic Models for Optimal Investment and Risk Management in Continuous Time* (World Scientific, Teaneck, NJ).
6. Fleming W, Mitter S (1982) Optimal control and nonlinear filtering for nondegenerate diffusion processes. *Stochastics* 8:226–261.
7. Kappen HJ (2005) Linear theory for control of nonlinear stochastic systems. *Phys Rev Lett* 95:200201.
8. Mitter S, Newton N (2003) A variational approach to nonlinear estimation. *SIAM J Control Opt* 42:1813–1833.
9. Todorov E (2006) Linearly-solvable Markov decision problems. *Adv Neural Information Proc Syst* 19:1369–1376.
10. Todorov E (2008) General duality between optimal control and estimation. *IEEE Conf Decision Control* 47:4286–4292.
11. Todorov E (2009) Eigen-function approximation methods for linearly-solvable optimal control problems, *IEEE Int Symp Adapt Dynam Programm Reinforce Learning* 2:161–168.
12. Todorov E (2009) *Classic Maximum Principles and Estimation-Control Dualities for nonlinear Stochastic Systems*, preprint.
13. Todorov E (2009) *Compositionality of Optimal Control Laws*, preprint.
14. Todorov E (2009) *Efficient Algorithms for Inverse Optimal Control*, preprint.
15. Watkins C, Dayan P (1992) Q-learning. *Mach Learn* 8:279–292.
16. Kushner H, Dupuis P (2001) *Numerical Methods for Stochastic Optimal Control Problems in Continuous Time* (Springer, New York).
17. Bryson A, Ho Y (1969) *Applied Optimal Control* (Blaisdell Publishing Company, Waltham, MA).
18. Ng A, Russell S (2000) Algorithms for inverse reinforcement learning. *Int Conf Mach Learn* 17:663–670.
19. Liu K, Hertzmann A, Popovic Z (2005) Learning physics-based motion style with nonlinear inverse optimization. *ACM Trans Graphics* 24:1071–1081.