Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence

Open-Loop Planning in Large-Scale Stochastic Domains

Ari Weinstein * and **Michael L. Littman**+
Rutgers University / +Brown University
aweinst@cs.rutgers.edu, mlittman@cs.brown.edu

Abstract

We focus on effective sample-based planning in the face of underactuation, high-dimensionality, drift, discrete system changes, and stochasticity. These are hallmark challenges for important problems, such as humanoid locomotion. In order to ensure broad applicability, we assume domain expertise is minimal and limited to a generative model. In order to make the method responsive, computational costs that scale linearly with the amount of samples taken from the generative model are required. We bring to bear a concrete method that satisfies all these requirements; it is a receding-horizon open-loop planner that employs cross-entropy optimization for policy construction. In simulation, we empirically demonstrate near-optimal decisions in a small domain and effective locomotion in several challenging humanoid control tasks.

Humanoid locomotion tasks are difficult to plan in effectively due to a number of properties. Because humanoids have high-dimensional state and action spaces, methods that are effective in smaller domains may fail due to the curse of dimensionality. Dynamic legged walking has characteristics that violate assumptions made by traditional motion planning approaches (Ladd and Kavraki 2005). Discrete system changes are common in locomotion and occur in any domain where hard contacts or joint limits exist, which produces nonsmooth and nondifferentiable dynamics. In addition to the properties listed above, we are also concerned with the setting where stochasticity exists in the system dynamics, which violates any assumption of determinism. Characteristics of the policy space defining behavior in humanoid locomotion tasks introduce further difficulties, as the subspace of effective policies is very small in the entire space of policies (Erez 2011), and the landscape of policies with respect to their quality has many local optima (Erez, Tassa, and Todorov 2011).

Many planning algorithms require large amounts of expert knowledge to function. Common examples are knowledge of inverse kinematics, or shaping functions that approximate a value function. Aside from limiting applicability, these requirements may introduce fragility as failure is risked if the provided information is inaccurate. To make these planning algorithms general, simple to implement, and robust, we are concerned with approaches that require minimal domain knowledge, limited to a generative model (equivalent to a simulator). Because time sensitivity is important, we consider methods that parallelize simply and have computational cost linear in the number of samples taken from the generative model.

The approach used here plans in an open-loop manner, using cross-entropy (Rubinstein 1997) to optimize a sequence of actions with respect to the cumulative numeric reward obtained by the policy in simulation. As such, it is state agnostic and does not reason directly about dynamics (including whether or not they are deterministic, stochastic, continuous, or discontinuous), and is not directly impacted by the number of state dimensions in the domain. The approach requires only a generative model and has computational cost linear in the number of samples from the generative model (depending on the representation of the policy, costs may be super-linear in the planning horizon, although this value is independent of the samples from the generative model). As we demonstrate empirically in simulation, the method achieves near-optimal results in a deterministic small domain, and successfully produces plans for humanoid locomotion in several settings, including domains with stochasticity as well as erroneous generative models.

Background

This section introduces background material needed to apply an optimization algorithm to a sequential planning problem.

Cross-Entropy Optimization

Originally designed for rare event simulation (Rubinstein 1997), the cross-entropy method (CE) was later extended to perform global optimization by casting high value points as the rare event of interest (Rubinstein 1999). While the algorithm can be described generally at the cost of simplicity (Boer et al. 2005), we focus the version described in Algorithm 1. Briefly, the algorithm functions iteratively by: sampling a set of points $a_1 ... a_n$ from the distribution p, based on its current parameterization Φ_{g-1} (line 3); assigning values to $v_1 ... v_n$ to $a_1 ... a_n$ according to the (potentially stochastic) evaluation function f (line 4); selecting the

*This research was supported in part by National Science Foundation DGE-0549115.

ρ fraction of "elite" samples (lines 5 and 6); and then computing the new parameterization of p, $\hat{\Phi}_g$ used in the next iteration based on the elite samples (line 6).

Algorithm 1 Cross-entropy

1: **function** OPTIMIZE($p, \hat{\Phi}_0, f, \rho, n, G$)
2: **for** $g = 1 \rightarrow G$ **do**
3: $a_1 ... a_n \sim p(\cdot|\hat{\Phi}_{g-1}), \mathbf{a} \leftarrow a_1 ... a_n$
4: $v_1 ... v_n \leftarrow f(a_1) ... f(a_n), \mathbf{v} \leftarrow v_1 ... v_n$
5: sort **a** according to **v**
6: $\hat{\Phi}_g \leftarrow \text{argmax}_{\hat{\Phi}} \prod_{i=1}^{\lceil n\rho \rceil} p(a_i|\hat{\Phi})$
return a_1

The algorithm requires several items to be specified *a priori*. First among them is the type of distribution p, which defines main characteristics of how **a** is drawn. Ideally, $p(\cdot|\hat{\Phi}_0)$ would be the distribution that generates optimal samples in the domain (although, if that were known, no optimization would be necessary). Since, generally, this distribution is not known, or is difficult to sample from, other distributions are used. When domain expertise is limited, it should be ensured that $p(\cdot|\hat{\Phi}_0)$ has good support over the entire sample space. The update rule for the parameter vector $\hat{\Phi}_g$ in line 6 is defined as the maximum likelihood estimate for producing the elite samples in the current generation. In the case that p is a multivariate Gaussian, $\hat{\Phi}_g$ is updated according to:

$$\mu_g = \frac{\sum_{i=1}^{\lceil n\rho \rceil} X_i}{\lceil n\rho \rceil}$$
$$\sigma_g^2 = \frac{\sum_{i=1}^{\lceil n\rho \rceil} (X_i - \mu_g)^T (X_i - \mu_g)}{\lceil n\rho \rceil}$$
$$\hat{\Phi}_g = \langle \mu_g, \sigma_g^2 \rangle.$$

In this case, computing $\hat{\Phi}_g$ has a polynomial cost in the dimension of p, which is assumed to be independent of n and G. In practice, the costs for computing $\hat{\Phi}_g$ are not computationally significant.

There are a number of other parameters for CE. The parameter ρ determines the proportion of samples that are selected as elite, and is important because it impacts the rate at which $\hat{\Phi}_g$ changes from generation to generation. The variable n defines the size of each generation; it is important that this number be large enough so that the early generations have a good chance of sampling near the optimum. The number of generations evaluated by the algorithm is defined by G.

These parameters must all be set sensibly to find a good solution for the domain of interest. While doing so, tradeoffs in solution quality, computational requirements, convergence rates, and robustness must all be considered. To simplify the application of CE to various domains, there are methods that automatically adjust (or remove) these parameters. While CE employs an all-or-nothing approach over samples (defining the top ρ as elite and discarding the rest), ρ can be removed by weighing samples continuously, placing it in a broader family of other algorithms (Stulp and Sigaud 2012). While we use a fixed number of generations in optimization, another common method is to examine the stability of $\hat{\Phi}$ or **v** from generation to generation, and terminate when the change of the variable drops below a threshold. The fully automated cross entropy method further reduces the need to manually define parameters to CE by adjusting them automatically during execution (Boer et al. 2005).

CE has a number of beneficial properties. By attempting to perform global optimization, it avoids convergence to local optima. Because humanoid walking has many local optima that correspond to poor solutions, local search methods require expertise in the form of shaping functions to initialize search near the global optimum. Another property of CE is that its computational costs are linear in the number of samples Gn, and the method parallelizes trivially within a generation, meaning the optimization itself is not computationally intensive.

To our knowledge, there are no guarantees in terms of the rate of convergence or sample complexity. While proofs have shown convergence of CE to optimal results in both discrete (Costa, Jones, and Kroese 2007) and continuous domains (Margolin 2005), the conditions required are fairly strong and are violated in the experiments discussed here. Therefore, we are unable to make claims of optimality, even at the limit, for the CE in the settings considered.

Markov Decision Processes

A Markov decision process (MDP) M is described by a five-tuple $\langle S, A, T, R, \gamma \rangle$, where $S \subseteq \mathbb{R}^{|S|}$ is the state space ($|S|$ is the number of dimensions in the state space). $A \subseteq \mathbb{R}^{|A|}$ is the action space, T is the transition distribution, with $T(s'|s, a)$ denoting the distribution over next states s' induced by taking action a in state s. The deterministic reward function $R(s, a) \rightarrow \mathbb{R}$ defines the immediate reward for taking action a in state s, and $\gamma \in [0, 1)$ is the discount factor.

A policy dictates how the agent behaves in an MDP. We refer to $\pi : S \rightarrow A$ as a closed-loop policy, and $\pi : \mathbb{Z}^+ < H \rightarrow A$ as an open-loop policy. Since we are primarily concerned with open-loop planning, we will refer to an open-loop policy simply as a "policy." In this work, we are concerned with finding policies that maximize the finite horizon discounted cumulative reward, $\sum_{t=0}^{n} \gamma^t r_t$ which we refer to as the *return*. The only domain knowledge assumed here is a generative model, which provided with any (s, a) returns $r = R(s, a)$ and $s' \sim T(s, a)$. The generative model *only* allows sampling from R and T, and does grant access to information about them, such as their form.

Local Planning with Cross-Entropy

A major distinction exists between two approaches to planning, which we refer to as *global* and *local*. In global planning, the objective is to find a closed-loop policy that covers the entire state space of the MDP $\pi : S \rightarrow A$. CE has been used to perform global planning in both discrete and continuous MDPs by optimizing over parameters specifying of a policy (Stulp and Sigaud 2012).

One limitation of global planning is that the function representing π must be both expressive enough to represent ef-

fective closed-loop policies, and simple enough that an effective parameterization can be found quickly. Additionally, because global planners produce a mapping from all points in the state space to an action, such methods must consider the entire state space of the problem when producing a policy. In the absence of domain expertise, these factors combine to make the approach difficult to utilize in high dimensional domains requiring precise control, such as humanoid locomotion (Erez 2011).

Instead of developing a policy for all states simultaneously, local planners only attempt to develop a policy for a single query state (and perhaps a region around it) at any point in time $\pi : S' \subseteq S \rightarrow A$. Most commonly, local planners operate according to a receding-horizon, meaning that at each time step, planning is conducted to a maximum of H of steps into the future. By doing so, costs depend on the planning horizon H, instead of $|S|$, sidestepping the curse of dimensionality (Kearns, Mansour, and Ng 1999). One common way of performing local planning is is by the use of *rollouts*, which are length-H sequences of states, actions, and rewards (Tesauro and Galperin 1996).

Cross-entropy has been used for planning in a number of complex domains. A notable success of CE in global planning was the construction of an effective value function approximator for the challenging benchmark problem, Tetris (Szita and Lőrincz 2006). In applications more similar to what we consider here, CE has been used for local planning in MDPs in domains as complicated as simulated helicopter control and navigation (Kobilarov 2011). Additionally, CE has been been used in place of a uniform distribution to improve the performance of rapidly exploring random trees (Kobilarov 2012). A variant of covariance matrix adaptation evolution strategy, a method related to CE, has also been used for designing both skeletons and gaits (Wampler and Popović 2009), although it leverages more domain knowledge than the approach presented here.

Any general optimization algorithm, including CE, can be used to perform open-loop receding-horizon planning. An application of CE optimization to sequential decision making in an MDP is shown in Algorithm 2, and forms the algorithm of interest in this work.

Applying CE to local planning does not require any changes to the algorithm itself; only the manner it is used changes. The evaluation function f now performs a complete rollout according to the action sequence and evaluates the sequence according to its return (lines 4, 9-16); parameters necessary for the rollout are bound by the agent, and f is left to accept a vector a (encoding the open-loop policy) as a parameter during optimization in CE (line 4). For the fixed planning horizon H, optimization is now conducted over length-H action sequences (vectors of length $|A|H$) with respect to their returns (line 5). After optimization is complete, the first action in the sequence (line 6) is performed in the true domain (line 7), and the process repeats anew from the resulting state (line 8).

Using CE for local planning has a number of properties worth discussing. The form of planning discussed here performs *global* optimization, but *local* planning. Both properties reduce the need for domain expertise, as global optimization removes the need for shaping, while local planning removes the need for carefully constructed functions representing closed-loop policies.

Algorithm 2 Open-Loop Planning with CE

```
1: function AGENT(M, H, s_0, p, Φ̂_0, ρ, n, G)
2:     s ← s_0
3:     loop
4:         f ← λa. ROLLOUT(a, s, 0, H, M)
5:         a ← OPTIMIZE(p, Φ̂_0, f, ρ, n, G)
6:         a' ← a_1...a_|A|
7:         s' ~ T(s, a')
8:         s ← s'
9: function ROLLOUT(a, s, h, H, M)
10:    if h ≥ H then
11:        return 0
12:    R ← M_R, T ← M_T, A ← M_A, γ ← M_γ
13:    a' ← a_{h|A|}...a_{(h+1)|A|}
14:    r ← R(s, a')
15:    s' ~ T(·|s, a')
16:    return r + γ ROLLOUT(s', a, h + 1, H, M)
```

Since **v** is only a result of actual returns observed in the environment, the algorithm cannot produce divergent estimates of policy quality, which occurs among algorithms that extrapolate estimates (Gordon 1995), and does so while producing high quality policies. Because open-loop planners do not consider state, they are state agnostic, and have no dependence on state in their planning costs, and operate identically in continuous, discrete, hybrid, and partially observable state spaces (assuming a reset to the start state is possible in the generative model). On the other hand, because the approach does not consider state, stochastic domains with certain structures can cause convergence to suboptimal policies (Weinstein and Littman 2012), although we have never observed this issue in practice. Because function approximators are not used to represent policies, any H-step sequence of actions can be represented.

Related Work

In this section, we discuss a number of general purpose planning methods for continuous Markov decision processes that have similar properties to CE.

Hierarchical Open-Loop Optimization

Hierarchical open-loop optimization (HOLOP) also uses an optimization algorithm to perform open-loop planning (Weinstein and Littman 2012). Instead of CE, HOLOP uses hierarchical optimistic optimization (HOO) to perform optimization (Bubeck and Munos 2010). The primary advantages of HOO as an optimizer are its tight performance guarantees and built-in exploration policy. Based on the metric of regret (the cumulative difference between selected actions and optimal), HOO is optimal, with a rate of $\tilde{O}(\sqrt{T})$, where T is the total number of actions taken (Bubeck et al. 2008). Although the regret rate is independent of $|A|$ and H, it only holds after $T >> |A|H$.

Although HOLOP has excellent theoretical properties, it has characteristics that make planning slow in practice. Unlike CE, the method does not parallelize simply, as each choice is based on the entire proceeding history. The fact that all previous data is examined during each query also makes the computational cost polynomial in the number of rollouts performed (although a variant reduces the cost to $T \log T$). Additionally, the impact of new data is smaller in HOO. Whereas CE updates the distribution over all dimensions over the entire sample space after each generation, in HOO each new sample can only refine the policy in one region of one dimension. Both of these issues mean that in practice refining the policy takes a great deal of time and samples for HOO as compared to CE. Essentially, while HOO has better theoretical guarantees, CE is more useful in larger domains or when computation time is a concern.

Differential Dynamic Programming

Differential dynamic programming (DDP), a form of control related to model predictive control, has been used for successful control in helicopter flight as well as humanoid locomotion (Abbeel et al. 2007; Erez, Tassa, and Todorov 2011). Although DDP also optimizes return with respect to sequences of actions, it is not state agnostic, as it factors in state information with the assumption of quadratic dynamics in the domain. DDP performs local optimization, which introduces the need for domain expertise in the form of shaping in domains with many local optima (Erez 2011).

In domains with discrete system changes, stochasticity must be artificially introduced to smooth domains with discontinuities in the dynamics. This noise becomes another parameter that must be controlled, because it is desirable to add as little noise as possible, but introducing too little noise may result in failure (Tassa and Todorov 2010). Another result of the violation of these assumptions is that regularization must used to prevent divergence by ensuring small policy changes between generations. Controlling the regularization variable leads to a trade off between safety from divergence and speed of convergence (Erez 2011).

It should be stressed that DDP has had significant successes in a number of complex domains. We discuss the limitations of this approach because the proper selection of shaping, noise, and control of regularizers requires domain expertise, and can make the approach difficult to apply.

Empirical Results

We present the performance of CE in a simple domain and in several humanoid locomotion domains. The differences between these two settings show the generality of the approach; we show near-optimal performance in the small domain, as well as acceptable policies in the humanoid locomotion domain.

Double Integrator

The double integrator (Santamaría, Sutton, and Ram 1996), models the motion of an object along a surface. The state is represented by a position p and velocity v. The action, a, sets the acceleration of the object. $R(p, a) = -(p^2 + a^2)$,

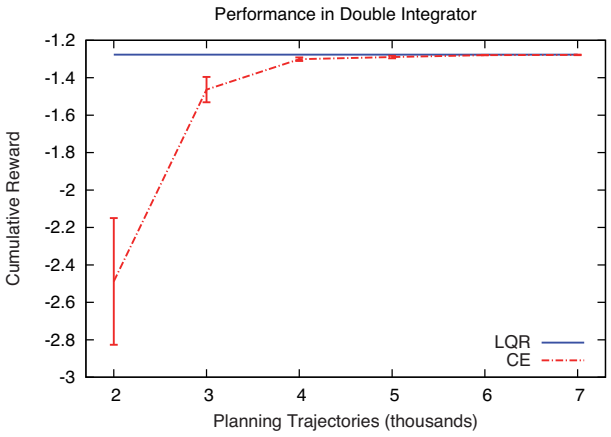


Figure 1: Performance of planning in the double integrator with cross-entropy compared to optimal.

and $s_0 = (p, v) = (0.95, 0)$. Because the domain has linear dynamics and a quadratic reward function, standard control theory techniques allow a linear quadratic regulator to be computed (LQR) that produces the optimal continuous-time undiscounted infinite-horizon closed-loop policy. To accurately evaluate the performance of LQR and CE, all aspects of the domain are deterministic. Each episode lasts for 100 time steps, with a time step for integration of 0.05. Episodes are long enough for both approaches to bring the object close to the origin, so the results are representative of the cumulative reward until stabilization.

The cumulative reward per episode obtained by CE for varying numbers of trajectories (Gn) per planning step, as well as the LQR solution appear in Figure 1. In all cases, $H = 30$, $\rho = 0.1$, $G = 30$, with n constant across generations in each trial. The mean cumulative reward of CE with $Gn = 7,000$ is not statistically significantly different from optimal ($p < 0.05$). In fact, in some episodes the solution found by CE is slightly better than the optimal continuous time LQR solution, due to time discretization (CE, however, would be unable to outperform discrete time LQR). With $Gn = 7,000$, each planning step took less than 0.1 seconds on an Intel Core i7-3930k.

A visualization of the planning performed by CE in the double integrator from s_0 is rendered in Figure 2. Alternating in shades of red and blue, trajectories are grouped according to increasing generation during optimization, in planes of increasing height along the vertical axis. These planes correspond to the trajectories developed in generations 0, 7, 14, 21, and 28. The policy at generation 0 (lowest on the vertical axis) is simply a random walk in the domain; planning is initialized with each action in the sequence represented by independent Gaussians with $\mu = 0$ and $\sigma = 3$. The trajectory rendered highest along the vertical axis in black is the trajectory selected by LQR. The start state is represented by the vertical cyan line. As can be seen, the basic form of the policy forms quickly, with the later generations performing minute refinements.

It is worth mentioning that the trajectories from gener-

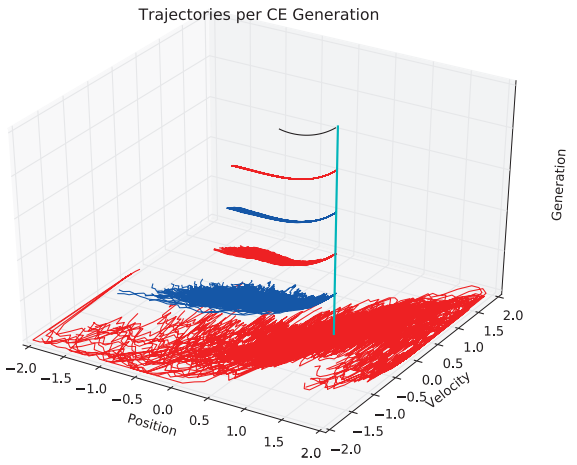


Figure 2: Trajectories from selected generations in cross-entropy applied to the double integrator from the episode start state.

ation 28 and the LQR policy (2nd to highest, and highest, respectively) are initially similar, but differ later in the trajectories. While differences from the LQR policy should be regarded as errors, CE can still achieve very high performance due to the fact that it uses receding-horizon planning. Because planning is completely restarted at each step, the planner only needs to produce a near-optimal *initial* action.

Humanoid Walking

In this series of experiments, we demonstrate the ability of open-loop planning by CE to produce effective strategies for humanoid locomotion in settings that are deterministic, stochastic, or contain a flawed generative model.

As modeled here, the walker's planar body consists of a pair of legs, a pair of arms, and a torso. The legs have upper and lower segments, while the arms and torso are each made of one segment. At each of the 7 joints (controlling the lower legs, upper legs, torso, and arms), a desired angular velocity is set a teach time step, so the action space for planning is these 7 values. There is no actuated ankle, so the body is underactuated when a leg is in contact with the ground. In a compact representation, the state is 18 dimensions, as one part of the body must be specified in terms of location and velocity, and all other points can be described in terms of its angle and angular velocity from the joint it connects to. The reward function is the velocity of the hip along the x-axis. Any state where a part of the body other than a foot contacts the floor is terminal with a large penalty. $\rho = 0.2$ and CE is allowed 10,000 trajectories of planning per time step distributed evenly across with $G = 30$, and $H = 30$. In this setting, planning took approximately 130 seconds per step on an Intel Core i7-3930k. Aside from parallelization, the simulation and planning algorithms were not optimized, with the simulation run in PyODE (PyODE 2010).[1] The sampling distribution p is a multivariate Gaussian.

In our simplest experiment, called the deterministic walker, we demonstrate the ability of CE to find an effective gait when the domain is deterministic. A stroboscopic rendering of the policy is shown in Figure 3(a), with locomotion going from right to left. It is worth noting that the gait is highly dynamic; there are entire classes of planning devoted to legged locomotion that are unable to produce this type of behavior, such as classical zero moment point controllers. As compared to such controllers, those that can move dynamically move both appear more natural and are more efficient (Manchester et al. 2009). The increasing distance between each successive rendering from right to left indicates that the agent is accelerating throughout the experiment, increasing reward per time step as the episode progresses.

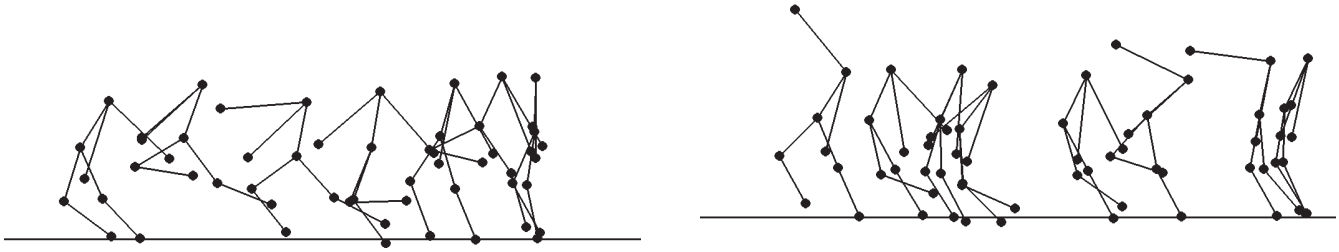
In the next experiment, called the stochastic walker, the basic setting is expanded by introducing stochasticity in the form of by random "pushes". These pushes are applied to one joint selected uniformly at random at each time step, are of a uniformly random magnitude, and are always toward the right (opposite the direction of desired locomotion). This type of noise makes planning difficult because it is not zero-mean, and perturbs an individual segment strongly at each time step (as opposed to an equivalent force being distributed evenly over the body). Figure 3(b) is a rendering of the performance of CE planning in this noisy domain. As compared to the deterministic walker, locomotion takes roughly 3.5 times as long to cover the same distance.

In the final, most difficult walking experiments, we demonstrate the ability of CE to plan in settings where the generative model used for planning is erroneous. In one of the experiments, the domain itself is deterministic, but the model incorrectly introduces the stochastic pushes. In the other experiment, the situation is reversed. Although we do not include figures due to space restrictions, locomotion is also performed successfully in these more difficult settings, albeit with a slower gait. In the deterministic walker, using a false stochastic generative model results in locomotion that takes 1.7 times as long as when the correct generative model is used. Compared to the stochastic walker, planning with an incorrect deterministic model produces locomotion taking 1.4 times as long as planning with a correct model.

Humanoid Stair Descent

The purpose of the final experimental domain is to demonstrate the ability of open-loop planning with CE to enable a humanoid to traverse uneven terrain. In particular, the final task requires the descent of a flight of stairs. Aside from the addition of stairs, the setting is identical to the deterministic walker. The solution to the task found by CE appears in Figure 4. As can be seen, the policy found is to run to the edge of the top step and then take a leap that clears the entire flight of stairs. The landing is performed successfully and running proceeded after the landing (not rendered). We anticipated a policy that would walk down the steps, but because the goal is to maximize the velocity of the hip along the horizontal axis, jumping across the the flight of stairs is superior to deliberately walking down it step by step, as long as the landing can be performed without falling.

[1] Due to space constraints, all details cannot be adequately presented here. Therefore the complete implementation is available: `http://code.google.com/p/cross-entropy/`



(a) The deterministic walker, renderings from every 10^{th} frame.

(b) The stochastic walker, renderings from every 30^{th} frame.

Figure 3: Stroboscopic rendering of the policies found by cross-entropy for humanoid locomotion.

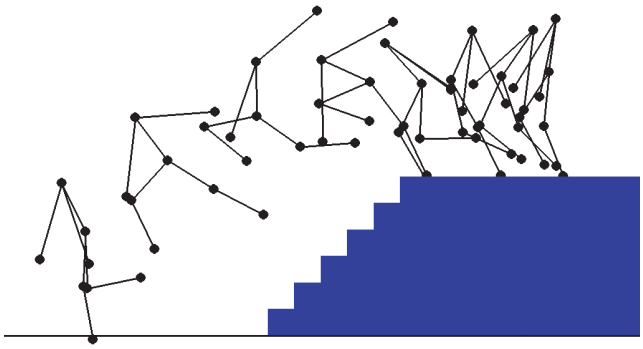


Figure 4: Cross-entropy finding a creative solution to the stair-descent problem. Renderings from every 10^{th} frame.

Discussion

In this paper, the ability of cross-entropy to both find extremely exact solutions in small domains, as well as effective solutions in very complex domains was demonstrated. The only domain expertise brought to bear was access to a black-box generative model of the domain.

Aside from its applicability to a large number of different settings, and its ability to plan even with flawed generative models, a significant advantage of CE for open-loop planning is the algorithmic simplicity of the method; it is extremely easy both to implement and verify correctness of the implementation. Computational and memory requirements are low as computational costs are linear in the number of samples drawn, and memory requirements are only linear in $n\rho$. Finally, within a generation, the method parallelizes trivially. This property is increasingly important, as the number of available cores on mainstream processors is increasing much faster than individual processor speed.

Although the computational costs of CE are linear in the number of samples, the actual running time of the algorithm depends heavily on the cost of the evaluation function. In domains that have simple dynamics, such as the double integrator, this is not a great concern. For domains that are expensive to evaluate, such as full physics simulations, actual running time can be extensive, or even prohibitive. Scaling the simulations into 3 dimensions, as opposed to the planar models here, would require even larger sample sizes and higher computational costs. Future research regarding improved responsiveness of the algorithm when coupled with expensive evaluation functions is warranted.

There are a number of methods that can be used to speed convergence to effective policies. As mentioned, CE lends itself well to parallelization. While parallelization was utilized here, ideally the number of processors would be equal to n; in these experiments while $n = 30$, the number of processors devoted to parallelization was 10. Increased parallelization for complex planning problems is promising, as it has been demonstrated that parallelization of CE yields the most significant speedups in domains that are computationally expensive (Evans, Keith, and Kroese 2007). Even with perfectly efficient parallelization, the amount of time required by the algorithm is still linear in the number of generations; it may be possible to mitigate this issue by increasing n, while decreasing ρ and G, so that the same number of trajectories are spread among fewer generations. The fully adaptive cross-entropy algorithm may also allow for enhanced speed, by adaptively determining a number of parameters of the algorithm, such as G and n.

An advantage of the approach presented is that while domain expertise is not required in order for the approach to be effective, additional knowledge can be incorporated in order to improve efficiency. One option is to use an adaptive approach for setting $\hat{\Phi}_0$. In the experiments here, $\hat{\Phi}_0$ is fixed to always ensure broad support over the sample space so that planning is likely to succeed, which is generally wasteful. A simple method to increase performance is to "warm start" $\hat{\Phi}_0$ based on $\hat{\Phi}_G$ from the previous phase of planning, or to use learning algorithms to estimate appropriate $\hat{\Phi}_0$ based on s and previous experience in the domain. Other methods that have been successful in similar settings is the creation of value function estimates to replace the final value in each rollout, and making simplifying assumptions about the domain (limiting search to limit cycles) (Tassa, Erez, and Smart 2008), and could be used as well.

References

Abbeel, P.; Coates, A.; Quigley, M.; and Ng, A. Y. 2007. An application of reinforcement learning to aerobatic helicopter flight. In *Proceedings of Neural Information Processing Systems 19*.

Boer, P.-T. D.; Kroese, D. P.; Mannor, S.; and Rubinstein, R. Y. 2005. A tutorial on the cross-entropy method. *Annals of Operations Research* 134.

Bubeck, S., and Munos, R. 2010. Open loop optimistic planning. In *Conference on Learning Theory*.

Bubeck, S.; Munos, R.; Stoltz, G.; and Szepesvári, C. 2008. Online optimization of χ-armed bandits. In *Proceedings of Advances in Neural Information Processing Systems 22*, volume 22, 201–208.

Costa, A.; Jones, O. D.; and Kroese, D. 2007. Convergence properties of the cross-entropy method for discrete optimization. *Operations Research Letters* 35(5):573 – 580.

Erez, T.; Tassa, Y.; and Todorov, E. 2011. Infinite-horizon model predictive control for periodic tasks with contacts. In *Proceedings of Robotics: Science and Systems*.

Erez, T. 2011. *Optimal Control for Autonomous Motor Behavior*. Ph.D. Dissertation, Washington University in Saint Louis.

Evans, G. E.; Keith, J. M.; and Kroese, D. P. 2007. Parallel cross-entropy optimization. In *2007 Winter Simulation Conference*, 2196–2202. IEEE.

Gordon, G. J. 1995. Stable function approximation in dynamic programming. In *The Twelfth International Conference on Machine Learning*, 261–268.

Kearns, M.; Mansour, S.; and Ng, A. 1999. A sparse sampling algorithm for near-optimal planning in large Markov decision processes. In *Proceedings of International Joint Conference on Artificial Intelligence*.

Kobilarov, M. 2011. Cross-entropy randomized motion planning. In *Proceedings of Robotics: Science and Systems*.

Kobilarov, M. 2012. Cross-entropy motion planning. *International Journal of Robotics*.

Ladd, A. M., and Kavraki, L. E. 2005. Motion planning in the presence of drift, underactuation and discrete system changes. In *Proceedings of In Robotics: Science and Systems I*, 233–241.

Manchester, I.; Mettin, U.; Iida, F.; and Tedrake, R. 2009. Stable dynamic walking over rough terrain : Theory and experiment. In *14th International Symposium on Robotics Research : Lucerne, Switzerland*, 1–16. Springer-Verlag.

Margolin, L. 2005. On the convergence of the cross-entropy method. *Annals of Operations Research* 134(1):201–214.

PyODE. 2010. version 2010-03-22.

Rubinstein, R. Y. 1997. Optimization of computer simulation models with rare events. *European Journal of Operations Research* 99:89–112.

Rubinstein, R. Y. 1999. The cross entropy method for combinatorial and continuous optimization. *Methodology And Computing In Applied Probability* 1:127–190.

Santamaría, J. C.; Sutton, R. S.; and Ram, A. 1996. Experiments with reinforcement learning in problems with continuous state and action spaces. *Adaptive Behavior*.

Stulp, F., and Sigaud, O. 2012. Path integral policy improvement with covariance matrix adaptation. In *Proceedings of the 29th International Conference on Machine Learning (ICML)*.

Szita, I., and Lőrincz, A. 2006. Learning tetris using the noisy cross-entropy method. *Neural Computation* 2936–2941.

Tassa, Y., and Todorov, E. 2010. Stochastic complementarity for local control of discontinuous dynamics. In *Proceedings of Robotics: Science and Systems*.

Tassa, Y.; Erez, T.; and Smart, W. 2008. Receding horizon differential dynamic programming. In Platt, J.; Koller, D.; Singer, Y.; and Roweis, S., eds., *Proceedings of Advances in Neural Information Processing Systems 20*, 1465–1472.

Tesauro, G., and Galperin, G. R. 1996. On-line policy improvement using monte-carlo search. In *NIPS*, 1068–1074.

Wampler, K., and Popović, Z. 2009. Optimal gait and form for animal locomotion. *SIGGRAPH*.

Weinstein, A., and Littman, M. L. 2012. Bandit-based planning and learning in continuous-action markov decision processes. In *Proceedings of the 22nd International Conference on Automated Planning and Scheduling*, 306–314.