Optimistic Planning for the Near-Optimal Control of General Nonlinear Systems with Continuous Transition Distributions ⋆

Lucian Buşoniu * **Levente Tamás** *

* *Department of Automation, Technical University of Cluj-Napoca, Romania (e-mail: {lucian.busoniu, levente.tamas}@ aut.utcluj.ro).*

Abstract: Optimistic planning is an optimal control approach from artificial intelligence, which can be applied in receding horizon. It works for very general nonlinear dynamics and cost functions, and its analysis establishes a tight relationship between computation invested and near-optimality. However, there is no optimistic planning algorithm that searches for closed-loop solutions in stochastic problems with continuous transition distributions. Such transitions are essential in control, where they arise e.g. due to continuous disturbances. Existing algorithms only search for open-loop input sequences, which are suboptimal. We therefore propose a closed-loop algorithm that discretizes the continuous transition distribution into sigma points, and call it sigma-optimistic planning. Assuming the error introduced by sigma-point discretization is bounded, we analyze the solution returned, showing that it is near-optimal. The algorithm is evaluated in simulation experiments, where it performs better than a state-of-the-art open-loop planning technique; a certainty-equivalence approach also works well.

Keywords: Optimal control, planning, nonlinear predictive control, artificial intelligence.

1. INTRODUCTION

Optimal control problems arise in many areas of technology. Here we consider the optimal control of general nonlinear systems with nonquadratic, discounted cost functions, in discrete time and for discrete inputs (control actions). We employ the *optimistic planning* (OP) class of algorithms originating in artificial intelligence (Munos, 2014; Buşoniu et al., 2012). These algorithms work by exploring, at each state encountered, a space of adaptive-horizon solutions (e.g. sequences of actions). Usually, only the first action of the best solution found is applied, and then the procedure is repeated in receding horizon. OP is based on insights from optimization, bandit theory, and classical planning/graph search, while from the control point of view it is classified as nonlinear model-predictive control (MPC) (Grüne and Pannek, 2011; Grimm et al., 2005; Camacho and Bordons, 2004, Ch. 9). OP is, however, quite different from traditional nonlinear MPC approaches, e.g. by fully analyzing the complete implementation, down to a solution with guaranteed near-optimality. The main features of OP are a tight relationship between the computation budget allocated to the algorithm and near-optimality, together with the generality of dynamics and cost functions: due to discounting, the only requirement is cost boundedness. The functions may e.g. be nonsmooth, in which case gradient-based MPC is not applicable.

OP algorithms have been introduced e.g. in (Hren and Munos, 2008; Bubeck and Munos, 2010; Buşoniu and Munos, 2012; Weinstein and Littman, 2012, 2013), and they have performed well in control problems (Weinstein and Littman, 2013). However, there does not yet exist an OP method that finds *closed-loop* solutions for stochastic systems with *continuous transition distributions*, and our aim in this paper is to develop such an algorithm. This type of transitions is essential in practical applications of control, arising e.g. due to continuous disturbances and noise. Some OP techniques can be applied to continuous transition distributions (Bubeck and Munos, 2010; Weinstein and Littman, 2012, 2013) but they only search for open-loop action sequences. Such solutions are suboptimal in the stochastic case, and closed-loop solutions are needed to react to the transition realizations. On the other hand, our previous closed-loop OP (Buşoniu and Munos, 2012) only works for stochastic transitions that can end up in a finite number of states.

The proposed method tackles continuous transition distributions by discretizing them into *sigma points* as in the unscented transform (Julier and Uhlmann, 1997; Ristic et al., 2004, Ch. 2). The method is therefore called sigma-OP. This simple idea allows us to directly apply closed-loop OP to the discretized version of the model; the solution it returns is then applied to the original system. Building on existing OP guarantees, we show that this solution is near-optimal, where the bound includes a new, constant term due to the error (assumed bounded) made by the unscented transform in approximating the relevant expectations. The method is evaluated in simulations on a linear system and a nonlinear one.

It should be noted that OP for stochastic systems is related to planning-based approaches for so-called partially

⋆ This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PNII-RU-TE-2012-3-0040, contract number 58/2013, and by SCIEX-NMS-CH project number 12.239.

observable problems (Ross et al., 2008; Silver and Veness, 2010; Smith and Simmons, 2004), where not all the states are measured – the AI approach to output feedback.

Next, Section 2 introduces OP for the discrete-distribution case. The new sigma-OP for continuous transition distributions is described in Section 3, analyzed in Section 4, and evaluated in simulation experiments in Section 5. Section 6 concludes with some ideas for future work.

2. BACKGROUND: OPTIMISTIC PLANNING FOR MARKOV DECISION PROCESSES

Consider a system with nonlinear stochastic dynamics:

$$x_{k+1} \sim f(x_k, u_k, \cdot) \tag{1}$$

where the transition function f provides for each pair of state x and action u the probability distribution $f(x, u, x')$ over next states x'. A reward function is defined, which measures the quality of state transitions: $r_{k+1} = \rho(x_k, u_k, x_{k+1})$.

We require discretized actions $u \in U$ with $M = |U|$ their number, and bounded rewards $r \in [0, 1]$ (note that arbitrary bounds can be accommodated by scaling and translation). We introduce the algorithm for the more restrictive case where each transition has a *finite* number N of outcomes with known probabilities. Such a problem is a Markov decision process (MDP), hence the algorithm is called OP-MDP. Note that action discretization reduces performance, but the loss is often manageable, as we illustrate later in the examples. Discrete actions may even be preferred due to e.g. bandwidth constraints on the communication between controller and actuator (De Persis and Frasca, 2013).

Before stating the control objective, it will be useful to introduce the solution concept of OP-MDP, called a *tree policy*. This solution will be local to the current system state, conventionally denoted x_0. It must first be explained that OP-MDP iteratively explores an infinite tree that represents all possible stochastic evolutions of the system starting from the current state. The root is this current state, and each node has at most NM children, corresponding to all possible states reachable by applying all possible actions. Formally, denote a state node by s, labeled by an actual state x. The infinite planning tree $\mathcal{T}_\infty$, of which Figure 1 only shows a few top nodes, is defined recursively as follows. First, the root node s_0 is labeled by the current state x_0, and then each node s is expanded by adding, for any state x' for which $f(x, u, x') > 0$ for some u, a new child node s' labeled by x'. Note that Figure 1 explicitly includes also the action nodes in circles.

A tree policy h_∞ is then an assignment of actions to a subtree $\mathcal{T}_{h_\infty}$ of $\mathcal{T}_\infty$, $h_\infty : \mathcal{T}_{h_\infty} \to U$, recursively taking into account only the nodes reached under the action choices made so far:

$$\mathcal{T}_{h_\infty} = \{s' \in \mathcal{T}_\infty | s' = s_0 \text{ or } \exists s \in \mathcal{T}_{h_\infty}, s' \in \mathcal{C}(s, h_\infty(s))\}$$

where the actions are assigned as desired; $\mathcal{C}(s, u)$ denotes the child nodes of s along action u. The branching factor of $\mathcal{T}_{h_\infty}$ is (at most) N.

The optimal control objective is to find at state x_0 a policy h_∞ maximizing the expected return:

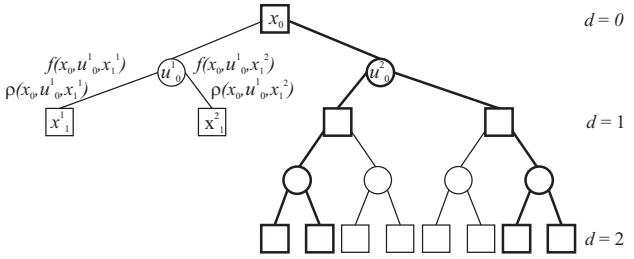

Fig. 1. Illustration of OP-MDP tree for $N = M = 2$. The squares are state nodes labeled by states x, and the actions u are explicitly included as circle nodes. Transition arcs to next states are labeled by probabilities f and rewards ρ. Superscripts index the possible actions and state outcomes, while subscripts are depths, which increase only with the state node levels. The thick subtree highlights a tree policy.

$$V^{h_\infty}(x_0) = \mathrm{E}\left\{\sum_{k=0}^{\infty} \gamma^k r_{k+1}\right\} \tag{2}$$

where the expectation is taken over all future trajectories possible under h_∞ (all paths in $\mathcal{T}_{h_\infty}$). Finally, denote the optimal value $v^* = V^*(x_0) := \sup_{h_\infty} V^{h_\infty}(x_0)$. This formulation is nonstandard, since typically state feedback control policies $\pi(x)$ are sufficient to achieve the optimal values. Nevertheless, the tree policy formulation helps to understand the approach, and does not lose generality.

Of course, OP-MDP can only work with finite trees and policies; such finite policies are denoted simply by h, and one of them is exemplified in Figure 1. They must be well-defined, connected subtrees $\mathcal{T}_h$ at the top of $\mathcal{T}_\infty$, so that any node is either fully expanded (with all nonzero probability children) or not at all. The leaves of $\mathcal{T}_h$ are denoted by $\mathcal{L}_h$. We will treat policies h and their corresponding trees $\mathcal{T}_h$ interchangeably. Define three values:

$$\begin{aligned} \ell(h) &= \sum_{s \in \mathcal{L}_h} P(s)\, R(s) \\ b(h) &= \sum_{s \in \mathcal{L}_h} P(s)\, [R(s) + \frac{\gamma^{d(s)}}{1-\gamma}] \\ v(h) &= \sum_{s \in \mathcal{L}_h} P(s)\, [R(s) + \gamma^{d(s)} V^*(x(s))] \end{aligned} \tag{3}$$

where $R(s)$ is the discounted return accumulated along the path from the root to s, $P(s)$ is the probability of reaching leaf s (the product of the individual transition probabilities along the path), and $d(s)$ gives the depth of s. So, $\ell(h)$ is the expected partial return accumulated by h and is a lower bound for the expected return of any complete, infinite policy h_∞ starting with h; $b(h)$ is an upper bound on these expected returns; and $v(h)$ is the expected return when continuing optimally below h. It is important to note that $b(h) = \ell(h) + \sum_{s \in \mathcal{L}_h} P(s) \frac{\gamma^{d(s)}}{1-\gamma}$. We denote the summation in this expression by $\delta(h)$, the diameter [1] of h; and $c(s) = P(s) \frac{\gamma^{d(s)}}{1-\gamma}$, the contribution of node s to the diameter. The diameter can be intuitively seen as the uncertainty on the value of h, while $c(s)$ is the contribution of the leaf to this uncertainty.

[1] This is, indeed, a true diameter in an appropriately defined metric over the space of policies.

Using this development, OP-MDP is easy to understand: it builds a subtree of $\mathcal{T}_\infty$ by refining at each iteration an optimistic policy $h^\dagger$ that maximizes b (intuitively seen as the most promising policy). A policy has multiple leaf nodes; the algorithm expands one that has maximal contribution to the diameter. The algorithm stops after n node expansions, and at the end returns a policy h^* maximizing ℓ (a safe choice). The first action u_0 of this policy is applied to the system, and the algorithm is re-applied from the new state. OP-MDP is related to classical AO* search and described in more detail in (Buşoniu and Munos, 2012).

The algorithm is guaranteed to find a near-optimal action u_0, in the sense:

$$v^* - v(u_0) \leq \delta^* \tag{4}$$

where δ^* is the smallest diameter of any policy that was selected for expansion, and v is from (3). This difference measures the loss in performance due to the first action applied, and is meaningful for an algorithm such as OP-MDP, which separately computes actions at each encountered state. For a given n, diameter δ^* is smaller when the subtree of $\mathcal{T}_\infty$ that OP-MDP must expand has fewer nodes, and Buşoniu and Munos (2012) characterize this subtree's size. Briefly, in the worst case the full tree with branching factor NM must be expanded, which means $n = O(NM^d)$ expansions are required to reach depth d, and thus to obtain a diameter and near-optimality of $\frac{\gamma^d}{1-\gamma}$. In typical problems, the branching factor is smaller, and δ^* decreases faster.

In practice, upper and lower bounds are stored for nodes rather than tree policies, in the form of B and L-values. B-values are backed up along the tree:

$$B(s) = \max_u \underbrace{\sum_{s' \in \mathcal{C}(s,u)} f(x,u,x')[\rho(x,u,x') + \gamma B(s')]}_{=:T(B;s,u)} \tag{5}$$

starting from $\frac{1}{1-\gamma}$ at the leaves, where $x = x(s)$ and $x' = x(s')$. The operator $T(B;s,u)$, parameterized by function B, has been introduced. The L-values are similarly found, by applying (5) but this time using $T(L;s,u)$ and starting from 0 at the leaves. Optimistic policies (subtrees) $h^\dagger$ are constructed starting downwards from the root and always following actions that lead to maximal values of $T(B;s,u)$ in (5), while near-optimal policies h^* select actions with maximal $T(L;s,u)$.

3. SIGMA-OP FOR CONTINUOUS DISTRIBUTIONS

Our goal in this paper is to extend OP-MDP to continuous distributions over next states x'. Formally, denote the probability density function of x' after taking u in x by $\tilde{f}(x,u,x')$. The reward function is unchanged, $\rho(x,u,x')$. The extension of tree policies h_∞ is complicated, so consider instead state feedbacks $\pi : X \to U$, which as stated above can represent optimal solutions. The value function is $W^\pi(x_0) = \mathrm{E}\left\{\sum_{k=0}^\infty \gamma^k r_{k+1}\right\}$ where actions are taken with π, and the goal is to achieve the optimum $W^*(x_0) = \sup_\pi W^\pi(x_0)$. The ideal backups would then involve expectations over continuous variables:

$$B(x) = \max_u \underbrace{\mathrm{E}_{x' \sim \tilde{f}(x,u,\cdot)}\{\rho(x,u,x') + \gamma B(x')\}}_{=:\tilde{T}(B;x,u)} \tag{6}$$

where we slightly abuse the notation by directly using states x instead of state nodes s. At the end of this section we will explain how a well-defined tree, with nodes s, is obtained. Note that operator $\tilde{T}$ is a generalization of T.

Our main idea is simple: exploit the unscented transform to approximate these expectations, by discretizing the random variable x' into a set of *sigma points* (Ristic et al., 2004, Ch. 2). Denote by $\mu(x,u)$ and $\sigma(x,u)$ the mean and covariance of x'; note they can change with the origin (x,u) of the state transition. The means and covariances must be known (true for many forms of $\tilde{f}$), or otherwise their estimation must be computationally cheap. The sigma points are then:

$$\mathcal{X}_i(x,u) = \mu(x,u) + \begin{cases} 0, & i = 0 \\ [\sqrt{(m+\kappa)\sigma(x,u)}]_i, & i = 1,\ldots,m \\ -[\sqrt{(m+\kappa)\sigma(x,u)}]_i, & i = m+1,\ldots,2m \end{cases}$$

and their weights:

$$\alpha_0(x,u) = \frac{\kappa}{m+\kappa}, \alpha_i(x,u) = \frac{1}{2(m+\kappa)}, i = 1,\ldots,2m$$

where κ is a tuning parameter and $[\cdot]_i$ denotes the ith row of the argument matrix.

Then, the expected value in (6) is approximated by a weighted summation:

$$\begin{aligned} \tilde{T}(B;x,u) &\approx \sum_{i=0}^{2m} \alpha_i(x,u)[\rho(x,u,\mathcal{X}_i(x,u)) + \gamma B(\mathcal{X}_i(x,u))] \\ &= T(B;x,u) \end{aligned}$$

which is accurate up to the second order of the Taylor expansion of the function inside the expectation (Ristic et al., 2004). By interpreting the weights α_i as probabilities (which is possible as long as κ is chosen positive), the summation is entirely similar to that in (5), which is why we directly reused the operator T. The approximate algorithm expands a node by only creating children for the sigma points ($N = 2m+1$ children for each state-action pair), and then applies the discretized operator T to backup B and L values.

To make this formally complete, define a discretized, approximate version of the original continuous-distribution problem, by taking $f(x,u,x') = \alpha_i(x,u)$ when $x' = \mathcal{X}_i(x,u)$, and 0 otherwise. The reward function ρ is kept unchanged. OP-MDP is then simply applied to this discretized problem, to find an approximately optimal action u_0 which is applied to the real, original system. We call the overall approach sigma-OP (short for "OP with sigma-point discretization").

4. ANALYSIS AND DISCUSSION

Since the discretized problem is a valid MDP, the guarantees of OP-MDP directly hold *for this discretized problem.* The near-optimality bound (4) is essential: $v^* - v(u_0) \leq \delta^*$, where all the notations of Section 2 are kept for the discretized problem. The goal is, however, to bound the suboptimality in the *original* problem. We therefore analyze

the corresponding quantity $w^* - w(u_0)$ under the original value function, i.e. with $w^* := W^*(x_0)$ and $w(u_0) := \tilde{T}(W^*; x_0, u_0)$. Note that, similarly, $v(u_0) = T(V^*; x_0; u_0)$ by definition (3).

Define an approximate value iteration algorithm:

$$V_{t+1}(x) = \max_u T(V_t; x, u) \tag{7}$$

with V_0 arbitrarily initialized. These updates are closely related to sigma-OP: when $V_0 = \frac{1}{1-\gamma}$ (or 0) they correspond to B-value (or L-value) updates on the infinite tree $\mathcal{T}_\infty$.

Assumption 1. There exists an $\varepsilon \geq 0$ and a bounded initial value function V_0 so that, for any V_t and any x, u, sigma-point approximation makes an error of at most ε: $|\tilde{T}(V_t; x, u) - T(V_t; x, u)| \leq \varepsilon$, and the same is true for W^*: $|\tilde{T}(W^*; x, u) - T(W^*; x, u)| \leq \varepsilon$.

Since the unscented transform guarantees accuracy up to the second order, this assumption essentially requires the terms under the expectations (rewards and value functions) to be well-behaved. For example, if the rewards (and therefore the value functions) are quadratic in x', then $\varepsilon = 0$. Under Assumption 1, the following main result holds.

Theorem 1. The action returned by sigma-OP is near-optimal: $w^* - w(u_0) \leq \delta^* + 2\frac{\varepsilon}{1-\gamma}$.

Proof. The assumption implies that for any t and x:

$$|V_{t+1}(x) - \max_u \tilde{T}(V_t; x, u)| \leq \varepsilon$$

where the second term is the update *exact* value iteration would apply to V_t. In addition, since (7) is just value iteration on the discretized MDP, it converges to V^*. Given these conditions and the boundedness of W^* (resulting from the boundedness of the rewards), standard results in approximate dynamic programming (Bertsekas and Tsitsiklis, 1996, Sec. 6.5.3) guarantee that: [2]

$$|W^*(x) - V^*(x)| \leq \frac{\varepsilon}{1-\gamma}, \quad \forall x \tag{8}$$

Now:

$$\begin{aligned} v(u_0) &= T(V^*; x_0, u_0) \\ &= T(W^*; x_0, u_0) + \gamma \sum_{i=0}^{2m} \alpha_i [V^*(\mathcal{X}_i) - W^*(\mathcal{X}_i)] \\ &= w(u_0) + T(W^*; x_0, u_0) - \tilde{T}(W^*; x_0, u_0) \\ &\quad + \gamma \sum_{i=0}^{2m} \alpha_i [V^*(\mathcal{X}_i) - W^*(\mathcal{X}_i)] \end{aligned}$$

by the definition of $v(u_0)$ and $w(u_0)$, where for readability we skipped the argument (x, u) of α_i and $\mathcal{X}_i$. The first difference in the last formula is bounded in magnitude by ε by Assumption 1, and the second by $\gamma\frac{\varepsilon}{1-\gamma}$ due to (8). Therefore, $|v(u_0) - w(u_0)| \leq \varepsilon + \gamma\frac{\varepsilon}{1-\gamma} = \frac{\varepsilon}{1-\gamma}$.

Combining this with $|W^*(x_0) - V^*(x_0)| \leq \frac{\varepsilon}{1-\gamma}$ from (8), and with (4), $v^* - v(u_0) \leq \delta^*$, the final result follows. □

The theorem says that sigma-OP will make, in addition to the error δ^* made by OP in the discretized problem, (up to) a constant error $\frac{2\varepsilon}{1-\gamma}$ due to the sigma point approximation. Connecting this to the analysis of OP-MDP, in the worst case where the full tree must be expanded in the order of depth, we have $\delta^* = \frac{\gamma^d}{1-\gamma}$ where $n = \frac{NM^d - 1}{NM - 1}$ expansions are required to reach depth d, so that $w^* - w(u_0) \leq (\gamma^{\frac{\log n}{\log NM}} + 2\varepsilon)\frac{1}{1-\gamma}$. In typical problems, only a smaller subtree must be expanded, so the δ^* part of the bound decreases faster with n.

Note that the OP-MDP analysis closely characterizes the size of this subtree and the dependence of δ^* on n, but only asymptotically, for δ^* around 0 ($n \to \infty$). This is not appropriate in sigma-OP, because the constant error $2\frac{\varepsilon}{1-\gamma}$ dominates asymptotically; instead, in future work it will be useful to investigate the behavior of the tree for δ^* on the order of nonzero constant $\frac{\varepsilon}{1-\gamma}$.

In any case, the analysis indicates that computation should not be invested to shrink the diameter beyond a constant value, since as n grows the performance will plateau around this constant anyway. This constant is on the order of the error made by sigma-point approximation.

5. SIMULATION RESULTS

In our experiments we compare sigma-OP with three alternative planning algorithms. The first uses the same tree structure as sigma-OP, but expands nodes uniformly, in the order of depth. It is still a correct algorithm (it attains a near-optimal solution given enough computation), but it shrinks diameters slower: it always behaves as sigma-OP would in the worst case. Uniform planning is used as a baseline to confirm that optimistic expansion makes sense despite approximation errors.

The second alternative is OP for deterministic systems (OPD) (Hren and Munos, 2008), which we applied to the nominal, deterministic model, while the actions returned were heuristically executed in the true stochastic system. OPD can be viewed as "certainty-equivalence" planning. Finally, we choose a state-of-the-art method called HOLOP (Weinstein and Littman, 2012). HOLOP has different characteristics than sigma-OP: it seeks solutions represented as action sequences, which in contrast to closed-loop policies, cannot represent optimal controls in the stochastic case. HOLOP works for continuous actions, while the other three methods require discretized actions. So in effect HOLOP and sigma-OP are solving different optimal control problems, but they can both be applied as approximations to continuous-action continuous-distribution stochastic systems, as we will do below.

5.1 DC motor stabilization

The first problem concerns a DC motor with state variables: shaft angle $x_1 \in [-\pi, \pi]$ rad, angular velocity $x_2 \in [-16\pi, 16\pi]$ rad/s, and action variable: voltage $u \in [-10, 10]$ V. The dynamics are linear:

$$x_{k+1} = Ax_k + Bu_k + z_k, \quad A \approx \begin{bmatrix} 1 & 0.0095 \\ 0 & 0.9100 \end{bmatrix}, \; B \approx \begin{bmatrix} 0.0084 \\ 1.6618 \end{bmatrix}$$

where z is zero-mean Gaussian distributed with covariance $\sigma = 0.1 \cdot \mathrm{diag}(1, 1)$, leading to a stochastic component of considerable amplitude. The two states are kept

[2] In fact, sigma-point discretization leads to a classical type of value function approximator called an averager.

within their bounds by saturation, also when discretizing into sigma points, and the actions are discretized in $\{-10, 0, 10\}$ V, so $M = 3$. The goal is stabilization at zero, and is described by the reward function:

$$r_{k+1} = -x_k^T Q x_k - u_k^T R u_k,\ Q = \mathrm{diag}(1, 0),\ R = 0.001 \quad (9)$$

with discount factor $\gamma = 0.95$. Using the known variable bounds, the reward is scaled and translated into $[0, 1]$. The DC motor is chosen because its nominal dynamics are simple, so we can focus on the effects of noise.

The four planning algorithms were applied in receding horizon, from the initial state $[-\pi, 0]^T$ and for a duration of 1 s (100 steps, due to a sampling time of 0.01 s). For each algorithm and parameter setting, 25 independent runs were performed. In sigma-OP and uniform planning, the number of sigma points was $N = 5$ ($m = 2$), and κ was set to 0.001. The algorithms were tested for a range of computational budgets expressed as numbers of transitions simulated during planning at each step: $n_\mathrm{t} = 150, 300, 600, 1200, 2400, 4800$. Since for sigma-OP and uniform planning budgets n are given in terms of node expansions, and each expansion takes $NM = 15$ transitions, n is taken $\lceil n'/15 \rceil$ where $\lceil \cdot \rceil$ denotes ceiling. HOLOP is additionally parameterized by the horizon K over which it searches for action sequences, and for each n_t we tried values $5, 10, 25, 75, 100$ for K; the experiment with the best upper confidence bound on the return is reported (in this problem, $K = 5$ was always the best). Since the problem is linear and quadratic, by disregarding the state and action constraints a continuous-action optimal solution is analytically computed as in (Bertsekas, 2007, Sec. 3.2), and its optimal value is included on the graphs.

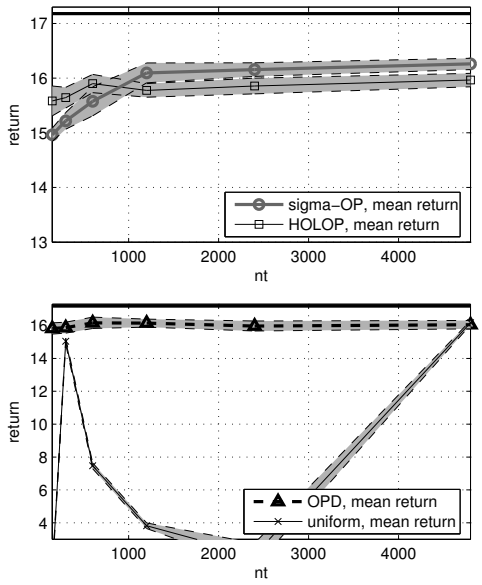

Fig. 2. Return for the DC motor. Mean performances are shown, with their 95% confidence intervals as a shaded region. The horizontal solid line shows the analytically computed optimal value. For readability, results are shown in two graphs with different vertical scale.

The results are shown in Figure 2, where performance is measured by the discounted return obtained in the experiment. For larger budgets sigma-OP is better than HOLOP. Certainty-equivalence OPD on the other hand performs as well as sigma-OP. Note however that using it means the analytical guarantees of Theorem 1 are sacrificed, since OPD is only a heuristic in the stochastic problem. Uniform planning has unreliable performance, sometimes reaching good values but then becoming worse again for larger budgets. This indicates a good algorithm must search optimistically. Since it is a correct algorithm, uniform planning would eventually stabilize to a good performance for very large budgets. Because actions are discretized, no algorithm reaches the continuous-action optimal value.

5.2 Inverted pendulum swing-up

The second problem involves swinging up and stabilizing an underactuated inverted pendulum. Due to limited power, from certain states (e.g., pointing down) the pendulum needs to be swung back and forth to gather energy, prior to being pushed up and stabilized. While being a standard control benchmark, the swing-up problem supplies an interesting challenge to planning algorithms: the swing-ups must be planned over a longer horizon, and solutions that seem good over a short horizon will not work. The first state $x_1 = \alpha$ is the angle and wraps around in the interval $[-\pi, \pi)$ rad; the second state is the angular velocity $x_2 = \dot{\alpha} \in [-15\pi, 15\pi]$ rad/s. The action $u \in [-3, 3]$ V is the motor voltage, and was discretized into $\{-3, 0, 3\}$ V. A quadratic reward function of the form (9) was used, with $Q = \mathrm{diag}(1, 0)$ and $R = 0.3$, and was normalized into $[0, 1]$. The noise is again zero-mean Gaussian, now with covariance $\sigma = 0.015 \cdot \mathrm{diag}(1, 1)$.

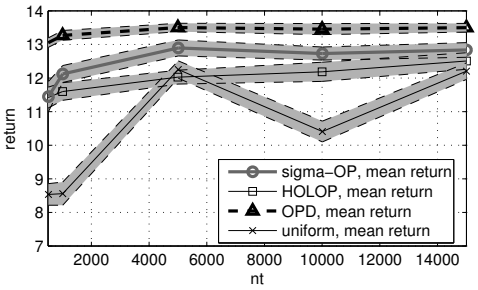

Fig. 3. Return for the inverted pendulum.

The simulation time was chosen to be 5 s with a sampling time of 0.05 s (100 steps), and the initial state was $x_0 = [-\pi, 0]^T$ (pointing down). The same values as for the DC motor were set for the sigma points parameters, while the budgets were $n_\mathrm{t} = 500, 1000, 5000, 10000, 15000$ (and $n = \lceil n'/15 \rceil$). Figure 3 reports the results of a batch of 35 experiments. For the K parameter in HOLOP, values $5, 10, 15, 20, 25, 30, 40, 50, 75, 100$ were attempted, and the best results corresponding to the values of n_t above are obtained for, respectively, $K = 10, 5, 10, 15, 10$.

Sigma-OP remains overall better than HOLOP, although for $n_\mathrm{t} = 500$ and 15000 their performances are not statistically different. OPD this time works better than all other algorithms, confirming that it may be a good choice in practice despite the lack of guarantees. Uniform planning is still unreliable.

Figure 4 shows a controlled trajectory with sigma-OP, for $n = 667$. Chattering is observed due to controlling to

an unstable equilibrium using only discrete actions, and also because sometimes the random transitions move the system away from the equilibrium and it must be brought back. At certain points even a new swing may be necessary.

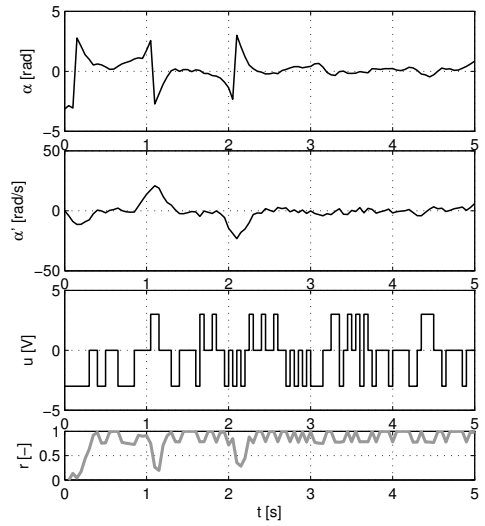

Fig. 4. A controlled trajectory.

Regarding computation, it is dominated by the number of simulations, which is set the same for all algorithms. However, small differences arise due to their different internal workings. In our Matlab implementation, HOLOP is more expensive, while the other algorithms are cheaper and their relationships change with the problem.

6. CONCLUSION

An optimistic planning method was introduced to solve general nonlinear, stochastic optimal control problems with discrete actions and continuous transition distributions. The method was analyzed and shown to return near-optimal solutions. It successfully solved a linear problem and a nonlinear one. A heuristic certainty-equivalence approach that plans using the deterministic model also performed very well empirically.

Sigma points are just one way to discretize the transition distribution, and other methods could be applied. E.g. one limitation of sigma points is that most of them have equal probabilities, which can lead to a large branching factor in the OP tree (Buşoniu and Munos, 2012). An alternative is to use a particle-filter discretization, for which the probabilities are less symmetrical, similarly to Silver and Veness (2010). The branching factor may then be too large due to the number of particles, and different tree expansion strategies may be necessary. It would also be interesting to see whether the certainty-equivalence approach remains competitive for multimodal distributions.

REFERENCES

Bertsekas, D.P. (2007). *Dynamic Programming and Optimal Control, Vol. II, 2nd Ed.* Athena Scientific, Belmont, MA.

Bertsekas, D.P. and Tsitsiklis, J.N. (1996). *Neuro-Dynamic Programming.* Athena Scientific.

Bubeck, S. and Munos, R. (2010). Open loop optimistic planning. In *Proceedings 23rd Annual Conference on Learning Theory (COLT-10)*, 477–489. Haifa, Israel.

Buşoniu, L. and Munos, R. (2012). Optimistic planning for Markov decision processes. In *Proceedings 15th International Conference on Artificial Intelligence and Statistics (AISTATS-12)*, volume 22 of *JMLR Workshop and Conference Proceedings*, 182–189. La Palma, Canary Islands, Spain.

Buşoniu, L., Munos, R., and Babuška, R. (2012). A review of optimistic planning in Markov decision processes. In F. Lewis and D. Liu (eds.), *Reinforcement Learning and Adaptive Dynamic Programming for Feedback Control*. Wiley.

Camacho, E.F. and Bordons, C. (2004). *Model Predictive Control*. Springer-Verlag.

De Persis, C. and Frasca, P. (2013). Robust self-triggered coordination with ternary controllers. *IEEE Transactions on Automatic Control*, 58(12), 3024–3038.

Grimm, G., Messina, M., Tuna, S., and Teel, A. (2005). Model predictive control: For want of a local control lyapunov function, all is not lost. *IEEE Transactions on Automatic Control*, 50(5), 546–558.

Grüne, L. and Pannek, J. (2011). *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer.

Hren, J.F. and Munos, R. (2008). Optimistic planning of deterministic systems. In S. Girgin, M. Loth, R. Munos, P. Preux, and D. Ryabko (eds.), *Recent Advances in Reinforcement Learning*, volume 5323 of *Lecture Notes in Computer Science*, 151–164. Springer.

Julier, S. and Uhlmann, J. (1997). A new extension of the kalman filter to nonlinear systems. In *Proceedings 11th International Symposium on Aerospace/Defense Sensing, Simulations and Controls (AeroSense-97)*.

Munos, R. (2014). The optimistic principle applied to games, optimization and planning: Towards foundations of Monte-Carlo tree search. *Foundations and Trends in Machine Learning*, 7(1), 1–130.

Ristic, B., Arulampalam, S., and Gordon, N. (2004). *Beyond the Kalman Filter: Particle Filters for Tracking Applications*. Artech House.

Ross, S., Pineau, J., Paquet, S., and Chaib-draa, B. (2008). Online planning algorithms for POMDPs. *Journal of Artificial Intelligence Research (JAIR)*, 32, 663–704.

Silver, D. and Veness, J. (2010). Monte-Carlo planning in large POMDPs. In J.D. Lafferty, C.K.I. Williams, J. Shawe-Taylor, R.S. Zemel, and A. Culotta (eds.), *Advances in Neural Information Processing Systems 23*, 2164–2172. MIT Press.

Smith, T. and Simmons, R.G. (2004). Heuristic search value iteration for POMDPs. In *Proceedings 20th Conference in Uncertainty in Artificial Intelligence*, 520–527. Banff, Canada.

Weinstein, A. and Littman, M.L. (2012). Bandit-based planning and learning in continuous-action Markov decision processes. In *Proceedings 22nd International Conference on Automated Planning and Scheduling (ICAPS-12)*. São Paulo, Brazil.

Weinstein, A. and Littman, M.L. (2013). Open-loop planning in large-scale stochastic domains. In *Proceedings 27th AAAI Conference on Artificial Intelligence (AAAI-13)*, 1436–1442. Bellevue, Washington, US.