Discounted near-optimal control of general continuous-action nonlinear systems using optimistic planning

Lucian Buşoniu, Előd Páll, Rémi Munos

***Abstract*— We propose an optimistic planning method to search for near-optimal sequences of actions in discrete-time, infinite-horizon optimal control problems with discounted rewards. The dynamics are general nonlinear, while the action (input) is scalar and compact. The method works by iteratively splitting the infinite-dimensional search space into hyperboxes. Under appropriate conditions on the dynamics and rewards, we analyze the shrinking rate of the range of possible values in each box. When coupled with a measure of problem complexity, this leads to an overall convergence rate of the algorithm to the infinite-horizon optimum, as a function of computation invested. We provide simulation results showing that the algorithm is useful in practice, and comparing it with two alternative planning methods.**

I. Introduction

We consider optimal control problems that require maximizing a discounted sum of rewards (the value), along an infinitely long discrete-time trajectory of the system [9], [16]. Such problems are encountered not only in automatic control, but also in many other fields including artificial intelligence (AI), operations research, economics, etc. When the system and reward function have a general form, numerical algorithms must be applied to solve the problem approximately.

We focus on algorithms that solve the problem locally, for the current state of the system, obtaining a sequence of actions (inputs). The initial action is applied, and the procedure is repeated online for subsequent states. This is called receding horizon model predictive control [6], or online planning in AI [11]. The direct dependence on the state space size is removed, while computation still grows with the action space size and with the search horizon. Nonlinear predictive control, for example, often searches over a fixed finite horizon. The near-optimality of such solutions may be unclear when measured via the true, infinite-horizon value. A good approach is to search directly over the space of infinitely long solutions. This is the approach taken by the optimistic planning (OP) class of algorithms [15], which we consider here.

OP methods originate in AI and perform a branch-and-bound search over the space of infinitely long sequences, always refining the region with the best upper bound on the value – hence the "optimistic" label. The main strengths of OP are the generality of the dynamics and rewards addressed, and a tight relation between computation and near-optimality, which exploits ideas from reinforcement learning [15]. Many OP variants have been proposed for discrete actions, e.g. [4], [8], [10], [13], [17]. In contrast, we focus here on continuous actions, since they are essential in control.

We propose *optimistic planning with continuous actions* (OPC), which works in general nonlinear systems with scalar bounded actions. The method iteratively splits the infinite-dimensional hypercube of continuous-action sequences into smaller hyperboxes, leading to an adaptive search horizon. Under Lipschitz continuity of the dynamics and rewards and a stability-like condition, we derive an upper bound on the range of values inside a box (a diameter), and thereby an optimistic selection rule for the box to split next. An essential insight is that each dimension k contributes to the bound with weight γ^k, where γ is the discount factor, and this is used to select the specific dimension to split. We characterize the rate at which the diameter shrinks with the number of splits, and define (as a measure of problem complexity) the branching factor of an associated tree [5], [8]. Using these, we derive an overall near-optimality guarantee for the algorithm as a function of computation invested, measured by the number of transitions simulated. Empirical results in a linear and a nonlinear system validate the method in practice.

Several other OP methods have been proposed for continuous actions, but without an analysis; OPC is the first to guarantee a convergence rate. Lipschitz planning (LP) [7] uses a similar upper bound but lacks the insight on the impact γ^k, so it uses a heuristic rule to choose which dimension to split. Our earlier method called simultaneous optimistic optimization for planning (SOOP) [3] exploits an inkling of this insight, imposing instead of γ^k a tunable empirical weight α^k. The box selection rule in SOOP is also heuristic.

Other continuous-action planners only optimize over finite horizons, e.g. HOOT [12], sequential planning [7], or HOLOP [18]. OPC applies the *principle* of deterministic optimistic optimization (DOO) [14] to control, while the *analysis* of DOO does not work because its assumptions are not satisfied for infinite-horizon continuous-input problems, so we provide novel analysis adapted to this setting.

Next, Section II formalizes the problem and Section III describes OPC. Sections IV and V provide analysis and simulation results. Section VI concludes the paper.

II. Problem statement

We consider an optimal control problem for a discrete-time nonlinear system:

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

L. Buşoniu and E. Páll are with the Automation Department, Technical University of Cluj-Napoca, Romania (lucian@busoniu.net, pall.elod@gmail.com). R. Munos is with Google DeepMind London, UK (munos@google.com). 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.

where $x \in X \subseteq \mathbb{R}^p$, $u \in U$, and U will be described in our main assumption below. A function $\rho : X \times U \to \mathbb{R}$ assigns a numerical reward $r_k = \rho(x_k, u_k)$ to each state-action pair. Under a fixed initial state x_0, define an infinitely-long sequence of actions $\boldsymbol{u}_\infty = (u_0, u_1, \dots)$ and the infinite-horizon discounted value of this sequence:

$$v(\boldsymbol{u}_\infty) = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, u_k) \tag{2}$$

where $\gamma \in (0, 1)$ is the discount factor, and $x_{k+1} = f(x_k, u_k)$. Discounting is used in many other works, e.g. [1], [9]. The objective is to find (a near-optimal approximation of) the optimal value:

$$\sup_{\boldsymbol{u}_\infty} v(\boldsymbol{u}_\infty) =: v^*$$

and an action sequence that achieves this (near-optimal) value.

So far the problem is extremely general: no specific form is required for either the dynamics f or the reward function ρ. Next, we impose some assumptions that allow us to derive an efficient algorithm.

Assumption 1: The following conditions hold.

(i) The rewards are bounded in $[0, 1]$.

(ii) The action is a real scalar, bounded in the unit interval, so that $U = [0, 1]$.

(iii) The dynamics and rewards are Lipschitz, i.e. $\exists L_f, L_\rho$ so that $\forall x, x' \in X, u, u' \in U$:

$$\|f(x, u) - f(x', u')\| \le L_f(\|x - x'\| + |u - u'|)$$
$$|\rho(x, u) - \rho(x', u')| \le L_\rho(\|x - x'\| + |u - u'|)$$

where $\|\cdot\|$ is an appropriately chosen norm.

(iv) The dynamics satisfy $\gamma L_f < 1$.

Reward boundedness as in Assumption 1(i) can be achieved e.g. by saturating a possibly unbounded original reward function. This changes the optimal solution, but is often sufficient in practice. Another example is when physical limitations in the system are modeled by saturating the states and actions, from which a reward bound follows.

In combination with Assumption 1(i), discounting will ensure that returns are bounded for *any* sequence – a property required by the analysis. Under appropriate stability conditions, boundedness may also be guaranteed without discounting, but only for (near-)optimal sequences; tightening the analysis so that it still holds in this case is an interesting topic for future work.

The scalar action from Assumption 1(ii) could be generalized to multiple dimensions, e.g. by always splitting along all of these dimensions in the algorithm of Section III. This would however introduce overhead in the analysis that would not be useful in grasping its main features, so here we choose to restrict to the scalar case. On the other hand, the compact nature of U is fundamental, since our algorithm numerically refines this action space. In both Assumptions 1(i) and 1(ii), the unit interval is taken only for convenience, and can be achieved by simply rescaling any bounded interval.

Assumption 1(iii) is a regularity condition, while Assumption 1(iv) can be interpreted as a stability requirement: the dynamics need not be strictly contractive on their own, but must become so when combined with a shrink rate given by the discount factor γ.

III. Optimistic planning for continuous actions

We apply the principles of DOO [14] to maximize the objective function v over the space of infinitely long sequences U^∞. The main idea is to iteratively split the search space into smaller subsets, where at each iteration the set to split further is selected optimistically, as the one with the largest upper bound on the values of sequences within it.

To derive the splitting procedure, U^∞ can be visualized as an infinite dimensional hypercube, with each dimension k the action space at step k. This hypercube is iteratively split into smaller hyperboxes (boxes, for short), like in [3], [7], each of which gets a unique index i. Such a box $\mathcal{U}_i \subseteq U^\infty$ is the cross-product of a sequence of intervals $(\mu_0^i, \dots, \mu_{K_i-1}^i, U, U, \dots)$ where $\mu_k^i \subseteq U$ and $K_i - 1$ is the largest discretized dimension; for all further dimensions $\mu_k^i = U$. A box is further explored by splitting into M pieces the interval of some dimension k, which corresponds to discretizing the action at step k. In Figure 1, left an example exploration of U^∞ is shown. Define d_k^i to be the length of the interval μ_k^i in box $\mathcal{U}_i$, and u_k^i a sample action taken somewhere in this interval (e.g., at the center). For each box, the sequence of rewards r_k^i obtained by applying u_k^i from x_0 is computed by simulating the system.

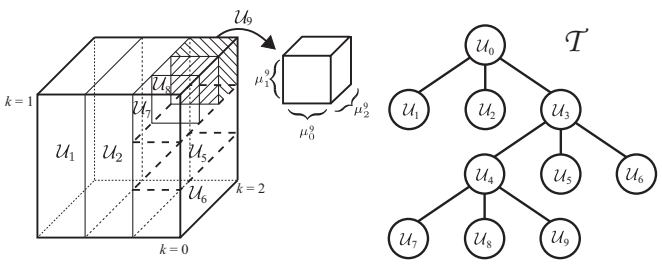

Fig. 1. Left: Example partition of U^∞ after 3 splits. Here $M = 3$, which is its smallest value with the nice feature that the center point of the parent box can be reused for the middle child box. Dimensions 4 and higher are left out of the figure. Note that boxes that have already been split ($\mathcal{U}_3, \mathcal{U}_4$) are not labeled, to avoid clutter. Right: Tree corresponding to this partition.

The collection of boxes will be organized into a tree $\mathcal{T}$ with the root consisting of the entire space, and where each node has M children, one for each of the M boxes resulting from its splitting, see Figure 1, right. Each node is labeled by index i of the box, as well as by the box itself, and we will use these notations interchangeably. The depth h of a box i in this tree is therefore equal to the number of splits needed to obtain the box, and the root has depth 0. Denote by $s_i : \{0, 1, \dots\} \to \{0, 1, \dots\}$ a function that gives the number of splits along dimension k, so $s_i(k) = 0$ when $k \ge K_i$. We have:

$$h = \sum_{k=0}^{\infty} s_i(k) \tag{3}$$

Note that $d_k^i = M^{-s_i(k)}$. Note also that a given box may be obtained along multiple paths along the tree, but for simplicity the algorithm does not make use of this information, and keeps all the copies. In a practical implementation it is of course advisable to merge them. Denote the leaves of $\mathcal{T}$ by $\mathcal{L}$; at any iteration, these leaves must be considered for splitting, and a leaf with the largest upper bound on the values of sequences in its box is selected.

To find the upper bounds, a crucial requirement is a Lipschitz property of v with respect to its argument $\boldsymbol{u}_\infty$.

Lemma 2: Given Assumption 1, for any $\boldsymbol{u}_\infty, \boldsymbol{u}'_\infty \in U^\infty$:

$$|v(\boldsymbol{u}_\infty) - v(\boldsymbol{u}'_\infty)| \leq \frac{L_\rho}{1-\gamma L_f} \sum_{k=0}^{\infty} \gamma^k |u_k - u'_k| \quad (4)$$

In addition to this Lipschitz property, an important observation is that only the rewards $r_0^i, \dots, r_{K_i-1}^i$ of the sample sequence $u_0^i, \dots, u_{K_i-1}^i$ are known. Nevertheless, the other rewards are at most 1, and this fact can be used to complete the bound. Extend by convention the finitely long sample sequences of actions and rewards of the box into infinitely long versions by appending zeros, so that $\boldsymbol{u}_\infty^i = (u_0^i, \dots, u_{K_i-1}^i, 0, 0, \dots)$, and let:

$$v(\boldsymbol{u}_\infty^i) = \sum_{k=0}^{K_i-1} \gamma^k r_k^i$$

Define $L_v = \frac{\max\{1, L_\rho\}}{1-\gamma L_f}$ and a new metric:

$$|v(\boldsymbol{u}_\infty) - v(\boldsymbol{u}'_\infty)| \leq L_v \sum_{k=0}^{\infty} \gamma^k |u_k - u'_k| =: \ell(\boldsymbol{u}_\infty, \boldsymbol{u}'_\infty) \quad (5)$$

Then, for any $\boldsymbol{u}_\infty \in \mathcal{U}_i$ we have:

$$\begin{aligned} v(\boldsymbol{u}_\infty) &\leq v(\boldsymbol{u}_\infty^i) + \sup_{\boldsymbol{u}_\infty \in \mathcal{U}_i} \ell(\boldsymbol{u}_\infty, \boldsymbol{u}'_\infty) \\ &\leq v(\boldsymbol{u}_\infty^i) + L_v \sum_{k=0}^{\infty} \gamma^k d_k^i =: b(\mathcal{U}_i) \end{aligned} \quad (6)$$

The upper-bound property holds because, given $d_k^i = 1$ for $k \geq K_i$, the tail of the sum satisfies $L_v \sum_{k=K_i}^{\infty} \gamma^k d_k^i \geq \frac{\gamma^{K_i}}{1-\gamma}$, and this latter term is an upper bound on the discounted return after step K_i. So, at the expense of some conservativeness due to the constant L_v, we have eliminated the need to separately handle discretized and undiscretized dimensions.

The upper bound $b(\mathcal{U}_i)$ is also called a b-value, and the quantity $\delta(\mathcal{U}_i) := L_v \sum_{k=0}^{\infty} \gamma^k d_k^i$ is a diameter of the box $\mathcal{U}_i$, which measures the uncertainty on the values in this box.

Thus, the algorithm starts with the full box U^∞, and proceeds by splitting at each iteration an optimistic box $i^\dagger$ that maximizes the b-value (6). The only remaining question is which dimension of this box to split, and the choice is intuitively clear – split one that has maximal contribution to the diameter, so as to minimize the resulting uncertainty:

$$\arg\max_k L_v \gamma^k d_k^i = \arg\max_k \gamma^k d_k^i \quad (7)$$

This maximization will produce at most dimension K_i, since its contribution is larger than all later dimensions. So, the method either refines further an already discretized dimension, or starts splitting the first undiscretized dimension. The overall resulting algorithm is called *optimistic planning with continuous actions*, OPC, see Algorithm 1. At the end, it returns a sample sequence with the largest value v.

Algorithm 1 Optimistic planning with continuous actions

input: state x_0, model f, ρ, split factor M, budget n
initialize $\mathcal{T}$ with root $\mathcal{U}_0 = U^\infty$
while computation n not exhausted **do**
  select box $i^\dagger = \arg\max_{i \in \mathcal{L}} b(\mathcal{U}_i)$
  select $k^\dagger = \arg\max_k \gamma^k d_k^{i^\dagger}$
  expand $\mathcal{U}_{i^\dagger}$ along $k^\dagger$: create its M children on $\mathcal{T}$
end while
output $\hat{\boldsymbol{u}} := \boldsymbol{u}_\infty^{i^*}$ where $i^* = \arg\max_{i \in \mathcal{L}} v(\boldsymbol{u}_\infty^i)$

Computation is measured by the number n of evaluations of the model, i.e. of the pair f, ρ, since for a nonlinear system simulating f is often expensive. Note that each box expansion may take up to $MK_{i^\dagger}$ evaluations. In our experiments, we always use $M = 3$, sample at interval centers, and reuse samples, which leads to a cost per box expansion of 3 model calls when $k^\dagger = K_{i^\dagger}$, and $2(K_{i^\dagger} - k^\dagger)$ otherwise. On reaching the budget limit algorithm may either be allowed to expand its last box, or immediately stopped while rolling back the changes made at the interrupted step.

OPC is similar to LP in [3], [7]. However, LP does not use the streamlined expression (5) for the metric, instead stopping at an intermediate formula. This prevents the important insight that each dimension contributes with factor γ^k in the metric. The SOOP algorithm in [3] does empiricaly apply this intuition but for a discount factor α in the metric, different from γ – and without a theoretical justification.

Our metric allows us to analyze in detail the near-optimality of OPC, in the next section. Importantly, although OPC applies the main *principle* of DOO, the *guarantees* of DOO cannot be directly applied, since the boxes obtained do not satisfy certain geometric properties required by DOO [14]. Thus, we need to provide novel analysis, adapted to the setting of continuous actions.

IV. Analysis

The main analytical objective is to characterize the sub-optimality of the algorithm as a function of the computational budget n invested. First, we will provide an a posteriori guarantee, which is standard for the optimistic class of algorithms, and shows that the sub-optimality is at most the smallest diameter among expanded boxes (Proposition 3). The major novelty of the analysis is the characterization of this diameter, which is nontrivial due to the infinite dimensionality of each box (Theorem 4). Finally, we will define a measure of problem complexity, in the form of an effective branching factor of the subtree explored (Definition 5), and using this put the computational budget in relation to the smallest diameter above, leading to an a priori near-optimality result (Theorem 6). Due to space limitations,

we only provide a proof sketch for the most interesting result – Theorem 4.

Proposition 3: OPC only expands boxes $\mathcal{U}$ that satisfy $b(\mathcal{U}) \geq v^*$. Further, the sub-optimality $v^* - v(\hat{\boldsymbol{u}})$ of the returned sequence is at most $\delta_{\min}$, the smallest diameter among all expanded boxes.

Since $\delta_{\min}$ is known after the algorithm terminates, Proposition 3 provides a directly computable, a posteriori bound. We are however interested in a stronger, a priori guarantee. To this end we characterize the decrease of the diameters with the depth in the tree.

Theorem 4: If $M > 1/\gamma$, then $\exists c > 0$ so that for each box $\mathcal{U}$ at depth h, $\delta(\mathcal{U}) \leq c\sqrt{2h(\tau-1)}\gamma^{\sqrt{2h\frac{\tau-1}{\tau^2}}} =: \delta_h$, where $\tau = \left\lceil \frac{\log 1/M}{\log \gamma} \right\rceil$ and $\lceil \cdot \rceil$ denotes the ceiling operator (the smallest integer at least as large as the argument).

Proof: (sketch) To upper-bound the diameter of an arbitrary box $\mathcal{U}$, notice first that due to dimension selection (7), all the boxes at a given depth h have the same shape. Recall function $s(k)$, which (given $M > 1/\gamma$) decreases in steps of at most 1, see also Figure 2. Denote the lengths of the constant ranges in s by $\tau_0, \tau_1, \ldots, \tau_N$ where N is the last, infinitely long range with $s = 0$. It can be shown by induction that:

$$\begin{cases} \tau_0 \leq \tau \\ \tau_i \in \{\tau - 1, \tau\} \quad \text{for } 1 \leq i < N \\ \tau_N = \infty \end{cases} \tag{8}$$

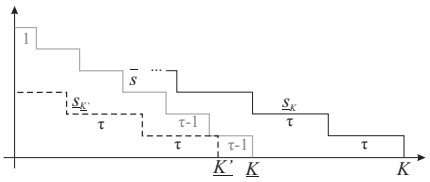

Fig. 2. Various split functions used in the proof. Continuous black: $\underline{s}_K$ for arbitrary K; gray: $\bar{s}$ yielding $\underline{K}$; dashed black: $\underline{s}_{\underline{K}'}$ where $\underline{K}' \leq \underline{K}$.

The next step is to find, for fixed h, a lower bound on $s(k)$ under constraints (3), (8). If the length K of the box is fixed and (3) is ignored (thereby relaxing the problem), then a lower bound $\underline{s}_K$ on any split function for this K is obtained by filling in s with ranges of τ; see Figure 2. Now $\underline{s}_K(k)$ is decreasing with K for any k, so we must find a lower bound on K, denoted $\underline{K}$. To this end, we fill in a function $\bar{s}$ with all ranges of $\tau - 1$, except τ_0 which is equal to 1, see again Figure 2; while imposing a relaxed version of (3), namely $h \leq \sum_{k=0}^{\infty} \bar{s}(k)$. After some calculation we get $\underline{K} \geq 4 - 2\tau + \sqrt{2h(\tau-1)} =: \underline{K}'$. Using this, we obtain a lower-bound split function as $\underline{s}_{\underline{K}'}$, which is $\sqrt{2h\frac{\tau-1}{\tau^2}} - \frac{k}{\tau} + (4/\tau - 2)$ for $k < \underline{K}'$ and 0 otherwise.

Plugging this function in the diameter $\delta(\mathcal{U})$, we get:

$$L_v \sum_{k=0}^{\infty} \gamma^k M^{-s(k)} \leq L_v \left[\sum_{k=0}^{\underline{K}'-1} \gamma^k M^{-\underline{s}_{\underline{K}'}} + \frac{\gamma^{\underline{K}'}}{1-\gamma} \right]$$
$$\leq c\sqrt{2h(\tau-1)}\gamma^{\sqrt{2h\frac{\tau-1}{\tau^2}}}$$

with c a positive constant, which is the desired result. ■

The main intuition for Theorem 4 is that since $\gamma < 1$, the term $\gamma^{\sqrt{2h\frac{\tau-1}{\tau^2}}}$ asymptotically dominates $\sqrt{2h(\tau-1)}$, and makes the diameter converge to zero; in formal asymptotic notation[1], $\delta_h = \tilde{O}(\gamma^{\sqrt{2h\frac{\tau-1}{\tau^2}}})$. Convergence is exponential not directly in the depth h, but in its square root modulated by $2(\tau-1)/\tau^2$. Condition $M > 1/\gamma$ imposed in the proposition is not restrictive since γ is often taken close to 1, and for $\gamma > 0.5$, $M = 2$ already suffices. This allows avoiding the rarely seen case $M \leq 1/\gamma$.

From Proposition 3, at depth h OPC only expands boxes in the set:

$$\mathcal{T}_h^* = \{\mathcal{U} \text{ at } h \,|\, b(\mathcal{U}) \geq v^*\}$$

This set will generally *not* contain all the nodes at h in the full tree $\mathcal{T}$. The final concept in deriving an a priori guarantee is a characterization of the size of $\mathcal{T}_h^*$.

Definition 5: The asymptotic branching factor is the smallest m so that $\exists C > 0$ for which $|\mathcal{T}_h^*| \leq Cm^h, \forall h$, where $|\cdot|$ denotes set cardinality.

Note that m may be noninteger but lies in the interval $[1, M]$ – since there is at least one node in $\mathcal{T}_h^*$, the one containing the optimal solution; and at most M^d (the entire set of nodes). This is very similar to the asymptotic branching factor in OPD [8] and plays a related role to other measures of problem complexity used to characterize optimistic methods, e.g. [4], [15]. However, its meaning is different in our continuous-action optimal control problem.

Theorem 6: For large budgets n, when $m > 1$ we have: $v^* - v(\hat{\boldsymbol{u}}) = \tilde{O}(\gamma^{\sqrt{\frac{2(\tau-1)\log n}{\tau^2 \log m}}})$, while when $m = 1$ $v^* - v(\hat{\boldsymbol{u}}) = \tilde{O}(\gamma^{n^{1/4}a})$ where $a = \sqrt{\frac{2(\tau-1)}{\tau^2\sqrt{MC}}}$ and C is the constant from Definition 5.

Thus, the convergence rate of the algorithm is modulated by the problem complexity as expressed by branching factor m. The smaller m, the easier the problem and the faster the bound reduces with increasing n. In particular, when $m = 0$ the bound is exponential in $n^{1/4}$ – faster than in the general case where the dependence is in $\log n$.

V. Simulation Results

OPC is evaluated for an example with linear dynamics, and a strongly nonlinear system.

A. DC motor

Consider a DC motor with states: shaft angle $x_1 \in [-\pi, \pi]$ rad, angular velocity $x_2 \in [-15\pi, 15\pi]$ rad/s, and action: voltage $u \in [-10, 10]$ V, where the bounds are enforced by saturation. The dynamics are linear:

$$f(x,u) = Ax + Bu, \quad A \approx \begin{bmatrix} 1 & 0.0095 \\ 0 & 0.9100 \end{bmatrix}, \; B \approx \begin{bmatrix} 0.0084 \\ 1.6618 \end{bmatrix}$$

The goal is to stabilize both states at zero, and is described by the unnormalized reward function $\tilde{\rho}(x,u) = -(x_1^2 + 0.001x_2^2)$, which is normalized to $[0,1]$ using the known

[1]Let $g, h : (0,\infty) \to \mathbb{R}$. Statement $f(t) = \tilde{O}(g(t))$ means that $\exists a > 0, b \geq 0$ so that $f(t) \leq a(\log g(t))^b g(t), \forall t > 0$. When the statement is made for large t, the inequality must hold for $\forall t \geq t_0$ where $t_0 > 0$.

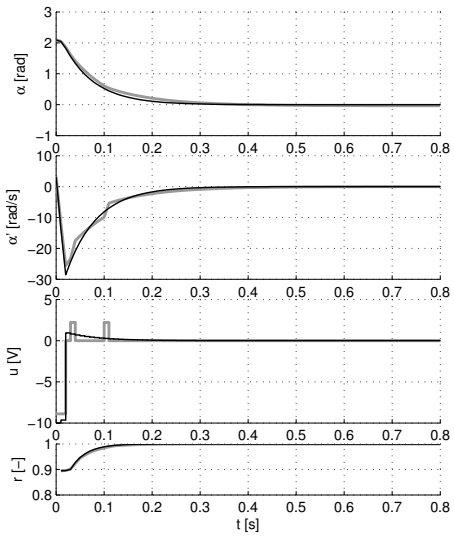

Fig. 3. Controlled trajectory of the DC motor with OPC (gray) and the optimal solution (black), from $x_0 = [2\pi/3, \pi]^\top$.

bounds on the states. The input is also normalized to $[0, 1]$ to apply OPC. We consider this system because of its simplicity: disregarding the saturation, an optimal solution can be computed analytically as in Ch. 3 of [2], and we compare OPC with this solution. Although here Lipschitz constants might be found, in general they are difficult to compute so we treat L_v as a tuning parameter and set it to 10. The budget n is 500, which is sufficient to find a good solution.

Figure 3 shows the results. Although of course the algorithm only approximately discretizes the continuous sequences so the controls are coarser than the optimal ones, the trajectories of the states are very close and the rewards nearly the same; indeed the discounted returns along the trajectory are 2.2668 for OPC and 2.2681 for the optimal solution.

B. Rotational pendulum

Our second example is the Quanser rotational pendulum, see Figure 4. This system consists of a heavy rod sitting on an unactuated rotational joint at the end of an intermediate, horizontal link actuated through a motor. The problem has four state variables: the angle $\alpha \in [-\pi, \pi)$ rad of the pendulum (zero pointing up), the angle $\theta \in [-\pi, \pi)$ rad of the horizontal link (zero pointing forward), their angular velocities $\dot\alpha, \dot\theta$ in $[-100, 100]$ rad/s, and the input voltage in $[-9, 9]$ V. The two angles "wrap around" the ends of the interval to obtain the true, manifold state space of the system. The dynamics are:

$$\ddot\alpha = \frac{ad\sin\alpha - b^2\dot\alpha^2\sin\alpha\cos\alpha - be\dot\theta\cos\alpha + bfu\cos\alpha}{ac - b^2\cos^2\alpha}$$
$$\ddot\theta = \frac{-bc\dot\alpha^2\sin\alpha + bd\sin\alpha\cos\alpha - ce\dot\theta + cfu}{ac - b^2\cos^2\alpha}$$

where $a = 0.0112$, $b = 0.0046$, $c = 0.0048$, $d = 0.2099$, $e = 0.0729$, $f = 0.1281$ are computed from the physical parameters. The goal is to reach the zero equilibrium, but

Fig. 4. Rotational pendulum.

the system is underactuated so the pendulum must first be swung back and forth to accumulate energy, which means that long trajectories must be found; the horizontal link further complicates the dynamics. The problem is therefore challenging for planning methods. The unnormalized reward function is $\tilde\rho(x, u) = -(\alpha^2 + 0.05\theta^2)$. The sampling time is chosen 0.05 s.

In this problem we use OPC with a tighter b-value, which exploits the knowledge that rewards are at most 1 and that action samples are always at the centers of the intervals:

$$b(\mathcal{U}_i) = \sum_{k=0}^{K_i-1} \gamma^k \min\{1, r_k^i + L_\rho \sum_{j=0}^{k} L_f^{k-j} \frac{d_j^i}{2}\} + \frac{\gamma^{K_i}}{1-\gamma} \quad (9)$$

We compare OPC with the two algorithms closest to it: LP [7] in a variant that uses bound (9) but selects dimensions using its own rule, different from OPC; and SOOP [3] which does not compute upper bounds at all, but expands all the boxes that may be optimistic in a certain sense. Recall that SOOP uses a criterion similar to (7) to select dimensions, but with a tuning parameter α replacing γ.

We set $\gamma = 0.85$ and a large budget of $n = 2000$ so that the algorithms can find a near-optimal solution. We treat again the Lipschitz constants as tuning parameters and to study their impact on performance, we set them equal $L_\rho = L_f =: L$ and vary L in the set $\{0.7, 0.8, \ldots, 1.2, 1.5, 2, 2.5\}$, for both OPC and LP. For SOOP α is varied in $\{0.5, 0.7, 0.8, 0.85, 0.9, 0.95\}$. The algorithms are run for 80 steps from initial state $\alpha = -\pi$, $\dot\alpha = \theta = \dot\theta = 0$, and the resulting returns are shown in Figure 5. The first observation is that Lipschitz constants around 1 perform best in OPC. OPC outperforms LP for nearly all values of L; since they use the same upper bound and only the dimension selection is different, this illustrates that the insight in Lemma 2 and (7) on the impact of the different dimensions pays off. Finally, even though SOOP is not theoretically well fundamented, it performs better than both OPC and LP. This is likely because SOOP does not use upper bounds so it does not suffer from the limitations of (9) (over- or underestimation of L, fixed L whereas smoothness

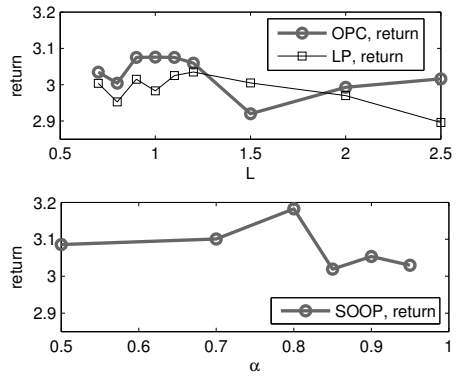

Fig. 5. Compared performance of OPC, LP, and SOOP.

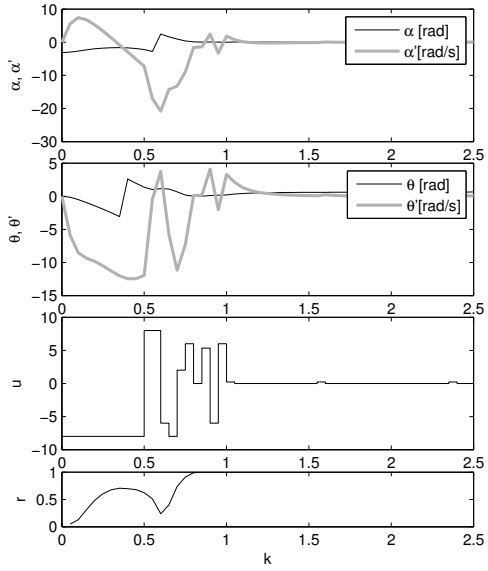

Fig. 6. Rotational pendulum trajectory with OPC, for $L = 1$.

varies over the state space, etc.) This result strongly points towards adapting SOOP so that it can be analyzed in the framework of Section IV.

The actual controlled behavior with OPC is illustrated in Figure 6. The swingup is achieved by applying large actions, while near the equilibrium small corrections are applied to keep the pendulum upright – which illustrates the benefits of the adaptive discretization procedure in OPC.

VI. Conclusions and future work

We have presented optimistic planning for continuous actions, which works for general nonlinear systems with scalar inputs. The convergence rate of the algorithm to the optimal value has been analyzed as a function of the computation invested. Empirical results showed that the algorithm works in practice, while an alternative algorithm called SOOP worked better in a nonlinear problem, despite its lack of guarantees.

This last result strongly points towards a promising direction of future work: extending OPC to sidestep the explicit usage of upper bounds like in SOOP, while keeping the algorithm compatible with the theoretical framework so that it can be analyzed. A practical step is extending the method to work with multiple action variables.

References

[1] D. Antunes, W. Heemels, and P. Tabuada, "Dynamic programming formulation of periodic event-triggered control: Performance guarantees and co-design," in *IEEE Conference on Decision and Control, Hawai: U.S.A.*, 2012, pp. 7212–7217.

[2] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Athena Scientific, 2007, vol. 2.

[3] L. Buşoniu, A. Daniels, R. Munos, and R. Babuška, "Optimistic planning for continuous–action deterministic systems," in *2013 IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL-13)*, Singapore, 16–19 April 2013.

[4] L. Buşoniu and R. Munos, "Optimistic planning for Markov decision processes," in *Proceedings 15th International Conference on Artificial Intelligence and Statistics (AISTATS-12)*, ser. JMLR Workshop and Conference Proceedings, vol. 22, La Palma, Canary Islands, Spain, 21–23 April 2012, pp. 182–189.

[5] L. Buşoniu, E. Páll, and R. Munos, "An analysis of optimistic, best-first search for minimax sequential decision making," in *2014 IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL-14)*, Orlando, 10–12 December 2014.

[6] L. Grüne and J. Pannek, *Nonlinear Model Predictive Control: Theory and Algorithms*. Springer, 2011.

[7] J.-F. Hren, "Planification optimiste pour systèmes déterministes," Ph.D. dissertation, Lille 1 University - Science and Technology, 2012.

[8] J.-F. Hren and R. Munos, "Optimistic planning of deterministic systems," in *Proceedings 8th European Workshop on Reinforcement Learning (EWRL-08)*, Villeneuve d'Ascq, France, 30 June – 3 July 2008, pp. 151–164.

[9] K. Katsikopoulos and S. Engelbrecht, "Markov decision processes with delays and asynchronous cost collection," *IEEE Transactions on Automatic Control*, vol. 48, no. 4, pp. 568–574, 2003.

[10] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *Proceedings 17th European Conference on Machine Learning (ECML-06)*, Berlin, Germany, 18–22 September 2006, pp. 282–293.

[11] S. M. La Valle, *Planning Algorithms*. Cambridge University Press, 2006.

[12] C. Mansley, A. Weinstein, and M. L. Littman, "Sample-based planning for continuous action Markov decision processes," in *Proceedings 21st International Conference on Automated Planning and Scheduling*, Freiburg, Germany, 11–16 June 2011, pp. 335–338.

[13] K. Máthé, L. Buşoniu, R. Munos, and B. D. Schutter, "Optimistic planning with a limited number of action switches for near-optimal nonlinear control," in *Proceedings 53nd Conference on Decision and Control (CDC-14)*, Los Angeles, USA, 15–17 December 2014, pp. 3518–3523.

[14] R. Munos, "Optimistic optimization of a deterministic function without the knowledge of its smoothness," in *Advances in Neural Information Processing Systems 24*, J. Shawe-Taylor, R. S. Zemel, P. L. Bartlett, F. C. N. Pereira, and K. Q. Weinberger, Eds., 2011, pp. 783–791.

[15] ——, "The optimistic principle applied to games, optimization and planning: Towards foundations of Monte-Carlo tree search," *Foundations and Trends in Machine Learning*, vol. 7, no. 1, pp. 1–130, 2014.

[16] M. L. Puterman, *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. Wiley, 1994.

[17] T. J. Walsh, S. Goschin, and M. L. Littman, "Integrating sample-based planning and model-based reinforcement learning," in *Proceedings 24th AAAI Conference on Artificial Intelligence (AAAI-10)*, Atlanta, US, 11–15 July 2010.

[18] A. Weinstein and M. L. Littman, "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, 25–19 June 2012.