Near-optimal control with adaptive receding horizon for discrete-time piecewise affine systems ⋆

Jia Xu * **Lucian Buşoniu** ** **Bart De Schutter** *

* *Delft University of Technology, the Netherlands*
E-mail: {j.xu-3, b.deschutter}@tudelft.nl
** *Technical University of Cluj-Napoca, Romania*
E-mail: lucian@busoniu.net

Abstract: We consider the infinite-horizon optimal control of discrete-time, Lipschitz continuous piecewise affine systems with a single input. Stage costs are discounted, bounded, and use a 1 or ∞-norm. Rather than using the usual fixed-horizon approach from model-predictive control, we tailor an adaptive-horizon method called optimistic planning for continuous actions (OPC) to solve the piecewise affine control problem in receding horizon. The main advantage is the ability to solve problems requiring arbitrarily long horizons. Furthermore, we introduce a novel extension that provides guarantees on the closed-loop performance, by reusing data ("learning") across different steps. This extension is general and works for a large class of nonlinear dynamics. In experiments with piecewise affine systems, OPC improves performance compared to a fixed-horizon approach, while the data-reuse approach yields further improvements.

Keywords: piecewise affine systems, nonlinear predictive control, optimistic planning, near-optimality analysis.

1. INTRODUCTION

Piecewise affine (PWA) systems are an important class of hybrid systems, defined by partitioning the state and input space into polyhedral regions and associating with each region a different affine dynamical description (Sontag, 1981). We focus here on discrete-time PWA systems that are continuous on the boundary of any two neighbouring polyhedral regions. These systems can approximate nonlinear smooth dynamics (Storace and De Feo, 2004).

For discrete-time PWA systems, several techniques have been developed for different kinds of optimal control problems. Regarding the constrained finite-time optimal control problem based on quadratic or linear performance criteria, a solution approach combining multiparametric programming and dynamic programming is presented by Borrelli et al. (2005). The optimal solution is a PWA state-feedback control law. Bemporad and Morari (1999) translate the aforementioned problem into a linear or quadratic mixed-integer programming problem that can be solved by using standard solvers. The constrained optimal control problem with a linear performance criterion is considered by Baoti et al. (2006).

On the other hand, optimal control problems with infinite-horizon costs arise in many fields (Bertsekas, 2012). The standard approach in model predictive control is to solve a fixed, finite-horizon problem at each step, and apply the first input of each of these solutions to obtain a closed-loop, receding horizon control (Grüne and Pannek, 2011). We followed this approach for PWA systems in Xu et al. (2016), where we applied a near-optimal, optimistic optimization algorithm (Munos, 2011). However, in problems where long horizons are necessary, such fixed-horizon approaches quickly become unfeasible due to uncontrolled growth of the computation with the horizon.

Here we aim to address this shortcoming, in the context of PWA systems and optimistic approaches. We thus focus on optimistic *planning* (Munos, 2014), rather than optimization – a class of *adaptive-horizon* approaches to solve discounted, infinite-horizon optimal control. Optimistic methods have their roots in reinforcement learning (Sutton and Barto, 1998) and bandit theory (Auer et al., 2002). In particular, we use our continuous-action optimistic planner (OPC) from (Buşoniu et al., 2016b), since it is the most suitable for PWA systems. As for the entire optimistic family, the performance guarantees of OPC place computation in a tight relationship with the near-optimality of the returned sequence. Unlike in Xu et al. (2016) however, OPC automatically balances refining the current-horizon solution, with extending this horizon; so it finds long solutions which are provably close to the true, infinite-horizon optimum.

Our first contribution is to adapt and evaluate OPC for discrete-time continuous PWA systems with a single input. The stage costs are weighted 1-norms or ∞-norms of the deviation of the state and input from the equilibrium point. This solution directly imports the near-optimality of OPC *at each call*, but may not be optimal in receding

⋆ This work was supported by the Chinese Scholarship Council, as well as by the Agence Universitaire de la Francophonie (AUF) and the Romanian Institute for Atomic Physics (IFA) under the AUF-RO project NETASSIST.

horizon. Therefore, we introduce a novel extension of OPC called OPC+, whose main element is remembering and reusing solutions from previous calls – in a sense, learning to improve the solution during the experiment, similar to Buşoniu et al. (2016a). Thanks to this, OPC+ guarantees the near-optimality of the closed-loop solution, without sacrificing per-call optimality. While motivated by PWA control, OPC+ and its guarantees hold for any nonlinear dynamics satisfying the OPC assumptions. An empirical study is performed for two PWA examples: one where short planning (control) horizons are sufficient for a good solution, and another where long horizons are necessary.

This paper is organized as follows. In Section 2, the problem formulation is presented, and Section 3 introduces the OPC method for general systems. Section 4 adapts OPC to PWA systems, and Section 5 describes and analyzes OPC+. In Section 6, the proposed approaches are illustrated in simulations. Section 7 concludes.

2. PROBLEM STATEMENT

Consider the discrete-time PWA system:

$$x_{k+1} = A_j x_k + B_j u_k + c_j, \quad \text{if } [x_k^\top, u_k^\top]^\top \in \Omega_j \tag{1}$$

where $x \in X \subseteq \mathbb{R}^{n_x}$ is the state, $u \in U \subseteq \mathbb{R}^{n_u}$ the control input, $A_j \in \mathbb{R}^{n_x \times n_x}$, $B_j \in \mathbb{R}^{n_x \times n_u}$, $c_j \in \mathbb{R}^{n_x}$ are the system matrices and vectors, and $\{\Omega_j\}_{j=1}^M$ is a polyhedral partition of the state-input space. The polyhedron Ω_j is given as $\Omega_j = \{x, u \mid F_j x + G_j u \leq h_j\}$ where F_j, G_j, h_j are suitable matrices and vectors. The sets X and U contain all the feasible state and input values. Moreover, we require that the right-hand side of (1) is continuous on the boundary of any two neighboring regions.

Let x_0 be the initial state and define an infinite-length control sequence $\boldsymbol{u}_\infty = (u_0, u_1, \dots)$, in which u_k is the control input at time step k. Let $U^\infty := U \times U \times \cdots$ be the space of infinite-length control sequences.

Given x_0, consider the infinite-horizon discounted cost function:

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

where $\gamma^k \in (0,1)$ is the discount factor and $g : X \times U \to \mathbb{R}$ is the stage cost. The optimal control objective is:

$$J^* := \inf_{\boldsymbol{u}_\infty \in U^\infty} J(\boldsymbol{u}_\infty) \tag{3}$$

We take the following stage cost:

$$g(x_k, u_k) = \|Q x_k\|_p + \|R u_k\|_p \tag{4}$$

where $p \in \{1, \infty\}$ and $Q \in \mathbb{R}^{n_Q \times n_x}$, $R \in \mathbb{R}^{n_R \times n_u}$ are full column rank matrices. So the solution to (3) regulates the system to the zero equilibrium. Note that by a linear substitution of the variables, any non-zero equilibrium point can be translated to the origin, so focusing on regulation to the origin is not restrictive.

3. BACKGROUND ON OPC

Optimistic planning for continuous actions (OPC) (Buşoniu et al., 2016b) is geared towards general nonlinear systems and maximization of cumulative rewards, so we present it in that setting first. Later, we explain how it can be adapted to solve the problem in Section 2. Given initial state $x_0 \in X$, OPC near-optimally solves:

$$v^* := \sup_{\boldsymbol{u}_\infty \in U^\infty} v(\boldsymbol{u}_\infty) := \sup_{\boldsymbol{u}_\infty \in U^\infty} \sum_{k=0}^{\infty} \gamma^k r(x_k, u_k) \tag{5}$$

where $\gamma, \boldsymbol{u}_\infty, X, U$ have the same meaning as before, $v : U^\infty \to \mathbb{R}$ is the value function to be maximized, $r : X \times U \to \mathbb{R}$ is the reward function, $f : X \times U \to X$ is the nonlinear dynamics, and v^* is the optimal value at x_0.

The OPC method works under the following assumptions:

Assumption 1. (i) Rewards are bounded in $[0, 1]$.

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

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

$$\|f(x,u) - f(x',u')\| \leq L_f(\|x - x'\| + |u - u'|)$$
$$|r(x,u) - r(x',u')| \leq L_r(\|x - x'\| + |u - u'|)$$

for some norm $\|\cdot\|$.

(iv) The discount factor $\gamma \in (1/3, 1/L_f)$.

We describe in Section 4 the impact of these assumptions in PWA systems, see Buşoniu et al. (2016b) for further discussion.

OPC performs a search over the space of action sequences U^∞, which can be visualized as an infinite-dimensional hypercube, with each dimension k the action space at step k. This hypercube is repeatedly refined into smaller boxes, each of which gets a unique index i. A box $\mathcal{U}_i \subseteq U^\infty$ is the cross-product of a sequence of intervals $(\mu_{i,0}, \dots, \mu_{i,K_i-1}, U, U, \dots)$ where $\mu_{i,k} \subseteq U$ and $K_i - 1$ is the largest discretized dimension; for all further dimensions $\mu_{i,k} = U$. Define $d_{i,k}$ to be the length of the interval $\mu_{i,k}$ in box i, and $u_{i,k}$ a sample action taken at the center of this interval. The center sequence of box i is $(u_{i,0}, \dots, u_{i,K_i-1})$. For each box, the rewards $r_{i,k}$ obtained by applying $u_{i,k}$ from x_0 are found by simulating the system. A box is refined into 3 subboxes by splitting the interval of some dimension k into 3 equal-length pieces, see Figure 1, left. [1]

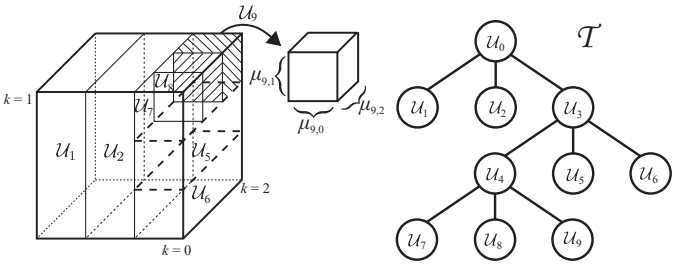

Fig. 1. Left: Example partition of U^∞ after 3 splits. Dimensions 4 and higher are left out of the figure. Right: Tree corresponding to this partition. (Figure taken from Buşoniu et al. (2016b).)

The collection of boxes is organized into a tree $\mathcal{T}$ with the root consisting of U^∞, and where each node has as children the 3 boxes resulting from its splitting, see Figure 1, right. The depth h of a box i in this tree is equal to the number

[1] This number of subintervals is the smallest for which the parent center sequence can be reused for the middle child.

of splits performed to obtain the box, with the root at depth 0. At each iteration, OPC refines an *optimistic* leaf, i.e. one that maximizes the b-value, defined as follows:

$$b(i) = v(i) + \delta(i) \tag{6}$$

Here, $v(i) = \sum_{k=0}^{K_i-1} \gamma^k r_{i,k}$ is the (horizon K_i) value of the center sequence, and $\delta(i) = L_v \sum_{k=0}^{\infty} \gamma^k d_{i,k}$ is the diameter of box i, an uncertainty on the values inside that box. Also, $L_v = \max\{L_r/(1-\gamma L_f), 1\}$ is a Lipschitz constant of the value function v. The b-value is upper bound on the values of sequences inside box i.

Once a maximal b-value box has been selected, a dimension to refine is chosen that has maximal contribution to the diameter:

$$\arg\max_{k\geq 0} L_v \gamma^k d_{i,k} = \arg\max_{k\geq 0} \gamma^k d_{i,k} \tag{7}$$

In this way, the refinement will minimize the resulting uncertainty. Although performed over infinitely many dimensions, this maximization will produce at most dimension K_i, since its contribution is larger than for later ones. So, either an already discretized dimension is refined further, or the first undiscretized dimension is split.

OPC runs until a budget n of calls to f has been exhausted, and then returns the center sequence $\widehat{\boldsymbol{u}}$ of box i^* with the largest value $v(i^*)$. Buşoniu et al. (2016b) bound its sub-optimality as follows. The nodes that OPC may expand at depth h are all in $\mathcal{T}_h^* = \{\text{nodes } i \text{ at depth } h \mid b(i) \geq v^*\}$. Let κ be the asymptotic branching factor of $\mathcal{T}^*$, defined as $\limsup_{h\to\infty} |\mathcal{T}_h^*|^{1/h}$; value κ characterizes the computational complexity of OPC. Then:

Theorem 2. (i) The sub-optimality $v^* - v(i^*) \leq \delta_{\min}$, with $\delta_{\min}$ the smallest diameter of any box expanded.
(ii) If $\kappa = 1$, $\delta_{\min} = \tilde{O}(\gamma^{n^{1/4}a})$ for a constant a.[2]
(iii) If $\kappa > 1$, $\delta_{\min} = \tilde{O}(\gamma^{\sqrt{\frac{2(\tau-1)\log n}{\tau^2 \log \kappa}}})$, with $\tau := \left\lceil \frac{\log 3}{\log 1/\gamma} \right\rceil$.

Thus the bound of Theorem 2(i) is available *a posteriori*, after the algorithm has finished; while the other two relations give *a priori* bounds.

Note that OPC can be empirically extended to multiple action dimensions, e.g. by always expanding all dimensions at the selected step k; although the analysis does not yet cover such extensions.

4. APPLYING OPC TO PWA SYSTEMS

To apply OPC to the optimal control problem of PWA systems from Section 2, we restrict the system to have a single input and assume that X and U are bounded sets. Additionally, the input is translated and rescaled to the unit interval. This ensures Assumption 1(ii). While boundedness reduces generality, in many practical systems physical limitations can be modeled by bounded states and actions. Further, define $L_g = \max\left\{\|Q\|_p, \|R\|_p\right\}$ and $\Lambda = \max_{x\in X} \|x\|_p$. Then, since $p \in \{1, \infty\}$:

$$\begin{aligned} g(x_k, u_k) &= \|Qx_k\|_p + \|Ru_k\|_p \\ &\leq \|Q\|_p \|x_k\|_p + \|R\|_p \|u_k\|_p \leq L_g(\Lambda + 1) \end{aligned} \tag{8}$$

[2] $\tilde{O}(\cdot)$ means the argument is an upper bound, up to constant and logarithmic factors.

Note that L_g is a Lipschitz constant in the p-norm for the stage cost g. Define then the reward function to be maximized as:

$$r(x, u) = 1 - \frac{g(x, u)}{L_g(\Lambda + 1)} \tag{9}$$

Since this function is in $[0, 1]$, Assumption 1(i) is satisfied.

Furthermore, in a similar way to (8), it can be proven that $L_f = \max_{j=1,\ldots,M} \max\left\{\|A_j\|_p, \|B_j\|_p\right\}$ is a Lipschitz constant for f. Since r in (9) is clearly Lipschitz with $L_r = \frac{1}{\Lambda+1}$, Assumption 1(iii) is satisfied.

Finding a feasible, good value for the discount factor γ in Assumption 1(iv) depends on L_f being not too large; to see this, rewrite the upper bound of Assumption 1(iv) as $\gamma L_f \leq 1$. This 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 γ.

Finally, solving the maximal-value problem (5) with reward function (9) is equivalent to solving the minimal-cost optimal control problem of PWA systems (3), which enables us to directly apply the OPC machinery to this latter problem. The guarantees from Theorem 2 hold after appropriately enlarging the bounds by the scaling factor of the rewards, e.g. Theorem 2(i) gives:

$$J(\hat{\boldsymbol{u}}) - J^* \leq L_g(\Lambda + 1)\delta_{\min}$$

with $J(\hat{\boldsymbol{u}})$ the finite sum of discounted stage costs up to the length of $\widehat{\boldsymbol{u}}$. Since Theorem 2(ii)-2(iii) gives asymptotic bounds that disregard constants, these remain the same when applied to the costs.

OPC is usually implemented in a receding-horizon scheme. The first component of $\widehat{\boldsymbol{u}}$ is applied to the system, resulting in a new state x_1. Subsequently, the procedure is repeated using x_1 as the updated initial state. This process leads to a receding-horizon controller. However, preserving the near-optimality guarantees for this closed-loop solution turns out to be nontrivial, and we dedicate the next section to this issue.

5. AN IMPROVED RECEDING-HORIZON ALGORITHM

A desirable property of receding-horizon algorithms is that closing the loop increases the cumulative reward. This is difficult for the original OPC. Indeed, examples can be constructed in which the solution constructed by OPC when applied at step $k = 1$, from x_1, has worse cumulative rewards than the tail of the initial sequence found at $k = 0$. Here, we consider the general case when arbitrary-length subsequences may be applied in-between loop closures/algorithm calls. Let the index of an OPC call be denoted by j. At call $j = 0$ OPC finds sequence $\boldsymbol{u}_0$ and applies an initial, head subsequence denoted $\boldsymbol{u}_0^H$, having length N_0 with N_0 at least 1 and at most the largest discretized dimension in $\boldsymbol{u}_0$. Then, OPC is called again at step N_0, where it returns $\boldsymbol{u}_1$, from which a head subsequence $\boldsymbol{u}_1^H$ of length N_1 is applied, and so on.

To avoid returning sequences with smaller values, we propose to initialize the algorithm at call $j + 1$, which occurs at step $k_{j+1} = \sum_{j'=0}^{j} N_j$, with the entire collection

of tails of $\boldsymbol{u}_j^H$ available at call j. Formally, if i_j^* is the optimal box at call j, then the following memory of tail subboxes is created:

$$\mathcal{M}_{j+1} = \{\mu_{i,N_j} \times \mu_{i,N_{j+1}} \times \dots \mid \text{node } i \in \mathcal{M}_j \\ \text{s.t. } K_i \geq N_j \text{ and } u_{i,k} = u_{i_j^*,k} \ \forall k < N_j\} \quad (10)$$

where $\mathcal{M}_0 = \emptyset$.

Before describing how these boxes are reused at call $j+1$, we ensure that they are actually valid for reuse, which is nontrivial because OPC only examines boxes of certain shapes, driven by the dimension selection rule (7).

Lemma 3. Assume that during box expansion, ties in dimension selection are always broken in favor of the smallest k. Then, for any j, boxes in the set $\mathcal{M}_{j+1}$ are valid for OPC when called at step k_{j+1}.

Proof. Take any set i expanded at call j, and let $k^\dagger$ be the dimension selected for expansion. (i) If $k^\dagger < N_j$, this expansion has no impact on the shape of sets in $\mathcal{M}_{j+1}$. (ii) Otherwise:

$$k^\dagger \in \operatorname*{arg\,max}_{k'=N_j,\dots,K_i} \gamma^k d_{i,k} = \operatorname*{arg\,max}_{k'=N_j,\dots,K_i} \gamma^{k-N_j} d_{i,k}$$

The first equality holds because $k^\dagger$ maximizes the impact $\gamma^k d_{i,k}$ among *all* dimensions, so also along dimensions above N_j, and we know it was among these dimensions since we are in case (ii). The second equality is obtained by simply dividing the maximized expression by γ^{N_j}, which does not change the result. But this latest maximization is the one applied by OPC at $j+1$, see again (7). Thus, whenever dimensions larger than N_j are split, they are split in the same order as they would have been at $j+1$. So the tail subboxes have correct shapes. ■

Algorithm 1 OPC+ at call j

1: **input:** x, f, r, n, L_v, memory $\mathcal{M}_j$
2: set $\mathcal{M}_j = \mathcal{M}_j \cup \{\text{root node labeled by box } U^\infty\}$
3: **while** computation budget n not exhausted **do**
4: select box $i^\dagger = \arg\max_{i \in \mathcal{M}_j} b(i)$
5: select $k^\dagger = \arg\max_k \gamma^k d_{i^\dagger,k}$
6: **for** $i' = 1$ to 3 **do** ▷ expand box $i^\dagger$ along $k^\dagger$
7: **if** new box i' not found in $\mathcal{M}_j$ **then**
8: create new box i' and add it to $\mathcal{M}_j$
9: **end if**
10: **end for**
11: remove parent box $i^\dagger$ from $\mathcal{M}_j$
12: **end while**
13: create new memory $\mathcal{M}_{j+1}$ with (10)
14: **output** $\boldsymbol{u}_j$, sequence of box $i_j^* = \arg\max_{i \in \mathcal{M}_j} v(i)$, and $\mathcal{M}_{j+1}$

Algorithm 1 gives the modified OPC variant with box reuse, which we call OPC+. A budget n is allotted per call. At every split, the algorithm checks whether any of the resulting boxes are already available in the memory, and if yes, it reuses them. For simplicity, the algorithm is stated in a way that only works on the memory, rather than creating a tree, but it is easy to see that when $\mathcal{M}_j$ is empty, this way of working reduces to the original OPC (the memory is always the set of leaves considered for expansion). For computational efficiency, it is recommended to remove duplicates from each set $\mathcal{M}_{j+1}$ when creating it at line 13.

The main advantage of the modified algorithm is that the cumulative sum of rewards is improved by closing the loop, as proven next. Let $[\cdot]$ denote the concatenation of the argument sequences, and $v(\boldsymbol{u})$ for a finite sequence $\boldsymbol{u}$ denote the partial discounted sum of rewards of this sequence, up to its length.

Theorem 4. Consider the closed-loop sequence obtained up to call j by the applying OPC+ using the receding-horizon procedure above, $\boldsymbol{u}_j^C = [\boldsymbol{u}_0^H, \boldsymbol{u}_1^H, \dots, \boldsymbol{u}_{j-1}^H, \boldsymbol{u}_j]$. Then, for any $j \geq j'$, $v(\boldsymbol{u}_j^C) \geq v(\boldsymbol{u}_{j'}^C)$.

Proof. Consider any call $j \geq 1$. By construction, $v(\boldsymbol{u}_j) \geq v(\boldsymbol{u}_{j-1}^T)$, because the tail sequence $\boldsymbol{u}_{j-1}^T$ is among the sequences considered by the algorithm at line 14. This leads to:

$$\begin{aligned} v([\boldsymbol{u}_{j-1}^H, \boldsymbol{u}_j]) &= v(\boldsymbol{u}_{j-1}^H) + \gamma^{K_{j-1}} v(\boldsymbol{u}_j) \\ &\geq v(\boldsymbol{u}_{j-1}^H) + \gamma^{K_{j-1}} v(\boldsymbol{u}_{j-1}^T) = v(\boldsymbol{u}_{j-1}) \end{aligned}$$

When $j \geq 2$, the value of the earlier sequence $[\boldsymbol{u}_0^H, \dots, \boldsymbol{u}_{j-2}^H]$ is fixed. Overall, we get $v(\boldsymbol{u}_{j+1}^C) \geq v(\boldsymbol{u}_j^C)$ which immediately implies the desired result. ■

By taking the limit as $j \to \infty$, we obtain e.g. that the complete, infinitely long closed-loop sequence is better than any finite one, and in particular than the initial sequence at the first call, $v(\boldsymbol{u}_\infty^C) \geq v(\boldsymbol{u}_0)$.

Note that the improvement property could have been obtained simply by remembering the single sequence returned by the algorithm, instead of the entire collection of tails. However, doing the latter has an additional advantage: some nodes resulting from expansions may be found in the memory instead of having to be re-simulated. Because simulating the nonlinear system – usually by numerical integration – dominates computation in practice, this should result in computation time savings, at the expense of some extra memory to store the sequences.

Let the sequence value function and the optimal value at step k_j, computed relative to the state x_{k_j}, be denoted by v_j and v_j^* respectively. It is important to realize that Proposition 4 does *not* imply that the sequence $\boldsymbol{u}_j$ returned by OPC+ at steps $j > 1$ is near-optimal *at that step*, with respect to v_j^*. Nevertheless, as shown next this property is in fact true, so the near-optimality properties of OPC are preserved by the modification. For the sake of brevity we only prove the equivalent of Theorem 2(i).

Theorem 5. Let $\delta_{\min}$ be the smallest diameter of any box expanded by OPC+, then $v_j^* - v_j(i_j^*) \leq \delta_{\min}$.

Proof. Take an arbitrary iteration of OPC+. Since the root node corresponding to the entire space is included in $\mathcal{M}$ at the start of the algorithm, it can be proven by an easy induction that there exists in $\mathcal{M}$ some box $\bar{i}$ containing an optimal solution. Thus, $b(\bar{i}) \geq v_j^*$, and since the node $i^\dagger$ expanded at this iteration maximizes the b-value, we have $b(i^\dagger) \geq v_j^*$.

Furthermore, at the end of the algorithm, there will exist a descendant node $\tilde{i}$ of $i^\dagger$ in $\mathcal{M}$ so that $v_j(\tilde{i}) \geq v_j(i^\dagger)$. This is true because the splitting rule divides intervals into three pieces, so the middle child either inherits the value

of the parent box (when an existing dimension is split) or adds a positive reward to it (when the first undiscretized dimension is split); thus each expansion creates at least one better child. Also, the box i_j^* returned satisfies $v_j(i_j^*) \geq v_j(\tilde{i})$ and so $v_j(i_j^*) \geq v_j(i^\dagger)$.

Combining the two inequalities obtained above, we get:

$$v_j^* - v_j(i_j^*) \leq b(i^\dagger) - v(i^\dagger) = \delta(i^\dagger)$$

Since the iteration was arbitrary, the result is proven. ■

The difference from the standard proof of Theorem 2(i) is the need to take into account the memory. Note that if OPC+ expands a box inherited from the previous step, that has a smaller diameter than those OPC would have expanded, then $\delta_{\min}$ is smaller for OPC+ than for OPC, and the near-optimality bound is improved.

The development in this section did not rely on the PWA structure from Section 2, so the OPC+ approach works in general, for any nonlinear dynamics and rewards satisfying the OPC Assumption 1.

6. EXAMPLES

Next, we apply OPC in two PWA examples: one where short planning (control) horizons are sufficient for a good solution, and another where long horizons are necessary. In both examples, we compare with the optimistic optimization (OO) approach from Xu et al. (2016). The OO approach is a branch-and-bound procedure similar to OPC, but it works for fixed, finite horizons, always refining all dimensions at the same rate. In contrast, the adaptive dimension selection procedure of OPC allows it to explore the space of infinite-horizon solutions, balancing the benefits of refining existing dimensions versus increasing the horizon. So we expect OPC to perform better in problems where long horizons are needed.

6.1 Adaptive cruise control

The first example is borrowed from Xu et al. (2016) and involves controlling the velocity x of a follower car so as to track the reference signal imposed by a leader car. The input u is the throttle/brake position. A scalar PWA model with two regions results, see Xu et al. (2016) for details. It should be noted that while the tracking cost function here is different from the regulation cost function in (4) that we considered above, the algorithm is easy to empirically extend to the tracking case. [3]

OO and OPC are compared in receding horizon, setting the same range of simulation budgets at each call. The tuning parameter of OO is the horizon N_p (the prediction and control horizons are taken equal) and we try values between 2 and 6. We aim to minimize the cumulative cost without discounting. This matches the cost function of OO, which is also undiscounted. For OPC, we treat the discount factor γ as a tuning parameter and try values $0.5, 0.6, \ldots, 0.9$. The results for the best and worst values of the tuning parameters of the two algorithms are shown in Figure 2. Note that n values are slightly different between the algorithms, because they are allowed to finish the last iteration even when it exceeds the imposed budget. We report the actual budget they used rather than the initial value set.

[3] The theoretical analysis for the tracking case is left for future work.

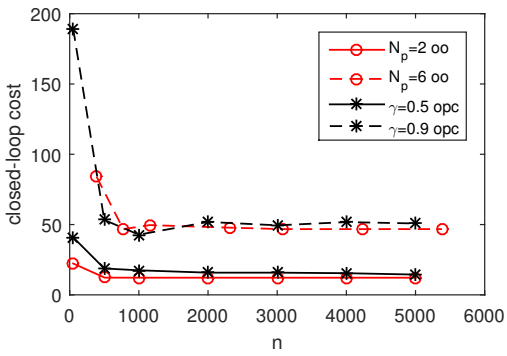

Fig. 2. Adaptive cruise control results.

Clearly, short horizons are best in this problem (choosing a smaller value for γ is intuitively similar to picking a shorter horizon, because the discounting gets close to zero after a smaller number of steps). Furthermore, because of this property, fixed-horizon OO solves the problem well. The performance of OPC is however very close to that of OO, so the loss incurred by applying the "inappropriate" OPC approach in this problem is small, indicating that OPC is a good choice for a default algorithm.

6.2 Inverted pendulum

Consider an inverted pendulum composed of a mass m attached to the end of a rod of length l and actuated by a DC motor, with the following dynamics:

$$\dot{x}_1 = x_2, \dot{x}_2 = J^{-1}[mgl\sin(x_1) - bx_2 - K^2x_2/R + Ku/R]$$

where x_1 is the angle $\alpha \in [-\pi, \pi)$ [rad] of the pendulum, x_2 is the angular velocity $\dot{\alpha} \in [-15\pi, 15\pi]$ [rad/s], and the control input $u \in [-1.25, 1.25]$ [V] is the voltage of the motor. The other parameters are $J = 10^{-4}$ [kg·m^2], $m = 0.03$ [kg], $l = 0.042$ [m], $b = 3 \cdot 10^{-6}$ [Nms/rad], $K = 0.0536$ [Nm/A], $R = 9.5$ [Ω], $g = 9.81$ [m/s^2]. The origin corresponds to the pendulum at rest, pointing up. With these voltage limits, from certain initial states such as pointing down, a destabilizing swing-up must first be performed prior to stabilization, so that a nonlinear, long-horizon solution is necessary.

The nonlinearity $\sin(x_1)$ is approximated by a continuous PWA function by partitioning the range of x_1 into 3 regions, with breakpoints selected to minimize the squared difference between the sine function and its approximation. Then, the continuous-time model is discretized with a sampling time $T_\mathrm{s} = 0.05$ [s]. A discrete-time PWA system is obtained, in the form of (1) with $M = 3$ subsystems.

We aim to design a feedback control that brings the pendulum from the state $x_0 = [-\pi, 0]^\top$ (pointing down) to $[0, 0]^\top$ (pointing up), which is expressed by the stage cost $g(x, u) = \|Qx\|_1 + \|Ru\|_1$, with $Q = \mathrm{diag}(1, 0)$ and $R = 0.001$. Figure 3 shows the results with the same algorithm settings as in the previous example. Long horizons are better in this problem, and the worst results with OPC are on par with the best ones using OO, clearly showing the benefits of OPC.

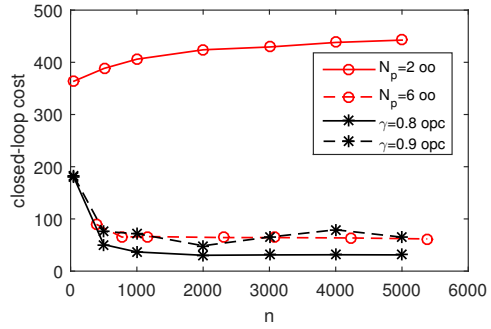

Fig. 3. Inverted pendulum results with OPC and OO for the worst and best parameters.

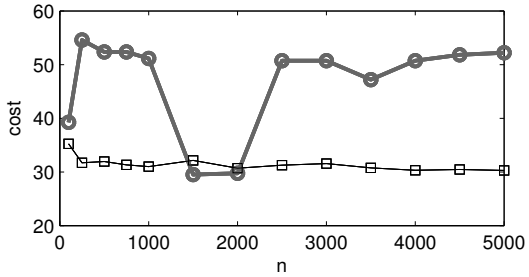

Fig. 4. Effects of box reuse: gray OPC, black OPC+.

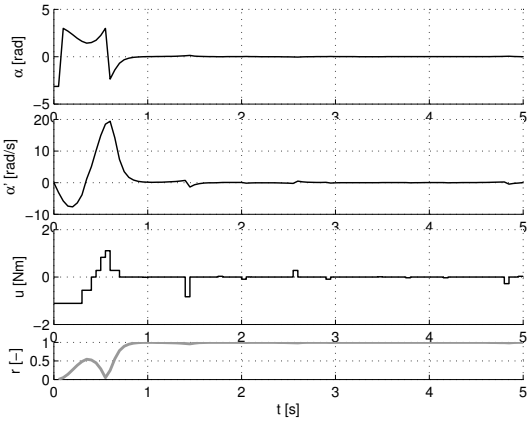

Fig. 5. OPC+ trajectory with $n = 2000$.

Next, we investigate the effect of reusing tail boxes, by comparing OPC with OPC+ in the same problem. We fix $\gamma = 0.9$ and run the algorithms for a similar range of budgets as above. As seen in Figure 4, in this case OPC+ has more reliable performance and is better than OPC for most budgets, except around $n = 1500$ where OPC finds some lucky solutions. The smaller and larger cost plateaus correspond, respectively, to using one or two swings to bring the pendulum up, see Figure 5 for an example with OPC+ where only one swing is used.

7. CONCLUSIONS

In this paper we tailored an adaptive-horizon planner called OPC to the receding-horizon optimal control of piecewise affine systems. We then introduced a modified version that provides guarantees on the closed-loop performance, by reusing data among different steps. This version is general and can be applied beyond PWA systems to other nonlinear dynamics. In an experimental study, OPC improved performance compared to the fixed-horizon approach of Xu et al. (2016) when the problem required long horizons; while OPC+ was even better.

The main open issue is to exploit the structure of the PWA problem to derive tighter near-optimality guarantees than in the general nonlinear case. In particular, we hope to identify large classes of PWA problems where the complexity, expressed by the branching factor κ, is small.

REFERENCES

Auer, P., Cesa-Bianchi, N., Fischer, P., 2002. Finite-time analysis of the multiarmed bandit problem. Machine Learning 47 (2-3), 235–256.

Baoti, M., Christophersen, F. J., Morari, M., 2006. Constrained optimal control of hybrid systems with a linear performance index. IEEE Transactions on Automatic Control 51 (12), 1903–1919.

Bemporad, A., Morari, M., 1999. Control of systems integrating logic, dynamics, and constraints. Automatica 35 (3), 407–427.

Bertsekas, D. P., 2012. Dynamic Programming and Optimal Control, 4th Edition. Vol. 2. Athena Scientific.

Borrelli, F., Baotic, M., Bemporad, A., Morari, M., 2005. Dynamic programming for constrained optimal control of discrete-time linear hybrid systems. Automatica 41 (10), 1709–1721.

Buşoniu, L., Daniels, A., Babuška, R., 2016a. Online learning for optimistic planning. Engineering Applications of Artificial Intelligence 55, 60–72.

Buşoniu, L., Páll, E., Munos, R., 6–8 July 2016b. Discounted near-optimal control of general continuous-action nonlinear systems using optimistic planning. In: Proceedings 2016 American Control Conference (ACC-16). Boston, US.

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

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

Munos, R., 2014. From bandits to Monte Carlo tree search: The optimistic principle applied to optimization and planning. Foundations and Trends in Machine Learning 7 (1), 1–130.

Sontag, E., 1981. Nonlinear regulation: The piecewise linear approach. IEEE Transactions on Automatic Control 26 (2), 346–358.

Storace, M., De Feo, O., 2004. Piecewise-linear approximation of nonlinear dynamical systems. IEEE Transactions on Circuits and Systems I: Regular Papers 51 (4), 830–842.

Sutton, R. S., Barto, A. G., 1998. Reinforcement Learning: An Introduction. MIT Press.

Xu, J., van den Boom, T., Buşoniu, L., De Schutter, B., 6–8 July 2016. Model predictive control for continuous piecewise affine systems using optimistic optimization. In: Proceedings 2016 American Control Conference (ACC-16). Boston, US.