Planning for optimal control and performance certification in nonlinear systems with controlled or uncontrolled switches ⋆

Lucian Busoniu [a], Jamal Daafouz [b], Marcos Cesar Bragagnolo [c], Irinel-Constantin Morărescu [b]

[a] *Automation Department, Technical University of Cluj-Napoca, Memorandumului 28, 400114 Cluj-Napoca, Romania*
[b] *Université de Lorraine, CRAN, UMR 7039 and CNRS, CRAN, UMR 7039, 2 av. Forêt de Haye, Vandœuvre-lès-Nancy, France*
[c] *Université de Reims Champagne-Ardenne, CReSTIC, BP 1039, 51687 Reims Cedex 2, France*

Abstract

We consider three problems for discrete-time switched systems with autonomous, general nonlinear modes. The first is optimal control of the switching rule so as to optimize the infinite-horizon discounted cost. The second and third problems occur when the switching rule is uncontrolled, and we seek either the worst-case cost when the rule is unknown, or respectively the expected cost when the rule is stochastic. We use optimistic planning (OP) algorithms that can solve general optimal control with discrete inputs such as switches. We extend the analysis of OP to provide certification (upper and lower) bounds on the optimal, worst-case, or expected costs, as well as to design switching sequences that achieve these bounds in the deterministic case. In this case, since a minimum dwell time between switching instants is often required, we introduce a new OP variant to handle this constraint, and analyze its convergence rate. We provide consistency and closed-loop performance guarantees for the sequences designed, and illustrate that the approach works well in simulations.

Key words: Switched systems; optimal control; planning; nonlinear systems.

1 Introduction

Switched systems consist of a set of linear or nonlinear dynamics called modes, together with a rule for switching between these modes [30]. They are employed to model real-world systems that are subject to known or unknown abrupt parameter changes such as faults [15,29], including for instance embedded systems in the automotive industry, aerospace, and energy management. This important class of hybrid systems is therefore heavily studied, with a main focus on stability and stabilization, see surveys [38,31] and papers [33,6,14,18,28]. Performance optimization for switched systems has also been investigated, see e.g. the survey [45] and [43,35,2,36,37,12]. Hybrid versions of the Pontryagin Maximum Principle or dynamic programming have been proposed [35,37], with the drawback of lacking efficient numerical algorithms. Suboptimal solutions with guaranteed performance include [41], [19]. The former efficiently represents the approximate value function using relaxations. The latter proves that the so-called min-switching strategies are consistent, i.e. that they improve performance with respect to non-switching strategies. Certification bounds [20] (lower and upper bounds on performance) are provided for linear switched systems with a dwell time assumption in [24]. In [12], the problem is treated by introducing modal occupation measures, which allow relaxation to a primal linear programming (LP) formulation. Overall, however, optimal control remains unsolved for general switched systems.

Motivated by this, our paper makes the following contributions. We propose an approach inspired from the field of planning in artificial intelligence, to either design switching sequences with near-optimal performance when switching is controllable, or to evaluate the performance when switching acts as a disturbance. We call the first problem PO, and the second either PW when the switching rule is unknown, in which case we estimate the worst-case performance; or PS when the switches

⋆ Corresponding author L. Buşoniu. This work was supported by the Agence Universitaire de la Francophonie (AUF) and the Romanian Institute for Atomic Physics (IFA) under the AUF-RO project NETASSIST; and by the PICS project No 6614 "Artificial-Intelligence-Based Optimization for the Control of Networked and Hybrid Systems". Additionally, the work of J. Daafouz and I.-C. Morărescu was partially funded by the National Research Agency (ANR) project "Computation Aware Control Systems" (No. ANR-13-BS03-004-02). Some early ideas related to the analysis in Section 5 were discussed with Remi Munos while developing the work [8], and we thank him for that.

Email addresses: lucian@busoniu.net (Lucian Busoniu), jamal.daafouz@univ-lorraine.fr (Jamal Daafouz), marcoscesarbragagnolo@gmail.com (Marcos Cesar Bragagnolo), constantin.morarescu@univ-lorraine.fr (Irinel-Constantin Morărescu).

evolve stochastically along a known Markov chain, in which case we evaluate the expected performance. Throughout, we consider a set of autonomous, general *nonlinear* modes, and a performance index consisting of the discounted infinite-horizon sum of general, non-quadratic stage costs. Optimistic planning [23,8,32] is used to search the space of possible switching sequences. In all cases, our approach guarantees certification, lower and upper bounds on the (expected) performance.

When it makes sense to do so, namely in PO and PW, the method also designs a switching sequence that achieves the certification bounds. Since a minimum dwell time δ between switching instants must often be ensured, we introduce a new optimistic planner called OPδ that handles this constraint, and analyze its convergence rate. The analysis provides consistency and closed-loop performance guarantees for the sequences designed. Different from typical results, consistency shows improvement with respect to any suboptimal sequences, not only stationary ones. Finally, we illustrate the practical performance of the approach in simulations for several linear examples and a nonlinear one.

Compared to the optimal control methods reviewed above, the advantages of our approach include: a characterization of the certification bounds, a procedure to design a worst-case sequence, a design method with minimum dwell time, improved consistency results, and the ability to handle very general nonlinear modes. While a high computational complexity is unavoidable due to this generality, our analysis is focused precisely on characterizing the relation between computation and quality of the bounds.

An important remark is that much of the literature focuses on stability [38,31], whereas our aim is to provide near-optimality guarantees. Stability is a separate, difficult problem for discounted costs [26,11,34]. Nevertheless, in some cases our approach can exploit existing stability conditions: e.g. for some types of linear modes stability may be guaranteed under a dwell time constraint using [18], in which case OPδ can enforce this constraint and thereby ensure stability.

The stochastic switching in PS leads to a Markov jump system, and there is a large body of literature dealing with such systems, again with a focus on linear modes [5,13], see e.g. [39] for optimal control. A recent nonlinear result is given in [44], where the stability properties of optimal mode inputs are analyzed for Markov jump systems with nonlinear controlled modes. The practical implementation in [44] works for unknown mode dynamics, but without error guarantees, whereas all our methods provide tightly characterized bounds.

In the context of existing planning methods, solving PO and PW without dwell-time is a straightforward application of optimistic planning [23]. In contrast, enforcing a minimum dwell-time requires deriving a novel algorithm and its accompanying analysis. Finally, solving PS can be seen as a special case of optimistic planning for stochastic systems [8], but the nature of this special case allows us to derive a streamlined analysis. Compared to the preliminary version of this work in [7], here we handle the new case of stochastic switching, provide consistency and closed-loop guarantees, and study two additional examples; in addition to including more technical discussion at several points in the paper.

Next, Section 2 formalizes the problem and Section 3 gives the necessary background. The approach is described in Section 4 for the optimal and worst-case problems PO and PW, and in Section 5 for the stochastic switching problem PS. Section 6 evaluates the planners in simulation examples of all these problems. Section 7 concludes.

List of symbols and notations

x, X, σ, S	state, state space, mode, set of modes
M	number of modes
f_σ, p	dynamics in mode σ, mode probabilities
$d, \boldsymbol{\sigma}_d$	depth, mode sequence of length/depth d
γ, g, G	discount factor, stage cost, cost bound
$J; \underline{J}, \overline{J}, \tilde{J}$	cost; optimal, worst-case, expected cost
$\rho, v, \tilde{v}$	reward function, value, expected value
r	reward value
n	computation budget
$\mathcal{T}, \mathcal{T}^*, \mathcal{L}(\mathcal{T})$	tree, near-optimal tree, leaves of $\mathcal{T}$
l, b	lower, upper bound on determ. value
L, B	lower, upper bound on expected value
l^*, b^*, L^*, B^*	best bounds found by the algorithms
d^*	largest depth found by the algorithms
ε	near-optimality or sub-optimality
κ	branching factor of near-optimal tree
K	complexity of dwell-time problem
β	complexity of stochastic problem
δ, Δ	minimum dwell time, dwell time
e, λ	leaf contribution, contribution cutoff
C, a, b, c	constants
$\underline{\cdot}, \overline{\cdot}$	quantity $\cdot$ in optimal, worst-case problem
$\cdot_\delta$	quantity $\cdot$ for minimum dwell-time δ
$O(\cdot), \Omega(\cdot)$	bounded above, below by $\cdot$ up to const.
$\tilde{O}(\cdot)$	bounded above by $\cdot$ up to log. terms
$[\cdot, \cdot]$	concatenation of two mode sequences

2 Problem statement

Consider a discrete-time nonlinear switched system with states $x \in X$. The system can be at each step k in one of M modes $\sigma \in S = \{\sigma^1, \dots, \sigma^M\}$, where each mode is autonomous:

$$x_{k+1} = f_{\sigma_k}(x_k) \tag{1}$$

The dwell time is defined as the number of steps during which the mode remains unchanged after a switch. A function $g(x_k, \sigma_k)$ assigns a numerical stage cost to each state-mode pair, e.g. quadratic in x_k up to saturation limits, see Example 1. Given a fixed initial state x_0, define an infinitely-long switching sequence $\boldsymbol{\sigma}_\infty = (\sigma_0, \sigma_1, \dots)$ and the infinite-horizon discounted cost of this sequence:

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

where $\gamma \in (0,1)$ is the discount factor and $x_{k+1} = f_{\sigma_k}(x_k)$. The dynamics f can be very general and a closed-form mathematical expression may not be available for them; the only requirement is that f can be simulated numerically.

To start with, we define two different problems:

PO. **Optimal control:** Find the optimal value $\underline{J} = \inf_{\boldsymbol{\sigma}_\infty} J(\boldsymbol{\sigma}_\infty)$ and a corresponding switching sequence that achieves it.

PW. **Worst-case switches**: Find the largest possible cost: $\overline{J} = \sup_{\boldsymbol{\sigma}_\infty} J(\boldsymbol{\sigma}_\infty)$, and a corresponding switching sequence that achieves it.

PO is useful when the switching rule can be controlled, while PW is interesting when switches are a disturbance and we are interested in the performance under the worst possible disturbance.

We will also consider a more refined case where the switches are known to evolve stochastically, following a Markov chain. In particular, the probability of moving from mode i to j is $P(\sigma_{k+1} = j \,|\, \sigma_k = i) = p(i,j)$, with $p \in [0,1]^{M \times M}$ known. The initial mode σ_0 is distributed with $p_0(\sigma_0)$, $p_0 \in [0,1]^M$ (if the initial mode is known, then p_0 can give it probability 1). Both p and p_0 must define valid probability distributions. In this case, we are interested in estimating the expected discounted cost.

PS. **Stochastic switches:** Find the expected discounted cost $\tilde{J} = \mathrm{E}_{\boldsymbol{\sigma}_\infty} \{J(\boldsymbol{\sigma}_\infty)\}$, over the possible switching sequences $\boldsymbol{\sigma}_\infty$ generated according to p_0, p.

In all three problems, we rely on a central assumption of cost boundedness.

Assumption 1 *The stage costs are bounded, so that* $g(x, \sigma) \in [0, G]$, $\forall x \in X, \sigma \in S$.

The main role of discounting and cost boundedness is to ensure that the infinite-horizon cost J in (2) is bounded to $[0, \frac{G}{1-\gamma}]$, which implies the same for the expected value $\tilde{J}$. Our planning algorithms rely on this boundedness property and would not be implementable without it. Note that many other works in control use discounting, e.g. [17,1,25]. Bounded costs are typical in AI methods for optimal control, such as the planning class that we use [27] and reinforcement learning [42]. A good way to achieve boundedness is by saturating a possibly unbounded original cost function, see Example 1. This changes the optimal solution (here, the sequence of switches) in ways that are nontrivial to analyze, but is often sufficient in practice. On the other hand, the physical limitations of the system may be meaningfully modeled by saturating the states and actions. In this case, a cost bound follows from the saturation limits.

Next, we impose a stability requirement.

Assumption 2 *For any sequence of switches $\boldsymbol{\sigma}_\infty$ that can occur, the system is stable from x_0 (the state trajectory is bounded).*

Regarding the qualifier "can occur", in Section 4.2 we will restrict the sequences so that a minimum dwell time is respected; in that case, only those sequences can occur and thus must lead to stability. Assumption 2 is natural in PW, since if the worst sequence destabilizes the system there is little point in investigating its cost. In the stochastic switched systems relevant to PS, more refined stability properties are usually assumed, such as almost sure stability [13]. Our Assumption 2 is stronger since it requires the system to be stable surely (in the probabilistic sense). The situation is more involved in PO, since our algorithms actually only examine near-optimal sequences, so strictly speaking the property is only needed for those sequences; however a formal analysis of this would require first a deep understanding of general stability properties with discounted cost, which are still in their infancy [34] and, as previously noted, outside the focus of this paper. Instead, we restrict ourselves in this paper to the rather strong Assumption 2, which allows us to focus on optimality. Note nevertheless that in some simple cases, like in the upcoming linear example, conditions to ensure stability exist.

Example 1 A classical switched system is obtained when the modes are linear and the cost is quadratic. Further, we saturate the quadratic cost to G to ensure Assumption 1:

$$f_{\sigma_k} = A_{\sigma_k} x_k$$
$$g(x_k, \sigma_k) = \min\{x^\top Q x, G\}$$

where Q is positive definite. For these dynamics, Theorem 1 in [18] provides a minimum dwell time which, if obeyed, guarantees stability for any switching sequence. We provide and analyze an algorithm that enforces a minimum dwell-time constraint in Section 4.2. □

3 Background: Optimistic planning for deterministic systems

This section introduces optimistic planning for deterministic systems (OP) [23,32], which forms the basis of our approach: it supplies independence of the mode dynamics, as well as a way to design sequences with known lower and upper bounds on the performance. Both PO and PW will be encompassed as variants of an optimal control problem that involves *maximizing a reward function* $\rho : X \times S \to [0, 1]$, where S is the discrete set of M actions. Given an initial state x_0, the value of a sequence is:

$$v(\boldsymbol{\sigma}_\infty) = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, \sigma_k) \tag{3}$$

and the optimal value is $v^* = \sup_{\boldsymbol{\sigma}_\infty} v(\boldsymbol{\sigma}_\infty)$. Under mild technical conditions, this optimum exists, together with a sequence that achieves it [3]. Denote $\rho(x_k, \sigma_k)$ by r_{k+1}, and a finite-length sequence of d actions by $\boldsymbol{\sigma}_d = (\sigma_0, \ldots, \sigma_{d-1})$.

At a high level, OP iteratively refines promising action sequences until a computational budget n, related to the number of evaluations of the model f, is exhausted. Based on the information accumulated about the values of these sequences, OP then chooses a sequence that is as good as possible.

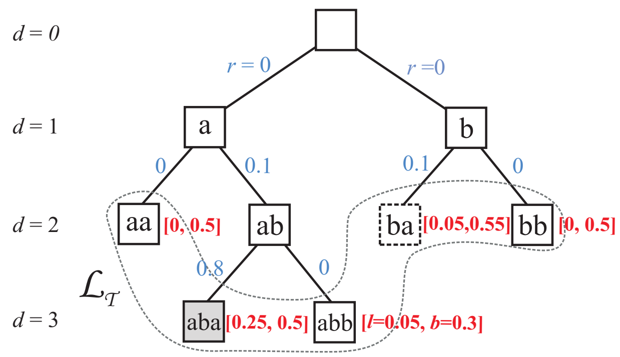

Fig. 1. Illustration of an OP tree $\mathcal{T}$. Nodes are labeled by action sequences, while arcs represent transitions and are labeled by the associated rewards, shown in blue. Near the nodes, lower bounds l and upper bounds b are shown in red boldface, see (4) for their definition. The leaves are enclosed in a dashed line. The tree is shown after 4 expansions, and $\gamma = 0.5$. (Figure best viewed in color.)

In more detail, the planning process can be visualized using a tree structure $\mathcal{T}$. Fig. 1 shows such a tree for a problem with two actions a and b (so, $M = 2$). Each node at some depth d is labeled by the corresponding action sequence $\boldsymbol{\sigma}_d$; for example, the gray node at $d = 3$ has sequence $\boldsymbol{\sigma}_3 = (\mathrm{a}, \mathrm{b}, \mathrm{a})$. Each node is also labeled by the state resulting from applying the sequence; state labels are not shown in the figure. Planning begins with a single root node labeled by the empty sequence and x_0, and proceeds by iteratively expanding nodes. The expansion of a node $\boldsymbol{\sigma}_d$, x_d consists of simulating all M actions from the associated state x_d, and adding for each j a child node labeled by the one-action-longer sequence $\boldsymbol{\sigma}_{d+1} = [\boldsymbol{\sigma}_d, \sigma^j]$ and by the state $f_{\sigma^j}(x_d)$, where $[\cdot, \cdot]$ denotes sequence concatenation. We will use nodes and sequences interchangeably. An arc between a parent and a child corresponds to a transition between the corresponding states, and is itself labeled by the reward associated with this transition. E.g. in Fig. 1, the arc leading to the gray node has reward 0.8. Unexpanded states in the tree $\mathcal{T}$ are called leaves, and the set of leaves is denoted $\mathcal{L}(\mathcal{T})$.

For any node/sequence $\boldsymbol{\sigma}_d$, because all the rewards at depths larger than d are in $[0, 1]$, we can define a lower bound $l(\boldsymbol{\sigma}_d)$ and an upper bound $b(\boldsymbol{\sigma}_d)$ on the values $v(\boldsymbol{\sigma}_\infty)$ of all infinite action sequences that share the initial subsequence up to $\boldsymbol{\sigma}_d$, as follows:

$$b(\boldsymbol{\sigma}_d) = \sum_{i=0}^{d-1} \gamma^i \rho(x_i, \sigma_i) + \frac{\gamma^d}{1-\gamma} =: l(\boldsymbol{\sigma}_d) + \frac{\gamma^d}{1-\gamma} \tag{4}$$

Here, $x_i, i = 0, \ldots, d-1$ is the state sequence obtained by applying $\boldsymbol{\sigma}_d$. The algorithm is *optimistic* because it expands at each iteration one most promising sequence, having the largest upper bound. After n node expansions, a *greedy*, "safe" sequence that maximizes l among the leaves is returned, together with bounds on the performance, see Algorithm 1.

Algorithm 1 Optimistic planning.

1: initialize tree $\mathcal{T} \leftarrow \{\boldsymbol{\sigma}_0\}$, the empty sequence
2: **for** $t = 1, \ldots, n$ **do**
3: find optimistic leaf: $\boldsymbol{\sigma}^\dagger \leftarrow \arg\max_{\boldsymbol{\sigma} \in \mathcal{L}(\mathcal{T})} b(\boldsymbol{\sigma})$
4: add to $\mathcal{T}$ the children of $\boldsymbol{\sigma}^\dagger$
5: **end for**
6: **return** sequence $\boldsymbol{\sigma}_d^* = \arg\max_{\boldsymbol{\sigma} \in \mathcal{L}(\mathcal{T})} l(\boldsymbol{\sigma})$, lower and upper bounds $l^* = l(\boldsymbol{\sigma}_d^*)$, $b^* = \max_{\boldsymbol{\sigma} \in \mathcal{L}(\mathcal{T})} b(\boldsymbol{\sigma})$

To exemplify, consider first the dashed node in Fig. 1. It has the upper bound $0 + \gamma \cdot 0.1 + \frac{\gamma^2}{1-\gamma} = 0.55$, which is maximal among all leaves, so this node is the optimistic one and will be expanded next. The gray node has the lower bound $0 + \gamma \cdot 0.1 + \gamma^2 \cdot 0.8 = 0.3$, again maximal, so this is the greedy node which would be returned if the algorithm were stopped. As a useful exercise, the reader may verify that the algorithm indeed obtains the tree of Fig. 1 after running for 4 iterations.

We will use the OP form described above to introduce our approach, but note that the actual implementation can be designed to avoid the explicit maximizations over the leaves. While OP is a type of nonlinear model-predictive control, its AI heritage (e.g. the A* graph search algorithm) leads to some atypical near-optimality guarantees, described next.

To analyze the complexity of finding the optimal sequence from x_0, define the near-optimal subtree:

$$\mathcal{T}^* = \{\boldsymbol{\sigma}_d \,|\, d \geq 0, v^* - v(\boldsymbol{\sigma}_d) \leq \frac{\gamma^d}{1-\gamma}\} \quad (5)$$

where the value $v(\boldsymbol{\sigma}_d) := \sup_{\boldsymbol{\sigma}_\infty} v([\boldsymbol{\sigma}_d, \boldsymbol{\sigma}_\infty])$ of a finite sequence $\boldsymbol{\sigma}_d$ is the best achievable after applying the actions in this sequence, by continuing optimally afterwards. A core property of OP is that it only expands nodes in $\mathcal{T}^*$. This subtree can be smaller than the complete tree containing all sequences, and to measure its size let $\mathcal{T}^*_d$ be the set of nodes at depth d on $\mathcal{T}^*$ and $|\cdot|$ denote set cardinality. Then, define the asymptotic branching factor as $\kappa = \limsup_{d\to\infty} |\mathcal{T}^*_d|^{1/d}$. This is a complexity measure for the problem, and intuitively represents an average number of children per node in the infinite subtree $\mathcal{T}^*$; see also below for its meaning in specific cases.

The upcoming theorem follows from the analysis in [23,32]. Parts (i), (ii) show that OP returns a long, near-optimal sequence with known performance bounds, and part (ii) quantifies the length and bounds via branching factor κ.

Theorem 3 *When OP is called with budget n:*[1]

(i) The optimal value v^*, as well as the value $v(\boldsymbol{\sigma}^*_d)$ of the sequence returned, are in the interval $[l^*, b^*]$. Further, the gap $\varepsilon := b^* - l^*$ satisfies $\varepsilon \leq \frac{\gamma^{d^*}}{1-\gamma}$ where d^* is the largest depth of any node expanded.

(ii) The length d of sequence $\boldsymbol{\sigma}^*_d$ is at least d^*.

(iii) If $\kappa > 1$, OP will reach a depth of $d^* = \Omega(\frac{\log n}{\log \kappa})$, and $\varepsilon = O(n^{-\frac{\log 1/\gamma}{\log \kappa}})$. If $\kappa = 1$, $d^* = \Omega(n)$ and $\varepsilon = O(\gamma^{cn})$, where c is a problem-dependent constant.

Note that d^* is the depth of the developed tree minus 1. The smaller κ, the better OP does. The best case is $\kappa = 1$, obtained e.g. when a single sequence is optimal and it always obtains rewards of 1, while all the other rewards on the tree are 0. In this case the algorithm only develops this sequence, and the gap decreases exponentially. In the worst case, $\kappa = M$, obtained e.g. when all the sequences have the same value, and the algorithm must explore the complete tree in a uniform fashion, expanding nodes in order of their depth.

[1] Let $g, h : (0, \infty) \to \mathbb{R}$. Statement $g(t) = O(h(t))$ (or $g(t) = \Omega(h(t))$) for large t means that $\exists t_0, c > 0$ so that $g(t) \leq ch(t)$ (or $g(t) \geq ch(t)$) $\forall t \geq t_0$. When the statement is made for small t, it means that $\exists t_0, c > 0$ so that the same inequalities hold for $\forall t \leq t_0$. Later on, we will also use notation $f(t) = \tilde{O}(g(t))$ for small (or large) t, which means that $\exists a > 0, b \geq 0, t_0 > 0$ so that $f(t) \leq a(\log g(t))^b g(t)$ $\forall t \leq t_0$ (or $\forall t \geq t_0$).

4 Solving the deterministic optimal-control and worst-case problems

In our first set of major results, we explain how PO and PW can be solved. We first explain how the OP algorithm can be applied off-the-shelf when there are no dwell time constraints. After that, a minimum dwell-time is considered, for which an extended algorithm with nontrivial analysis is necessary. These results were largely proven in our preliminary paper [7]; we include the proofs here too, so as to keep the paper self-contained. Then, we move on to fully novel contributions: consistency guarantees that show the OP solution is better than e.g. fixed-mode trajectories, and performance bounds in receding-horizon closed loop.

4.1 Applying OP to switched systems

OP can be applied to the system in Section 2 by interpreting the mode switches as discrete actions. To solve the optimal control problem PO and the worst-case problem PW, the reward function is taken, respectively, as:

$$\underline{\rho}(x, \sigma) := 1 - \frac{g(x, \sigma)}{G}, \quad \overline{\rho}(x, \sigma) = \frac{g(x, \sigma)}{G} \quad (6)$$

so that maximizing $\underline{\rho}$ is equivalent to minimizing costs g, and maximizing $\overline{\rho}$ to maximizing costs g. We use underline to denote quantities under PO and overline for PW, e.g. $\underline{\kappa}$ and $\overline{\kappa}$ are the complexity measures (branching factors) in the two problems. Then OP is simply applied with either of these two reward functions, and it will produce certification bounds and design a switching sequence that achieves them, as described next.

Corollary 4 *(i) When applied to PO, OP returns bounds $\underline{l}, \underline{b}$ so that the optimal value $\underline{J}$ is in the interval $[G(\frac{1}{1-\gamma} - \underline{b}), G(\frac{1}{1-\gamma} - \underline{l})]$, as well as a sequence $\underline{\boldsymbol{\sigma}}$ that achieves these bounds. The gap (interval size) is $G\underline{\varepsilon} = O(n^{-\frac{\log 1/\gamma}{\log \underline{\kappa}}})$ when $\underline{\kappa} > 1$, or $O(\gamma^{\underline{c}n})$ when $\underline{\kappa} = 1$.*

(ii) When applied to PW, OP returns bounds $\overline{l}, \overline{b}$ so that the worst-case value $\overline{J}$ is in the interval $[G\overline{l}, G\overline{b}]$, as well as a sequence $\overline{\boldsymbol{\sigma}}$ that achieves these bounds. The gap is $G\overline{\varepsilon} = O(n^{-\frac{\log 1/\gamma}{\log \overline{\kappa}}})$ when $\overline{\kappa} > 1$, or $O(\gamma^{\overline{c}n})$ when $\overline{\kappa} = 1$.

Proof: For any infinitely long sequence $\boldsymbol{\sigma}_\infty$, it is easily seen that the value under $\underline{\rho}$ is $\underline{v}(\boldsymbol{\sigma}_\infty) = \frac{1}{1-\gamma} - \frac{1}{G}J(\boldsymbol{\sigma}_\infty)$, and so $\underline{v}^* = \frac{1}{1-\gamma} - \frac{1}{G}\underline{J}$. Using this fact and Theorem 3, Part (i) is derived immediately. We similarly observe $\overline{v}(\boldsymbol{\sigma}_\infty) = \frac{1}{G}J(\boldsymbol{\sigma}_\infty)$ and derive Part (ii). ■

Here, $\underline{\varepsilon}$ and $\overline{\varepsilon}$ are the gaps attained by OP in Theorem 3, when applied with the costs rescaled to $[0, 1]$. So, the final gaps are G times larger, which is made explicit in the formulas.

4.2 Enforcing a dwell-time constraint

It is often important to ensure that after switching, the system remains in the same mode for a certain number of steps – the dwell time. This is because for some systems fundamental properties (stability, performance, etc.) can be guaranteed only under dwell time constraints, see e.g. Example 1 and [18]. Another reason is that in practice, it may be unsuitable or impossible to switch arbitrarily fast, so the designer must guarantee by construction a minimum dwell time. The dwell time may appear as a constraint fixed in advance or as a design parameter to be chosen.

Therefore, we introduce and analyze an algorithm that enforces a dwell time of at least δ along any switching sequence. Our starting point is OP, and most of the algorithm remains the same, including the lower and upper bounds, and the optimistic and greedy sequence selection rules. One important change is introduced in the node expansion procedure. Define a function $\Delta(\boldsymbol{\sigma})$, which takes as input any finite-length sequence $\boldsymbol{\sigma}$ and provides the last dwell time at the end of the sequence. Then, the dwell-time condition is checked for every node to be expanded. If the dwell time is at least δ, a switch can occur, and so children are created for all the actions (modes). Otherwise, a switch is not allowed, so only the child that keeps the mode constant is created. We call the algorithm OP with a dwell-time constraint (OPδ) and summarize it in Algorithm 2. By convention, it is assumed that the dwell time condition is satisfied at $d = 1$, see also the discussion in Section 4.4.

Algorithm 2 OP with a dwell-time constraint.

initialize tree $\mathcal{T} \leftarrow \{\boldsymbol{\sigma}_0\}$, the empty sequence
for $t = 1, \ldots, n$ **do**
  find optimistic leaf: $\boldsymbol{\sigma}^\dagger \leftarrow \arg\max_{\boldsymbol{\sigma} \in \mathcal{L}(\mathcal{T})} b(\boldsymbol{\sigma})$
  if $\Delta(\boldsymbol{\sigma}^\dagger) \geq \delta$ **then**
    create all children of $\boldsymbol{\sigma}^\dagger$
  else
    create one child, for the last action σ on $\boldsymbol{\sigma}^\dagger$
  end if
end for
return $\boldsymbol{\sigma}_d^* = \arg\max_{\boldsymbol{\sigma} \in \mathcal{L}(\mathcal{T})} l(\boldsymbol{\sigma})$, $l^* = l(\boldsymbol{\sigma}_d^*)$, $b^* = \max_{\boldsymbol{\sigma} \in \mathcal{L}(\mathcal{T})} b(\boldsymbol{\sigma})$

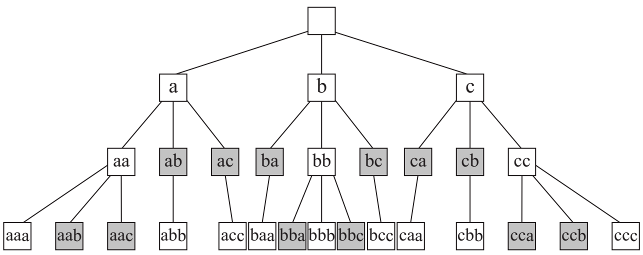

Fig. 2. Illustration of a constrained, OPδ tree for $\delta = 2$. Gray nodes have smaller dwell time than 2 – namely, 1.

As an example, Fig. 2 illustrates a complete tree down to depth 3, for $M = 3$ actions (a, b, c) and $\delta = 2$. The gray nodes have dwell time 1, as seen by a direct examination of their sequences; so they are allowed only one child, the one keeping the action unchanged. The children of the gray nodes have dwell time 2, so they are eligible for full expansion. Note that if δ were 3 instead, then these children would not satisfy the constraint either. Note that while the figure shows a uniform tree, the algorithm will usually only create some of the nodes on this tree, see the analysis.

We analyze in the sequel the bounds and gap provided by OPδ, in the general case of a reward function ρ. Then, by choosing the rewards as in (6), we will solve either PO or PW under the dwell-time constraint. Denote by $\boldsymbol{S}_\delta$ the set of sequences satisfying the constraint, and the constrained values:

$$v_\delta^* = \sup_{\boldsymbol{\sigma}_\infty \in \boldsymbol{S}_\delta} v(\boldsymbol{\sigma}_\infty)$$
$$v_\delta(\boldsymbol{\sigma}_d) = \sup_{\boldsymbol{\sigma}_\infty \text{s.t. } [\boldsymbol{\sigma}_d, \boldsymbol{\sigma}_\infty] \in \boldsymbol{S}_\delta} v([\boldsymbol{\sigma}_d, \boldsymbol{\sigma}_\infty])$$

Of course, the constrained optimum is generally worse than the unconstrained one, $v_\delta^* \leq v^*$, so enforcing the constraint comes at a price. Note that whenever the distinction between unconstrained and constrained values is not clear, we explicitly add the subscript δ to the quantity in the constrained problem.

As for OP, define the near-optimal constrained subtree:

$$\mathcal{T}_\delta^* = \{\boldsymbol{\sigma}_d \,|\, d \geq 0, \boldsymbol{\sigma}_d \in \boldsymbol{S}_\delta, v_\delta^* - v_\delta(\boldsymbol{\sigma}_d) \leq \frac{\gamma^d}{1-\gamma}\} \quad (7)$$

where $\boldsymbol{\sigma}_d \in \boldsymbol{S}_\delta$ means that there exists an infinite constrained sequence starting with $\boldsymbol{\sigma}_d$. Then, the following properties similar to OP also hold in the constrained case.

Lemma 5 *OPδ only expands nodes in $\mathcal{T}_\delta^*$, and the optimal constrained value v_δ^*, as well as the value $v_\delta(\boldsymbol{\sigma}_d^*)$ of the sequence returned, are in the interval $[l^*, b^*]$. Further, the gap $[l^*, b^*]$ satisfies $\varepsilon \leq \frac{\gamma^{d^*}}{1-\gamma}$ where d^* is the largest depth of any node expanded by OPδ.*

Proof: By definition of the algorithm all sequences expanded satisfy the first condition $\boldsymbol{\sigma}_d \in \boldsymbol{S}_\delta$. Furthermore, for any finite tree there exists some leaf sequence $\boldsymbol{\sigma}'$ so that $b(\boldsymbol{\sigma}') \geq v_\delta^*$, and since $\boldsymbol{\sigma}^\dagger$ maximizes the b-value, $b(\boldsymbol{\sigma}^\dagger) \geq v_\delta^*$, or equivalently $l(\boldsymbol{\sigma}^\dagger) + \frac{\gamma^d}{1-\gamma} \geq v_\delta^*$. This implies $v_\delta(\boldsymbol{\sigma}^\dagger) + \frac{\gamma^d}{1-\gamma} \geq v_\delta^*$, the same as the second condition in (7). So finally $\boldsymbol{\sigma}^\dagger \in \mathcal{T}_\delta^*$.

Clearly, $l^* \leq v_\delta(\boldsymbol{\sigma}_d^*) \leq v_\delta^*$ by definition. Consider now a leaf sequence $\boldsymbol{\sigma}'$ on the final tree, that is an initial subsequence of a constrained optimal sequence. Since b^* is the largest upper bound, $b^* \geq b(\boldsymbol{\sigma}') \geq v^*_\delta$, so combined with the first inequality we get $v^*_\delta, v_\delta(\boldsymbol{\sigma}^*_d) \in [l^*, b^*]$. Further, by expanding nodes the largest b-value on the tree can only decrease. Hence, for any node $\boldsymbol{\sigma}^\dagger$ previously expanded, found at some depth d, we have $b^* \leq b(\boldsymbol{\sigma}^\dagger)$ and also $l^* \geq l(\boldsymbol{\sigma}^\dagger)$, so $\varepsilon = b^* - l^* \leq b(\boldsymbol{\sigma}^\dagger) - l(\boldsymbol{\sigma}^\dagger) = \frac{\gamma^d}{1-\gamma}$. At least one such node is at d^*, so $\varepsilon \leq \frac{\gamma^{d^*}}{1-\gamma}$. Note that this proof is largely the same as for original OP, see e.g. [7], except that the dwell-time constrained values are substituted for the unconstrained ones. ■

So far the analysis simply established that OPδ preserves some interesting properties of OP. The main novelty of the constrained algorithm follows: the behavior of the new gap ε obtained. To this end, the cardinality of the near-optimal tree must be characterized using a new complexity measure, which is defined as follows.

Definition 6 *The complexity measure is the smallest value of K for which there exists a constant $C > 0$ so that $\left|\mathcal{T}^*_{d,\delta}\right| \leq C \cdot K^{d/\delta}$, $\forall d \geq 0$.*

Here $\mathcal{T}^*_{d,\delta}$ denotes the nodes of $\mathcal{T}^*_\delta$ at depth d. Note that due to the special cases analyzed below, a K always exists and belongs to the interval $[1, M\delta]$ (it may be noninteger). Constant K plays a similar role to the branching factor κ in the unconstrained problem, and in some cases a relationship between the two quantities can be found, as we show below. Our results hold for any pair C, K, but we take the smallest K. Using K, the gap ε is characterized as follows.

Theorem 7 *Given a computational budget n, the OPδ algorithm produces a gap $\varepsilon = O(n^{-\delta\frac{\log 1/\gamma}{\log K}})$ if $K > 1$, and $\varepsilon = O(\gamma^{\frac{n}{C}})$ when $K = 1$, where C is the constant from the definition of K.*

Proof: Define d_n to be the smallest depth so that $n \leq \sum_{i=0}^{d_n} \left|\mathcal{T}^*_{i,\delta}\right|$; this means OP$\delta$ has expanded nodes at d_n (perhaps not yet at $d_n + 1$), so $d^* \geq d_n$ and $\varepsilon \leq \frac{\gamma^{d_n}}{1-\gamma}$.

If $K > 1$, then[2] $n \leq \sum_{i=0}^{d_n} CK^{i/\delta} = C\frac{(K^{1/\delta})^{d_n+1}-1}{K^{1/\delta}-1} \leq c_1 K^{d_n/\delta}$, from which $d_n \geq \delta\frac{(\log n - \log c_1)}{\log K} \geq \delta \log n / \log K - c_2$. Thus, after some manipulations $\varepsilon \leq c_3 n^{-\delta\frac{\log 1/\gamma}{\log K}}$.

If $K = 1$, then $n \leq \sum_{i=0}^{d_n} C \leq C(d_n + 1)$, and $d_n \geq \frac{n-1}{C}$ leading to $\varepsilon \leq \gamma^{\frac{n-1}{C}}$. The theorem is proven. ■

While we measure complexity by the number n of nodes expanded, the number of children of a node may be either 1 or M, so the computational cost of expansion varies. Nevertheless, this only amounts to a constant factor in the relationships, and so it does not affect the asymptotic analysis. Next, we find the complexity measure K and illustrate its relation to κ in two interesting cases.

Case 1: All sequences optimal Consider a problem where all the rewards are identical, say equal to 1 or to 0. While any sequence is optimal in this problem, it is nevertheless an interesting case that highlights the (correct) behavior of the algorithm in general. In this case the algorithm must explore the entire tree uniformly, in the order of depth, so to find K we must count all the nodes at a given depth. Define the vector N_d of length δ, so that $N_{d,i}$ for $i < \delta$ counts the nodes $\boldsymbol{\sigma}_d$ with dwell time $\Delta(\boldsymbol{\sigma}_d) = i$. The last element is different, it counts all the nodes with dwell time *at least* δ, since they all behave exactly the same in the algorithm. Looking e.g. at Fig. 2, $N_3 = [6, 9]$ since the 6 gray nodes have dwell time one, and 9 have dwell time at least two (3 of these have dwell time three).

Each node with dwell time at least δ produces 1 child like itself, and $M - 1$ children of dwell time 1; and each node of dwell time $i < \delta$ produces 1 child of dwell time $i+1$. Writing this explicitly, we have $N_{d+1} = [N_{d,\delta}(M-1), N_{d,1}, ..., N_{d,\delta-2}, N_{d,\delta-1} + N_{d,\delta}]$. Using this, we will prove by induction that:

$$\begin{aligned} N_d \leq [&\delta^{j-1}M^j(M-1), \delta^{j-1}M^j(M-1), \ldots, \\ &\delta^{j-1}M^j(M-1), \delta^{j-1}M^j] \end{aligned} \tag{8}$$

where $j = \lceil d/\delta \rceil$ and (here and in the sequel) vector inequalities hold elementwise. By directly computing all counters for $d \leq \delta$, we see that relation (8) holds for $j = 0, 1$. E.g., in particular, $N_\delta = [M(M-1), M(M-1), \ldots, M(M-1), M]$. Then, assuming the relation holds at $d = j\delta$, we have:

$$\begin{aligned} N_{j\delta+1} \leq [&\delta^{j-1}M^j(M-1), \delta^{j-1}M^j(M-1), \ldots, \\ &\delta^{j-1}M^j(M-1), \delta^{j-1}M^{j+1}] \\ N_{j\delta+2} \leq [&\delta^{j-1}M^{j+1}(M-1), \delta^{j-1}M^j(M-1), \ldots, \\ &\delta^{j-1}M^j(M-1), 2\delta^{j-1}M^{j+1} - \delta^{j-1}M^j] \\ N_{j\delta+3} \leq [&2\delta^{j-1}M^{j+1}(M-1), \delta^{j-1}M^{j+1}(M-1), \ldots, \\ &\delta^{j-1}M^j(M-1), 3\delta^{j-1}M^{j+1} - 2\delta^{j-1}M^j] \\ &\ldots \\ N_{j\delta+\delta} \leq [&(\delta-1)\delta^{j-1}M^{j+1}(M-1), \\ &(\delta-2)\delta^{j-1}M^{j+1}(M-1), \ldots, \\ &\delta^{j-1}M^{j+1}(M-1), \\ &\delta\delta^{j-1}M^{j+1} - (\delta-1)\delta^{j-1}M^j] \end{aligned}$$

Clearly, all these vectors are smaller than $[\delta^j M^{j+1}(M-1), \delta^j M^{j+1}(M-1), \ldots, \delta^j M^{j+1}(M-1), \delta^j M^{j+1}]$ so the

[2] We denote by c_i positive constants whose value is unimportant to the asymptotic analysis.

induction is finished. Then, finally:

$$\left|\mathcal{T}^*_{d,\delta}\right| \leq \sum_{i=1}^{\delta} N_{d,i} \leq \delta^j M^{j+1} \leq M(\delta M)^{\lceil \frac{d}{\delta} \rceil} \leq M^2 \delta (\delta M)^{\frac{d}{\delta}}$$

so $K = M\delta$. Since in this problem $\mathcal{T}^*_{d,\delta}$ has the largest possible size, $M\delta$ is also the upper limit of the possible values of K.

Comparing to OP, K equals δ times the branching factor M that OP would have in a uniform tree without the constraint. The two algorithms explore trees that grow exponentially with the depth, but have different size. Consider the resulting rates: $\varepsilon = O(n^{-\frac{\log 1/\gamma}{\log M}})$ for OP, and $\varepsilon = O(n^{-\delta \frac{\log 1/\gamma}{\log M\delta}})$ for OPδ. Since $\delta \frac{\log 1/\gamma}{\log M\delta} \geq \frac{\log 1/\gamma}{\log M}$ for all $M, \delta \geq 2$, OPδ converges faster in this worst-case sense. This does not mean that OPδ is faster for any given particular problem, and in fact the relationship varies, see also Case 2 next. $\square$

Case 2: One optimal sequence In this case, a single sequence has maximal rewards (equal to 1), and all other transitions have a reward of 0. Here, two situations are possible. If the optimal sequence is within the constrained set, OPδ always expands this sequence further, we have $\left|\mathcal{T}^*_{d,\delta}\right| = 1$ and $K = 1$, the easiest type of problem. This also leads to the lower limit of 1 for K. In this situation, the original OP explores the same path so $\kappa = 1$. Thus the best-case convergence rate of the two algorithms is similar – exponential.

Otherwise, the optimal sequence leaves the constrained set at a node $\boldsymbol{\sigma}_0$ at some finite depth, and then the algorithm must explore uniformly the subtree having $\boldsymbol{\sigma}_0$ at the root (perhaps in addition to some other nodes). Then, since the analysis is asymptotic, for large depths, K has the maximal value of $M\delta$ again. Since OP is still allowed to refine the optimal sequence, $\kappa = 1$ and here introducing the constraint has made the problem significantly *more* difficult. $\square$

Having completed the analysis of generic OPδ, its properties in the context of PO and PW for switched systems are summarized in the following direct adaptation of Corollary 4. The differences are that the values become constrained, and the convergence rates change to those of OPδ.

Corollary 8 *(i) When applied to PO, OPδ returns bounds $\underline{l}, \underline{b}$ so that the optimal value $\underline{J}_\delta := \inf_{\boldsymbol{\sigma}_\infty \in \boldsymbol{S}_\delta} J(\boldsymbol{\sigma}_\infty)$ is in the interval $[G(\frac{1}{1-\gamma} - \underline{b}), G(\frac{1}{1-\gamma} - \underline{l})]$, as well as a sequence $\underline{\boldsymbol{\sigma}}$ that achieves these bounds. The gap is $G\underline{\varepsilon} = O(n^{-\delta \frac{\log 1/\gamma}{\log \underline{K}}})$ when $\underline{K} > 1$, or $O(\gamma^{n/\underline{C}})$ when $\underline{K} = 1$.*

(ii) In PW, OPδ returns bounds $\bar{l}, \bar{b}$ so that the worst-case value $\overline{J}_\delta := \sup_{\boldsymbol{\sigma}_\infty \in \boldsymbol{S}_\delta} J(\boldsymbol{\sigma}_\infty)$ is in the interval $[G\bar{l}, G\bar{b}]$, as well as a sequence $\overline{\boldsymbol{\sigma}}$ that achieves these bounds. The gap is $G\overline{\varepsilon} = O(n^{-\delta \frac{\log 1/\gamma}{\log \overline{K}}})$ when $\overline{K} > 1$, or $O(\gamma^{n/\overline{C}})$ when $\overline{K} = 1$.

So far we have covered the results in [7], providing additional technical insight. The next contributions are fully novel, and deal with the consistency and closed-loop performance of switching sequences returned by OP and OPδ. While answering these questions is likely more useful in PO, we give the analysis in a general form that is also applicable to PW, where a "better" solution means one that is closer to the worst-case performance.

4.3 Consistency guarantees

An important question in switched systems is whether the sequence found guarantees an improvement over some particular type of suboptimal solutions. Often, the trivial sequences that keep the mode constant are considered. This property is called consistency [19], and next we guarantee two versions of it, where we compare the (still suboptimal) solution found by OP with alternative suboptimal solutions. The first version shows improvement over *finitely long* sequences of (nearly) the same length as that returned by OP, while the second property proves that for any *infinitely long* sequence that is strictly suboptimal, the algorithm will find a better sequence given a sufficiently large budget n. Importantly, these guarantees require no particular structure on the sequences, so they hold not only for constant-mode suboptimal sequences, but also periodic ones, etc.

Theorem 9 *Let $\boldsymbol{\sigma}^*_d$ be the sequence returned by OP. Note that by Theorem 3, $d \in \{d^*, d^*+1\}$. Then:*

(i) For any sequence $\boldsymbol{\sigma}'_{d-1}$ of depth $d-1$, we have $l(\boldsymbol{\sigma}^*_d) \geq l(\boldsymbol{\sigma}'_{d-1})$.
(ii) Take an $\varepsilon_{\mathrm{ct}} > 0$ and consider any sequence $\boldsymbol{\sigma}_\infty$ that is strictly suboptimal with a suboptimality of at least $\varepsilon_{\mathrm{ct}}$, i.e. $v^* - v(\boldsymbol{\sigma}_\infty) \geq \varepsilon_{\mathrm{ct}}$. Then, for sufficiently large budget n the sequence returned will satisfy $v(\boldsymbol{\sigma}^*_d) \geq v(\boldsymbol{\sigma}_\infty)$.

Proof: For part (i), take an arbitrary sequence $\boldsymbol{\sigma}'_{d-1}$. If $\boldsymbol{\sigma}'_{d-1}$ corresponds to a node that was created by the algorithm, then the relation holds by definition, since the sequence returned has the largest lower bound on the created tree (including inner nodes, by the definition of l). Otherwise, there exists some ascendent sequence $\boldsymbol{\sigma}'_{d'}$ of $\boldsymbol{\sigma}'_{d-1}$, for $d' < d-1$, over which the parent sequence $\boldsymbol{\sigma}^*_{d-1}$ of $\boldsymbol{\sigma}^*_d$ was preferred for expansion, see Fig. 3. This means:

$$b(\boldsymbol{\sigma}^*_{d-1}) \geq b(\boldsymbol{\sigma}'_{d'}) \geq b(\boldsymbol{\sigma}'_{d-1})$$

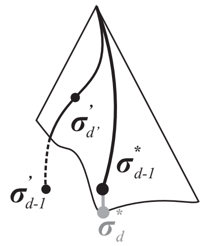

Fig. 3. Sequences from the proof of Theorem (i). Here, $\boldsymbol{\sigma}'_{d-1}$ is not on the tree, and $\boldsymbol{\sigma}'_{d'}$ may not be a leaf on the final tree.

because b-values decrease monotonically along every path. Equivalently:

$$l(\boldsymbol{\sigma}^*_{d-1}) + \frac{\gamma^{d-1}}{1-\gamma} \geq l(\boldsymbol{\sigma}'_{d-1}) + \frac{\gamma^{d-1}}{1-\gamma}$$

so finally $l(\boldsymbol{\sigma}^*_d) \geq l(\boldsymbol{\sigma}^*_{d-1}) \geq l(\boldsymbol{\sigma}'_{d-1})$, and part (i) is proven.

For part (ii), simply take a budget n that ensures a sufficiently large d^* so that $\frac{\gamma^{d^*}}{1-\gamma} < \varepsilon_{\mathrm{ct}}$. Then, $v^* - v(\boldsymbol{\sigma}^*_d) < \varepsilon_{\mathrm{ct}}$, see Theorem 3(i), which combined with the definition of $\varepsilon_{\mathrm{ct}}$ implies the desired result. ■

A similar guarantee holds for OPδ, restricted to the set of sequences that satisfy the dwell-time constraint. The proof will be skipped, since it consists simply in substituting the constrained values and sequences in the proof of Theorem 9.

Proposition 10 *For the sequence $\boldsymbol{\sigma}^*_d$ returned by OPδ:*

(i) Given any constrained sequence $\boldsymbol{\sigma}'_{d-1} \in \boldsymbol{S}_\delta$ of depth $d-1$, we have $l(\boldsymbol{\sigma}^*_d) \geq l(\boldsymbol{\sigma}'_{d-1})$.
(ii) Take an $\varepsilon_{\mathrm{ct}} > 0$ and consider any feasible sequence $\boldsymbol{\sigma}_\infty \in \mathbf{S}_\delta$ that is strictly suboptimal with a suboptimality of at least $\varepsilon_{\mathrm{ct}}$ with respect to v^*_δ, i.e. $v^*_\delta - v(\boldsymbol{\sigma}_\infty) \geq \varepsilon_{\mathrm{ct}}$. Then for sufficiently large budget n, $v_\delta(\boldsymbol{\sigma}^*_d) \geq v(\boldsymbol{\sigma}_\infty)$.

4.4 Closed-loop, receding-horizon application

The results above are for a single sequence starting at x_0. In practice (and in our examples below), the algorithms are used in receding horizon, by only applying the first action σ_0 of the sequence, then recomputing a new sequence from x_1 and applying its first action σ_1, etc. Of course, the complexity measure κ or K may be different at each encountered state. Importantly, for OPδ, if a switch occurs at step k, then to guarantee the dwell-time constraint the mode must be kept constant, keeping the loop open, until $k+\delta-1$, and OPδ only needs to be called again at step $k+\delta$.

When OPδ is applied in this way at $k \geq 1$, some of the nodes at $d = 1$ have dwell-time 1 so they become constrained. This is unlike the case in Fig. 2 where they are unconstrained. This restriction is easy to take into account in the implementation. Regarding the convergence rate analysis, the restriction will change which nodes are expanded at steps $k \geq 1$, so the complexity measure K computed without constraining the nodes at depth 1 may be different from the true value. However, the full range of values of K is still possible even for this constrained tree, e.g. because the subtree of the single unconstrained node at depth 1 can have any structure from those described in the special cases. So, adapting the analysis for steps $k \geq 1$ would not be very informative.

The following result shows that applying OP or OPδ in receding horizon can never lead to worse performance than that of the first sequence they return at x_0.

Proposition 11 *For either OP or OPδ, consider the first sequence $\boldsymbol{\sigma}_{d_0,0}$ returned by the algorithm at $k=0$, and the closed-loop sequence $\boldsymbol{\sigma}_{\infty,\mathrm{cl}}$ that it applies when used in receding horizon. Then, $v(\boldsymbol{\sigma}_{\infty,\mathrm{cl}}) \geq l(\boldsymbol{\sigma}^*_{d_0,0})$.*

Proof: Consider first OPδ at two steps k, j where it is consecutively applied. Define the sequence $\boldsymbol{\sigma}_{d_k,k}$ computed at k, and $\boldsymbol{\sigma}_{d_j,j}$ at j. In-between, the initial subsequence $\boldsymbol{\sigma}_{j-k,k}$ of length $j-k$ is applied; this length may be 1 or δ depending on whether a switch has occurred. Consider also the trees $\mathcal{T}_k$, $\mathcal{T}_j$ developed, see Fig. 4. It is essential to note that by the definition of the algorithm, it expands nodes in the same order in $\mathcal{T}_j$ as it did *in the subtree $\mathcal{T}_k(\boldsymbol{\sigma}_{j-k,k})$ of $\mathcal{T}_k$ with its root at $\boldsymbol{\sigma}_{j-k,k}$*. This is because, firstly, the constraint is enforced in closed loop so no new nodes become eligible for expansion at j with respect to k. Secondly, the b-values in $\mathcal{T}_j$ are an affine transformation of those in $\mathcal{T}_k$, so the nodes maximizing the b-value are the same.

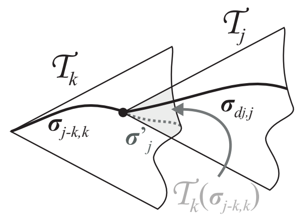

Fig. 4. Sequences and trees from the proof of Proposition 11.

Since OPδ has the same budget n when called at j, clearly $\mathcal{T}_k(\boldsymbol{\sigma}_{j-k,k}) \subset \mathcal{T}_j$, which means $l_j(\boldsymbol{\sigma}'_j) \leq l_j(\boldsymbol{\sigma}_{d_j,j}), \forall \boldsymbol{\sigma}'_j \in \mathcal{T}_k(\boldsymbol{\sigma}_{j-k,k})$. Subscripts were introduced in the lower bounds since they may differ at different steps even if they are computed for the same sequence, due to the fact that the root state is different and so the same actions can lead to different rewards. Since $\boldsymbol{\sigma}_{d_k,k} = [\boldsymbol{\sigma}_{j-k,k}, \boldsymbol{\sigma}'_j]$ for some $\boldsymbol{\sigma}'_j$, we have:

$$\begin{aligned} l_k(\boldsymbol{\sigma}_{d_k,k}) &= l_k(\boldsymbol{\sigma}_{j-k,k}) + \gamma^{j-k} l_j(\boldsymbol{\sigma}'_j) \\ &\leq l_k(\boldsymbol{\sigma}_{j-k,k}) + \gamma^{j-k} l_j(\boldsymbol{\sigma}_{d_j,j}) = l_k([\boldsymbol{\sigma}_{j-k,k}, \boldsymbol{\sigma}_{d_j,j}]) \end{aligned}$$

Thus, closing the loop after some number of actions and reapplying OPδ leads to an overall better value than just applying the first sequence. This is true at any step, so by applying it recursively, first for $\boldsymbol{\sigma}_{d_0,0}$ and $\boldsymbol{\sigma}_{d_1,1}$, then for the next pair of sequences, etc. we obtain the desired result.

In OP, the only changes are that $j = k + 1$ at any step, and there are no constraints on the sequences. With these changes, the proof becomes a special case of the argument above, so we are done. Note that this argument for OP (but not for OPδ) already appeared in the proof of Theorem 3 of [10]. ∎

5 Solving the stochastic-switches problem

Finally, we consider PS and propose a tree search algorithm to approximate the expected discounted cost. We introduce an appropriate complexity measure for this problem, and provide a bound on the approximation accuracy of the algorithm, which depends on the computation budget and on the complexity measure.

A similar tree structure to Fig. 1 will be used. In contrast to the deterministic case, the arcs are now also labeled by probabilities: the arc from the root to node i at depth 1 is labeled by $p_0(i)$, while an arc between modes i and j at greater depths is labeled by $p(i, j)$. A node at depth d will be associated as before to its sequence $\boldsymbol{\sigma}_d$, but now also to the probability of this sequence, equal to the product of probabilities along the path to the node:

$$P(\boldsymbol{\sigma}_d) = p_0(\sigma_0) \prod_{k=0}^{d-2} p(\sigma_k, \sigma_{k+1}) \tag{9}$$

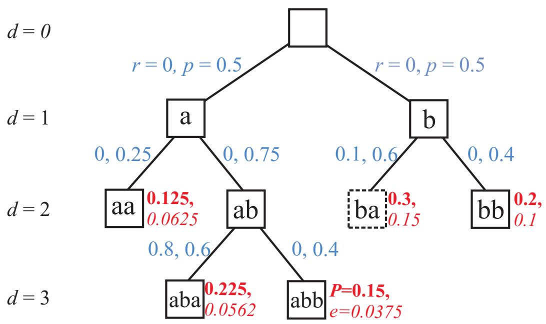

Fig. 5. Illustration of a stochastic tree. Transition probabilities are shown on arcs in blue, after the rewards. Near each leaf node, probabilities P are shown in red boldface, and contributions e in red italic. The discount factor is $\gamma = 0.5$. (Figure best viewed in color.)

Fig. 5 exemplifies using the same tree as in Section 3, but this time with stochastic mode transitions. In particular, $p_0(\mathrm{a}) = p_0(\mathrm{b}) = 0.5$, and $p(\mathrm{a}, \mathrm{a}) = 0.25, p(\mathrm{a}, \mathrm{b}) = 0.75, p(\mathrm{b}, \mathrm{a}) = 0.6, p(\mathrm{b}, \mathrm{b}) = 0.4$. Thus, the dashed node has probability $P((\mathrm{b}, \mathrm{a})) = p_0(\mathrm{b})p(\mathrm{b}, \mathrm{a}) = 0.5 \cdot 0.6 = 0.3$.

To find the expected cost, take the reward function $\bar{\rho}(x, \sigma) = \frac{g(x,\sigma)}{G}$. Define the expected value $\tilde{v} = \mathrm{E}_{\boldsymbol{\sigma}_\infty} \{v(\boldsymbol{\sigma}_\infty)\}$, and recall the definition (4) of the sequence bounds l and b. Using these, for any tree $\mathcal{T}$, the following quantities define lower and upper bounds on $\tilde{v}$:

$$\begin{aligned} L(\mathcal{T}) &:= \sum_{\boldsymbol{\sigma} \in \mathcal{L}(\mathcal{T})} P(\boldsymbol{\sigma})\, l(\boldsymbol{\sigma}) \\ B(\mathcal{T}) &:= \sum_{\boldsymbol{\sigma} \in \mathcal{L}(\mathcal{T})} P(\boldsymbol{\sigma})\, b(\boldsymbol{\sigma}) \\ &= L(\mathcal{T}) + \sum_{\boldsymbol{\sigma} \in \mathcal{L}(\mathcal{T})} P(\boldsymbol{\sigma}) \frac{\gamma^{d(\boldsymbol{\sigma})}}{1-\gamma} =: L(\mathcal{T}) + \varepsilon(\mathcal{T}) \end{aligned} \tag{10}$$

The depth of leaf sequence $\boldsymbol{\sigma}$ was denoted $d(\boldsymbol{\sigma})$ to highlight the fact that it varies among the leaves. Notation $\varepsilon(\mathcal{T})$ is the gap between the two bounds. The contribution of a leaf to this gap is defined as $e(\boldsymbol{\sigma}) := P(\boldsymbol{\sigma}) \frac{\gamma^{d(\boldsymbol{\sigma})}}{1-\gamma}$. At this point it becomes clear that a good algorithm should expand nodes in decreasing order of their contribution, so as to maximally reduce the uncertainty on the expected value. For example, in Fig. 5 the contribution of the dashed node (b, a) is $0.3 \cdot \frac{\gamma^2}{1-\gamma} = 0.15$, the largest among the leaves, so this node should be expanded next. Furthermore, by using the individual sequence bounds already computed in Fig. 1, we find $L(\mathcal{T}) = 0.0788$, $B(\mathcal{T}) = 0.4850$.

Algorithm 3 summarizes the procedure. The algorithm returns lower and upper bounds L^*, B^* but does not design a sequence, since this does not make sense in PS – many sequences may in fact occur. Note that because of reward scaling, the true expected cost $\tilde{J}$ is in the larger interval $[GL^*, GB^*]$.

Algorithm 3 Evaluation of stochastic switches.

1: initialize tree $\mathcal{T} \leftarrow \{\boldsymbol{\sigma}_0\}$, the empty sequence
2: **for** $t = 1, \ldots, n$ **do**
3: max-contrib. leaf: $\boldsymbol{\sigma}^\dagger \leftarrow \arg\max_{\boldsymbol{\sigma} \in \mathcal{L}(\mathcal{T})} e(\boldsymbol{\sigma})$
4: create all children of $\boldsymbol{\sigma}^\dagger$, labeled by $1, \ldots, M$
5: **end for**
6: **return** bounds $L^* = L(\mathcal{T})$, $B^* = B(\mathcal{T})$

This algorithm is a special case of optimistic planning for Markov decision processes, from [8], where discrete controlled decisions were allowed in addition to the stochastic transitions. The simpler case of PS, without controlled decisions, allows us to derive in the sequel a more direct analysis than in [8]. Denoting $\varepsilon^* = B^* - L^*$, we are interested in the evolution of ε^* with the budget n. We start by introducing a measure of the problem complexity. Let $\mathcal{T}_\infty$ denote the infinitely deep tree obtained by continuing with all possible sequences indefinitely.

Definition 12 *Define the subtree of sequences with contributions larger than* λ*:* $\mathcal{T}_\lambda = \{\boldsymbol{\sigma} \in \mathcal{T}_\infty \,|\, e(\boldsymbol{\sigma}) \geq \lambda\}$.

Then, the complexity measure is the smallest value of β for which there exist constants $a > 0, b \geq 0$ so that $|\mathcal{T}_\lambda| \leq a[\log(1/\lambda)]^b \lambda^{-\beta}, \ \forall \lambda > 0$.

The set $\mathcal{T}_\lambda$ is always a subtree at the top of $\mathcal{T}_\infty$, because the contributions monotonically decrease with increasing depth. Recalling footnote 1, we say $|\mathcal{T}_\lambda| = \tilde{O}(\lambda^{-\beta})$.

Theorem 13 *Given a budget of n expansions, when $\beta > 0$ the gap satisfies $\varepsilon^* = \tilde{O}(n^{-\frac{1-\beta}{\beta}})$. When $\beta = 0$, then $\varepsilon^* = \tilde{O}(\gamma^{c' n^{1/b}})$ for a problem-dependent constant $c' > 0$.*

Proof: Denote $n(\lambda) = a[\log(1/\lambda)]^b \lambda^{-\beta}$. When interpreted as a function of λ, $|\mathcal{T}_\lambda|$ is piecewise constant: it remains unchanged as long as λ does not equal the contribution of any node on the tree, and then jumps to a larger value when λ becomes equal to the contribution of some node(s). Consider now two consecutive values $\lambda_1 > \lambda_2$ at such discontinuities, taken so that $n(\lambda_1) \leq n < n(\lambda_2)$. Since nodes are expanded in order of their contribution and $|\mathcal{T}_{\lambda_1}| \leq n(\lambda_1)$, all nodes in $\mathcal{T}_{\lambda_1}$ have been expanded. Further, the decrease in contribution from a parent to its *largest*-contribution child is at most by a factor $\frac{\gamma}{M}$, when the probabilities are uniform (otherwise, a larger-contribution child can be found). This implies that the sequence of λ values at the discontinuities decreases at most with the same rate $\frac{\gamma}{M}$, so $\lambda_2 \geq \frac{\gamma}{M}\lambda_1$, or equivalently $\lambda_1 \leq \frac{M}{\gamma}\lambda_2$.

Next, the two cases for β are handled separately. When $\beta > 0$, solving $n < n(\lambda_2)$ we get $\lambda_2 \leq a_2[\log n]^{b_2} n^{-1/\delta}$ for positive constants a_2, b_2. Hence:

$$\varepsilon^* \leq |\mathcal{T}_{\lambda_1}|\, \lambda_1 \leq n\frac{M}{\gamma}\lambda_2 \leq a_2 \frac{M}{\gamma}[\log n]^{b_2} n^{1-\frac{1}{\beta}} = \tilde{O}(n^{-\frac{1-\beta}{\beta}})$$

The inequalities hold because the gap is at most the sum of the contributions of all the leaves of $\mathcal{T}_{\lambda_1}$ (since they were all expanded), and there are at most n such leaves, since there are at most n *nodes* on this subtree. We also used the inequalities for λ_1 and λ_2 derived above.

When $\beta = 0$, solving again $n < n(\lambda_2)$, we get $\lambda_2 \leq \exp[-(n/a)^{1/b}]$, so as before:

$$\varepsilon^* \leq n\frac{M}{\gamma}\exp[-(n/a)^{1/b}] \leq n\frac{M}{\gamma}\gamma^{cn^{1/b}} = \tilde{O}(\gamma^{cn^{1/b}})$$

for some constant c'. The exponential was rewritten in terms of γ to highlight that the increasing depth in the tree, and hence the decreasing discounting, is the main reason for the decrease in the gap. ■

When β is smaller, the problem is simpler and the bound on the gap decreases faster. In particular, the simplest case is when $\beta = 0$ and the size of the tree increases only logarithmically (it is important to note that in this case, b *must* be strictly positive because the size of $\mathcal{T}_\lambda$ cannot remain constant; this fact was used in the proof above). More insight is provided next, in two interesting, complementary special cases that are analogous to those in Section 4.2.

Case 1: Uniform probabilities Here the probabilities are "flat", unstructured so the problem is difficult: $p_0(i) = p(i,j) = 1/M, \ \forall i, j$. The contribution of any node at depth d is $e(\boldsymbol{\sigma}_d) = \frac{(\gamma/M)^d}{1-\gamma}$, and so the tree $\mathcal{T}_\lambda$ increases uniformly, one depth at a time. Given λ, define $d(\lambda) = \left\lceil \frac{\log \lambda(1-\gamma)}{\log \gamma/M} \right\rceil$ as an upper bound on the depth of $\mathcal{T}_\lambda$. The amount of nodes down to $d(\lambda)$ is $O(M^{d(\lambda)}) = O(M^{\frac{\log \lambda(1-\gamma)}{\log \gamma/M}}) = O(\lambda^{-\frac{\log M}{\log M/\gamma}})$, leading to $\beta = \frac{\log M}{\log M/\gamma}$. By applying Theorem 13, we get $\varepsilon^* = \tilde{O}(n^{-\frac{\log 1/\gamma}{\log M}})$.

The interpretation is that since the algorithm must expand the tree uniformly, it requires large computational effort to increase the depth and decrease the bound. Hence, this bound shrinks slowly (the exponent of n^{-1} is small). In particular, to get to depth d and obtain a gap $\frac{\gamma^d}{1-\gamma}$, the algorithm must expand $n = O(M^d)$ nodes, which is a more direct way to derive the same rate. Note also that the logarithmic term does not appear, so in this case using $\tilde{O}$ instead of O is just an artifact of the general proof of Theorem 13. □

Case 2: Structured probabilities For the second case, we take highly structured probabilities, close to a deterministic problem. Here, the algorithm focuses on high-probability paths and decreases the bound quickly. In particular, take $M = 2$ and p_0, $p(i,\cdot)\ \forall i$ equal to a Bernoulli distribution with probabilities $(q, 1-q)$ and q close to 1. The analysis of $|\mathcal{T}_\delta|$ is quite involved and was performed in the supplementary material of [8]. We give directly the result, $\beta = \frac{\log\left(\frac{e}{\eta}\right)^\eta}{\log 1/(q\gamma)}$ where $\eta = \frac{\log 1/(q\gamma)}{\log 1/(\gamma(1-q))}$. This value becomes smaller when q approaches 1 so the problem gets closer to deterministic. In particular, the limit of β as $q \to 1$ is 0. This recovers a fully deterministic problem, where the algorithm only needs to expand $n = d$ nodes to get to depth d, so the gap is $\varepsilon^* = O(\gamma^n)$. Note that this is a special case of the expression in Theorem 13, for $c' = b = 1$. □

6 Simulation Results

We start by evaluating the approach on several linear switched examples: the first for optimal control PO, the second for worst-case disturbance PW, and the third for stochastic, Markov switching PS. Linear modes are chosen because most of the literature focuses on them, so we can highlight relationships to existing techniques, and at the same time confirm that our approach solves well this baseline linear case. Afterwards, we test the approach on a nonlinear switched system, for PO. In all the experiments, cost bounds G were computed by setting saturation limits on the state variables, and applying the cost function g to these limits. The limits were taken large enough to not be reached along the controlled trajectories. The limit values are given separately for each example.

6.1 *Optimal control of the switching rule for linear modes and quadratic cost*

We solve PO for two linear switched systems: one in which stability can be guaranteed using [18], and another in which it cannot. The first system is Example 3 of [22], discretized with $T_s = 0.01\,$s. The saturation limits confined both state variables to stay at most 1.5 in absolute value. Over a 5 s long trajectory, OP stabilizes the system with an (undiscounted) cost of 25.69 in receding horizon, whereas the design method in [18] gives the larger cost of 32.38. Continuous-time solutions from [22] gave costs that, after rescaling by the sampling time to make them comparable to our discrete-time cost, have value 24.35 and 24.94. So OP gives results close to the state of the art in linear switched design.

The second system is from [16]:

$$A_1 = \begin{bmatrix} 0 & -1.01 \\ 1 & -1 \end{bmatrix}, A_2 = \begin{bmatrix} 0 & -1.01 \\ 1 & -0.5 \end{bmatrix} \qquad (11)$$

Fig. 6 shows successful results when our approach is applied to control the switching rule in receding horizon. In this figure, the initial state of the system is $x_0 = [-3, 3]^\top$ with a quadratic cost and $Q = I$ and state limit 10. We selected $\gamma = 0.98$, a budget $n = 100$, and an experiment length of 80 steps.

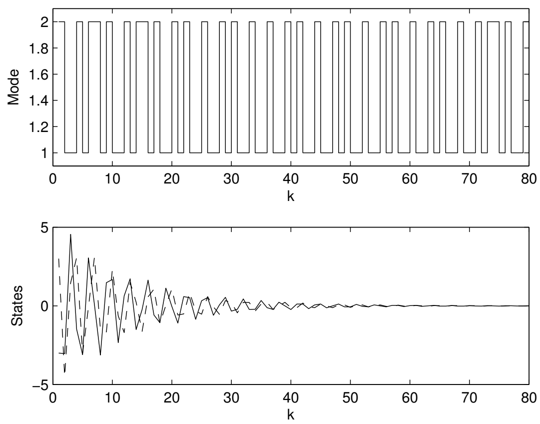

Fig. 6. Optimal control for linear unstable modes.

6.2 *Worst-case disturbance with dwell-time constraint*

Next, we illustrate a problem of type PW: worst-case disturbance. We borrow the example of [18], having two linear modes $A_1 = e^{B_1 T_s}$ and $A_2 = e^{B_2 T_s}$ with:

$$B_1 = \begin{bmatrix} 0 & 1 \\ -10 & -1 \end{bmatrix}, B_2 = \begin{bmatrix} 0 & 1 \\ -0.1 & -0.5 \end{bmatrix}$$

The sampling time is $T_s = 0.5$, the cost is quadratic with $Q = I$ and the initial state is $x_0 = [1, 1]^\top$. In [18] stability is guaranteed under a minimum dwell-time of $\delta = 6$, and we take advantage of this guarantee by applying the constrained algorithm OPδ with $\delta = 6$, and keeping the very first mode constant for 6 steps. We also investigate the simpler solution of just transforming the system into a 6-step one, and then running OP without constraints on the multi-step variant. In both cases, we select $\gamma = 0.98$, a budget $n = 500$ per call of the algorithm, an experiment length of 300 s, and state limit 30.

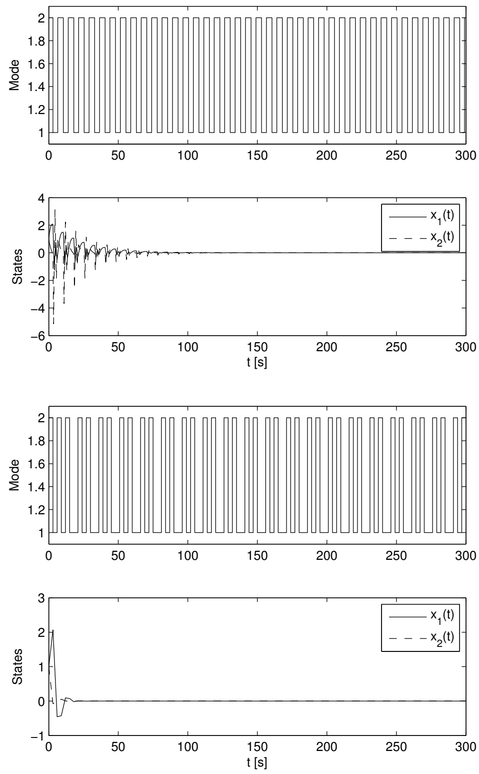

Fig. 7. Controlled trajectories for worst-case disturbance. Top: OPδ, bottom: OP using the multi-step system.

The results for the two approaches are shown in Fig. 7. The undiscounted cost obtained by running OPδ in receding horizon is 142.10, close to the upper bound 152.17 obtained by [18]. Note that we reached our value by *designing* a switching rule, whereas [18] do not. With the multi-step system, we get a cost of 69.94, clearly showing that the extra freedom provided by OPδ pays off.

6.3 *Estimating expected cost with a stochastic switching rule*

To exemplify PS and the results in Section 5, consider the second example of [21], with $M = 4$ second-order linear modes. The goal there was robust control with system uncertainty, not optimal control, so the results will not be comparable, but the system has the appropriate structure. We consider the closed-loop dynamics of each mode when controlled with the feedbacks designed in [21], and set the four unknown transition probabilities so that the overall transition matrix is:

$$p = \begin{bmatrix} 0.3 & 0.2 & 0.1 & 0.4 \\ 0.1 & 0.4 & 0.3 & 0.2 \\ 0.1 & 0.1 & 0.5 & 0.3 \\ 0.2 & 0.3 & 0.4 & 0.1 \end{bmatrix}$$

The initial mode probabilities are taken uniform, $p_0(i) = 0.25\ \forall i$. The initial state is $x_0 = [1, 1]^\top$, the cost function is quadratic with $Q = I_2$, and the state limit is 50.

Algorithm 3 is run with a budget up to $n = 10000$, and the evolution of the lower and upper bounds, together with the gap ε^*, is shown in Fig. 8. The bounds are clearly improving as the budget increases, although of course the improvement slows down due to the exponential costs of the algorithm; Theorem 13 characterizes the asymptotic decrease rate of ε^*. The final bounds for $n = 10000$ are $L^* = 0.2641$, $B^* = 1.1131$.

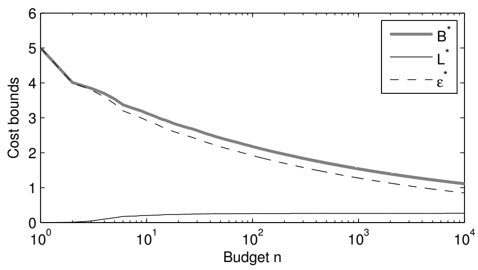

Fig. 8. Results for expected cost evaluation. The values are normalized, under $\overline{p}$. Note the logarithmic horizontal axis.

6.4 *Optimal control for nonlinear modes: Double tank*

For the final example, we consider PO for the double-tank system with nonlinear modes from [40]. The two states of the system correspond to the fluid levels in an upper and a lower tank. The output of the upper tank flows into the lower tank, the output of the lower tank exits the system, and the flow into the upper tank is restricted to be either 1 or 2. The two modes have continuous-time dynamics:

$$\dot{x}(t) = \begin{bmatrix} 1 - \sqrt{x_1(t)} \\ \sqrt{x_1(t)} - \sqrt{x_2(t)} \end{bmatrix}, \dot{x}(t) = \begin{bmatrix} 2 - \sqrt{x_1(t)} \\ \sqrt{x_1(t)} - \sqrt{x_2(t)} \end{bmatrix}$$

Different from [40], we numerically integrate the dynamics over sampling intervals of $T_s = 0.1$ to obtain the discrete-time modes f_1 and f_2, see again (1). The cost is defined as $(x - x^*)^\top Q(x - x^*)$ with $x^* = [0, 3]^\top$, and $Q = \mathrm{diag}(0, 2)$, so the first state is not optimized and the second must reach value 3.

We examine the effect of the computational budget n on performance, which is measured by the undiscounted cost along the trajectory in order to be consistent with usual formulations of optimal control of switched systems. OP is run for a range of budgets from 10 to 200 in increments of 2, using the initial state $x_0 = [2, 2]^\top$, $\gamma = 0.98$, a trajectory length of 20 s, and state limit 5. Fig. 9, top reports the results, showing that the cost decreases with larger budgets as expected, although the differences are small, showing that the problem is simple enough to be solved well with small budgets. Note that the cost no longer decreases for significantly larger budgets, which indicates the solution is likely already optimal (for discounted costs). Fig. 9, bottom shows the trajectory for $n = 200$, which stabilizes the level to 3 in around 6 s, like the approach in [40]. Since time is discrete and the input flow can take only two discrete values, the level must oscillate slightly around the desired value.

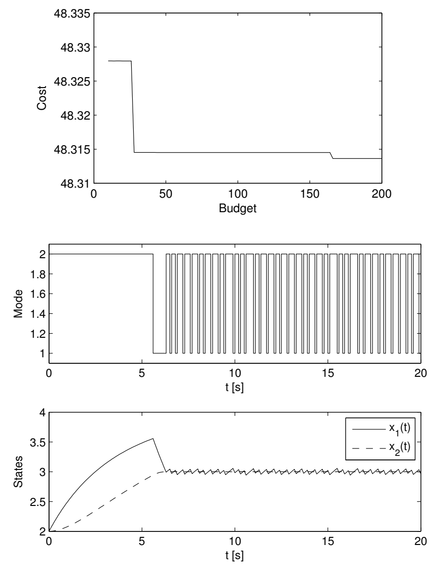

Fig. 9. Top: Influence of computational budget for the nonlinear tanks. Bottom: Control and state trajectories in the same problem.

7 Conclusions and future work

We introduced an approach to optimize or evaluate discounted costs in discrete-time switched systems with nonlinear modes. When the switches are controlled, a switching sequence is sought that minimizes the cost. When the switches are a disturbance, the approach estimates the maximal, worst-case costs (if the switching rule is unknown) or the expected cost (if switching probabilities are known). In the optimal control and worst-case settings, the approach is able to optionally include a minimum dwell-time constraint. It provides upper and lower bounds on the optimal, worst-case, or expected cost depending on the behavior of the switches, and designs a sequence that achieves the bounds in the deterministic cases. The convergence rate of the gap between bounds is characterized as a function of computation.

An important future direction is an explicit treatment of stability guarantees, either by connecting with existing conditions in the switched systems literature, or with our approach for systems without switches in [34]. Starting from other planing algorithms [8,9], approaches can be developed for so-called dual switching systems [4], where some of the switches are controlled and some are a disturbance, evolving either stochastically or with unknown rules. OPδ may also be modified to handle different constraints such as maximum dwell time, periodicity etc. which will require novel complexity analysis.

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] S. C. Bengea and R. A. DeCarlo, "Optimal control of switching systems," *Automatica*, vol. 41, pp. 11–27, 2005.

[3] D. P. Bertsekas and S. E. Shreve, *Stochastic Optimal Control: The Discrete Time Case.* Academic Press, 1978.

[4] P. Bolzern, P. Colaneri, and G. D. Nicolao, "Design of stabilizing strategies for dual switching stochastic-deterministic linear systems," in *Proceedings 19th IFAC World Congress*, Cape Town, South Africa, 24–29 August 2014, pp. 4080–4084.

[5] E. Boukas, *Stochastic Switching Systems: Analysis and design.* Springer, 2006.

[6] M. S. Branicky, "Multiple Lyapunov functions and other analysis tools for switched and hybrid systems," *IEEE Transactions on Automatic Control*, vol. 43, pp. 475–582, 1998.

[7] L. Buşoniu, M.-C. Bragagnolo, J. Daafouz, and C. Morarescu, "Planning methods for the optimal control and performance certification of general nonlinear switched systems," in *Proceedings 54th IEEE Conference on Decision and Control*, Osaka, Japan, 2015.

[8] 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.

[9] L. Buşoniu, E. Páll, and R. Munos, "An analysis of optimstic, 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.

[10] L. Buşoniu, R. Postoyan, and J. Daafouz, "Near-optimal strategies for nonlinear and uncertain networked control systems," *IEEE Transactions on Automatic Control*, vol. 61, no. 8, pp. 2124–2139, 2016.

[11] N. Cardoso De Castro, C. Canudas De Wit, and F. Garin, "Energy-aware wireless networked control using radio-mode management," in *Proceedings 2012 American Control Conference (ACC-2012)*, Montréal, Canada, 27–29 June 2012, pp. 2836–2841.

[12] M. Claeys, J. Daafouz, and D. Henrion, "Modal occupation measures and LMI relaxations for nonlinear switched systems control," *Automatica*, vol. 64, no. 2, pp. 143–154, 2016.

[13] O. Costa, M. Fragoso, and R. Marques, *Discrete-Time Markov Jump Linear Systems.* Springer, 2005.

[14] J. Daafouz, P. Riedinger, and C. Iung, "Stability analysis and control synthesis for switched systems: A switched Lyapunov function approach," *IEEE Transactions on Automatic Control*, vol. 47, pp. 1883–1887, 2002.

[15] D. Du, B. Jiang, and P. Shi, *Fault Tolerant Control for Switched Linear Systems.* Springer, 2015.

[16] M. Fiacchini and M. Jungers, "Necessary and suficient condition for stabilizability of discrete-time linear switched systems: a set-theory approach." *Automatica*, 2014.

[17] J. Filar, V. Gaitsgory, and A. Haurie, "Control of singularly perturbed hybrid stochastic systems," *IEEE Transactions on Automatic Control*, vol. 46, no. 2, pp. 179–190, 2001.

[18] J. C. Geromel and P. Colaneri, "Stability and stabilization of discrete-time switched systems." *International Journal of Control*, vol. 79, no. 7, pp. 719–728, 2006.

[19] J. C. Geromel, G. Deaecto, and J. Daafouz, "Suboptimal switching control consistency analysis for switched linear systems," *IEEE Transactions on Automatic Control*, vol. 58, pp. 1857–1861, 2013.

[20] J. C. Geromel and R. H. Korogui, "H2 robust filter design with performance certificate via convex programming," *Automatica*, vol. 44, pp. 937–948, 2008.

[21] A. P. Gonçalves, A. R. Fioravanti, M. A. Al-Radhawi, and J. C. Geromel, "H_∞ state feedback control of discrete-time Markov jump linear systems through linear matrix inequalities," in *Proceedings of the 18th IFAC World Congress*, Milano, Italy, 28 August – 2 September 2011, pp. 12 620–12 625.

[22] D. Henrion, J. Daafouz, and M. Claeys, "Optimal switching control design for polynomial systems: An LMI approach," in *Proceedings of the IEEE Conference on Decision and Control (CDC-13)*, 2013.

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

[24] M. Jungers and J. Daafouz, "Guaranteed cost certification for discrete-time linear switched systems with a dwell time," *IEEE Transactions on Automatic Control*, vol. 58, no. 3, pp. 768–772, 2013.

[25] 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.

[26] B. Kiumarsi, F. Lewis, H. Modares, A. Karimpour, and M.-B. Naghibi-Sistani, "Reinforcement q-learning for optimal tracking control of linear discrete-time systems with unknown dynamics," *Automatica*, vol. 50, no. 4, pp. 1167–1175, 2014.

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

[28] J. W. Lee and G. E. Dullerud, "Uniformly stabilizing sets of switching sequences for switched linear systems," *IEEE Transactions on Automatic Control*, vol. 52, pp. 868–874, 2007.

[29] H. Li, Y. Gao, P. Shi, and H. K. Lam, "Observer-based fault detection for nonlinear systems with sensor fault and limited communication capacity," *IEEE Transactions on Automatic Control*, vol. 61, no. 9, pp. 2745–2751, 2016.

[30] D. Liberzon, *Switching in Systems and Control.*, ser. Systems and Control: Foundations and Applications. Birkhauser, 2003.

[31] H. Lin and P. J. Antsaklis, "Stability and stabilizability of switched linear systems: A survey of recent results," *IEEE Transactions on Automatic Control*, vol. 54, no. 2, pp. 308–322, 2009.

[32] R. Munos, "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.

[33] S. Pettersson and B. Lennartson, "LMI for stability and robustness for hybrid systems," in *Proceedings 1997 American Control Conference (ACC-97)*, 1997, pp. 1714–1718.

[34] R. Postoyan, L. Buşoniu, D. Nešić, and J. Daafouz, "Stability analysis of discrete-time infinite-horizon optimal control with discounted cost," *IEEE Transactions on Automatic Control*, 2016, in press.

[35] P. Riedinger, C. Iung, and F. Kratz., "An optimal control approach for hybrid systems," *European Journal of Control*, vol. 9, pp. 449–458, 2003.

[36] C. Seatzu, D. Corona, A. Giua, and A. Bemporad., "Optimal control of continuous-time switched affine systems," *IEEE Transactions on Automatic Control*, vol. 51, pp. 726–741, 2006.

[37] M. S. Shaikh and P. Caines, "On the hybrid optimal control problem: Theory and algorithms," *IEEE Transactions on Automatic Control*, vol. 52, pp. 1587–1603, 2007.

[38] R. Shorten, F. Wirth, O. Mason, K. Wulff, and C. King, "Stability criteria for switched and hybrid systems," *Automatica*, vol. 49, no. 7, pp. 545–592, 2007.

[39] A. N. Vargas, E. F. Costa, and J. B. R. do Val, "Bounds for the finite horizon cost of Markov jump linear systems with additive noise and convergence for the long run average cost," in *Proceedings 45th IEEE Conference on Decision and Control (CDC-06)*, San Diego, US, 13–15 Dec 2006, pp. 5543–5548.

[40] R. Vasudevan, H. Gonzalez, R. Bajcsy, and S. S. Sastry, "Consistent approximations for the optimal control of constrained switched systems," *SIAM Journal on Control and Optimization*, 2012.

[41] J. H. W. Zhang and A. Abate, "Infinite-horizon switched LQR problems in discrete time: A suboptimal algorithm with performance analysis," *IEEE Transactions on Automatic Control*, vol. 57, pp. 1815–1821, 2012.

[42] M. Wiering and M. van Otterlo, Eds., *Reinforcement Learning: State of the Art*. Springer, 2012, vol. 12.

[43] X. Xu and P. J. Antsaklis, "Results and perspectives on computational methods for optimal control of switched systems," in *Hybrid Systems: Computation and Control*, 2003.

[44] X. Zhong, H. He, H. Zhang, and Z. Wang, "Optimal control for unknown discrete-time nonlinear Markov jump systems using adaptive dynamic programming," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 25, no. 12, pp. 2141–2155, 2014.

[45] F. Zhu and P. J. Antsaklis, "Optimal control of switched hybrid systems: A brief survey," *Discrete Event Dynamic Systems*, vol. 25, no. 3, pp. 345–364, 2015.