

Near-Optimal Strategies for Nonlinear and Uncertain Networked Control Systems

— [Source link](#) 

Lucian Busoniu, Romain Postoyan, Jamal Daafouz





Institutions: Technical University of Cluj-Napoca, University of Lorraine

Published on: 01 Aug 2016 - IEEE Transactions on Automatic Control (IEEE)

Topics: Networked control system, Transmission (telecommunications), Optimal control, Discrete time and continuous time and Control theory

Related papers:

- [Near-optimal strategies for nonlinear networked control systems using optimistic planning](#)
- [Networked predictive control of constrained nonlinear systems: Recursive feasibility and Input-to-State Stability analysis](#)
- [Networked Predictive Control of Uncertain Constrained Nonlinear Systems: Recursive Feasibility and Input-to-State Stability Analysis](#)
- [Uncertain Aol in Stochastic Optimal Control of Networked and Constrained LTI Systems.](#)
- [Design of networked control systems using a stochastic switching systems approach](#)

Share this paper:    

View more about this paper here: <https://typeset.io/papers/near-optimal-strategies-for-nonlinear-and-uncertain-34oy0t3z2l>



HAL
open science

Near-optimal strategies for nonlinear and uncertain networked control systems

Lucian Busoniu, Romain Postoyan, Jamal Daafouz

► **To cite this version:**

Lucian Busoniu, Romain Postoyan, Jamal Daafouz. Near-optimal strategies for nonlinear and uncertain networked control systems. *IEEE Transactions on Automatic Control*, Institute of Electrical and Electronics Engineers, 2016, 61 (8), pp.2124-2139. 10.1109/TAC.2015.2484358 . hal-01298507

HAL Id: hal-01298507

<https://hal.archives-ouvertes.fr/hal-01298507>

Submitted on 24 Aug 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Near-Optimal Strategies for Nonlinear and Uncertain Networked Control Systems

Lucian Buşoniu Romain Postoyan Jamal Daafouz

Abstract—We consider problems where a controller communicates with a general nonlinear plant via a network, and must optimize a performance index. The system is modeled in discrete time and may be affected by a class of stochastic uncertainties that can take finitely many values. Admissible inputs are constrained to belong to a finite set. Exploiting some optimistic planning algorithms from the artificial intelligence field, we propose two control strategies that take into account the communication constraints induced by the use of the network. Both strategies send in a single packet long-horizon solutions, such as sequences of inputs. Our analysis characterizes the relationship between computation, near-optimality, and transmission intervals. In particular, the first strategy imposes at each transmission a desired near-optimality, which we show is related to an imposed transmission period; for this setting, we analyze the required computation. The second strategy has a fixed computation budget, and within this constraint it adapts the next transmission instant to the last state measurement, leading to a self-triggered policy. For this case, we guarantee long transmission intervals. Examples and simulation experiments are provided throughout the paper.

Index Terms—networked control systems, optimal control, nonlinear systems, planning, predictive control.

I. INTRODUCTION

In a variety of applications, controllers are implemented over networks in order to reduce installation costs and to facilitate maintenance, leading to *networked control systems* (NCS). The control law therefore has to share the communication bandwidth with other network users. This constraint cannot be ignored in general, as it may have a serious impact on the system performance. Various methodologies for NCS have been developed over these last decades. Two main approaches are distinguished based on whether the transmissions are defined by a clock, see e.g. [7], [26], [44], or are triggered depending on the state of the plant, in which case we talk of *event-based control*, see [24] and the references therein.

In this paper, we develop an approach for the near-optimal control of nonlinear NCS, allowing for either time-triggered or self-triggered control strategies. The inputs are constrained to belong to a finite set. We focus on reducing the number of network transmissions, and we achieve this by sending long-horizon solutions such as sequences of inputs, like in e.g., [2], [14], [25], [38], [43]–[45]. This type of sporadic communication at known intervals is important in scheduling,

since it allows the communication protocol to assign priority to the control task only when needed. Further, by only executing the controller at transmission instants, computation is reduced [19]. We also show how to handle stochastic network delays between the controller and the actuators [48]. Other network effects such as packet drop-outs are not considered.

The main contribution of our approach is providing a tight relationship between near-optimality and length of transmission intervals on the one hand, and on the other the computational load of the algorithm. This algorithm includes a complete, explicit implementation of the optimizer. In the *time-triggered* strategy, the transmission intervals and near-optimality are simultaneously adjusted, and we analyze the required computation. The advantage of our *self-triggered* strategy is that computation is directly controlled, while transmission intervals adapt to the current state and may be significantly longer than in the time-triggered setting. All this is done for general, nonlinear and not necessarily smooth systems, and for general bounded rewards, where the optimal control objective is to maximize the discounted sum of rewards.

We are not aware of such explicitly implementable approaches with known computational bounds in the literature on nonlinear NCS. Instead, existing work on time-triggered NCS typically uses model-predictive control to handle delays and packet dropouts, e.g. [2], [43], without considering computation. Furthermore, only a few self-triggered NCS techniques can handle optimal control, and those target linear systems, e.g. [23], [25].

We borrow from artificial intelligence and adapt to NCS two recent *optimistic planning* (OP) algorithms: OP for Deterministic systems (OPD) [28] and OP for stochastic Markov Decision Processes (OP-MDP) [9]. It is these algorithms that lend our method its generality. They solve the optimal control problem at each state encountered by exploring a tree representation of possible sequences of actions (inputs) from that state. Thus, OP is a type of model-predictive control. Several OP algorithms were introduced, e.g. [8]–[10], [33], [36], and showed good performance in problems from control [36], medicine [10], and artificial intelligence [22].

We consider first deterministic systems, where we use OPD and exploit the fact that it returns long and near-optimal sequences of actions [28]. Thus, rather than sending only the first action in the sequence and then rerunning the algorithm, as done in [28], we choose to send a longer subsequence which is stored in a buffer and applied in open loop, as originally proposed by [2]. In the time-triggered strategy, a fixed communication period between the plant and the controller is *freely selected*, by choosing the sequence length. The algorithm is therefore called *Clock-triggered OP* (COP), and we analyze how suboptimality decreases and how

L. Buşoniu is with the Technical University of Cluj-Napoca, Romania, lucian@busoniu.net. R. Postoyan and J. Daafouz are with the Université de Lorraine, CRAN, UMR 7039 and the CNRS, CRAN, UMR 7039, France, {romain.postoyan,jamal.daafouz}@univ-lorraine.fr. This work was supported by a PHC-Brancusi grant, CNCS-UEFISCDI no. 781/2014 and Campus France no. 32610SE; by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PNII-RU-TE-2012-3-0040; and by the ANR project “Computation Aware Control Systems”, ANR-13-BS03-004-02.

computational complexity grows with the selected period. The second strategy enforces a fixed computation budget at every OP execution, and within this budget generates a sequence of actions from the last measured plant state. We analyze how the resulting near-optimality and sequence length (and therefore the communication interval) depend on the state, and call the algorithm *Self-Triggered OP (STOP)*. Sequence-based control was used among others by [14], [25], [38], [43]–[45]. In [45], different from our motivation of reducing network transmissions, the controller finds sequences of actions at steps when computational resources are large, and applies the sequences at future steps at which resources are insufficient.

In the stochastic case, we consider a class of uncertainties modeled by finitely many random outcomes. The core ideas are similar to the deterministic case, but open-loop sequences are no longer appropriate, so we use OP-MDP and a new solution concept: tree policies, which are feedback laws over possible realizations of the uncertainties up to an adaptive length. This also requires a different NCS architecture: a local, computationally cheap feedback controller is directly connected to the system and implements the tree policy, while a computationally powerful OP-MDP controller sits beyond the network. Since the sequence length now depends on the uncertainty realization, COP has no direct counterpart; instead, in the first stochastic strategy a desired near-optimality is imposed at each transmission. The analysis then bounds computation and leads to a probabilistic characterization of the length. For the second, self-triggered strategy, a computational budget is again imposed, and we investigate near-optimality and the related probabilistic lengths.

For both deterministic and stochastic systems, our strategies allow sending only an initial part of the solutions found: subsequences or subtrees. A tuning parameter allows moving from the original OP approach, which only applies the first action, to applying the complete solutions. Interestingly, we show that closing the loop more often does not necessarily lead to better performance, but may do better or worse depending on the problem. This is because solutions that are better in the long term may either be discovered or not, depending on the reward structure. Finally, the stochastic self-triggered method is applied to deal with random delays in the control channel.

Our analysis focuses on near-optimality and transmission intervals, which are important even in the absence of immediate stability guarantees. The connection between stability and optimality would be interesting, but is not yet understood for the discounted type of cost required by OPD and OP-MDP. For instance, when discounting is present stability cannot generally be guaranteed even in the linear case, as illustrated by [31], while [12] emphasized similar difficulties in the NCS setting. Many works in optimal control use discounted costs without guaranteeing stability, such as [1], [29] which consider network effects, as well as works on classic optimal control of linear systems [27], nonlinear systems [34], singularly perturbed systems [21], or systems with discrete-valued variables as in our case [5], [15], [20]. We have made an initial stability analysis under the exactly optimal control [40], but a complete solution is not yet available and falls outside the scope of this paper.

To make the development easier to follow, we first describe fully the deterministic case, starting with OPD and its analysis in Section II, followed by COP and STOP with their analysis and simulations in Section III. The stochastic case is similarly developed, with OP-MDP in Section IV and the NCS methods in Section V. Section VI focuses on the delayed case, and Section VII concludes the paper. We performed a preliminary study in the deterministic case in [11].

II. DETERMINISTIC CASE: BACKGROUND

We introduce the necessary techniques in detail, and we adapt their analysis to the NCS setting.

A. Optimal control problem

Consider an optimal control problem for a deterministic, discrete-time nonlinear system:

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

with state $x \in X$ and action $u \in U$. Here, X is an arbitrary space, such as \mathbb{R}^m or a discrete set. We restrict U to a finite set below. Each transition from x_k to x_{k+1} as a result of u_k is associated with a reward $r_{k+1} = \rho(x_k, u_k)$, and the goal is to find for each state x a sequence of actions $\mathbf{u}_\infty = (u_0, u_1, \dots) \in U^\infty$ that maximizes the infinite-horizon discounted return (value):

$$V^{\mathbf{u}_\infty}(x) = \sum_{k=0}^{\infty} \gamma^k r_{k+1} = \sum_{k=0}^{\infty} \gamma^k \rho(x_k, u_k) \quad (2)$$

where $x_0 = x, x_{k+1} = f(x_k, u_k)$ for $k \geq 0$, and $\gamma \in [0, 1)$ is the discount factor. Elements X, U, f, ρ , and γ together form a deterministic type of Markov decision process (MDP). The optimal value function is defined as $V^*(x) = \sup_{\mathbf{u}_\infty} V^{\mathbf{u}_\infty}(x)$, and under Assumption 1 below always exists and is unique, see Ch. 4 of [4]. Because the system is deterministic, action sequences are sufficient to represent the optimal solution.

Assumption 1: The action space is discrete (or discretized), $U = \{u^1, \dots, u^M\}$. Rewards are bounded in $[0, 1]$.

Reward boundedness is often assumed in the MDP literature, see e.g. Ch. 4 of [4] and [46], since it ensures boundedness of the value in (2). The main way to achieve boundedness is by saturating a possibly unbounded original reward function. This changes the optimal solution, but is often sufficient in practice. Then, the resulting bounded rewards can be normalized to $[0, 1]$. On the other hand, the physical limitations of the system may be meaningfully modeled by saturating the states and actions. In this case, a reward bound follows from the saturation limits.

Many systems have inherently discrete and finitely-many actions, because they are controlled by switches. This is the case e.g. in traffic signal control [18] or water level control by barriers and sluices [47]. When the actions are originally continuous, discretization reduces performance, but the loss is often manageable. Discretized actions may even be preferable due to their benefits in NCS: the size of communication packets can be reduced by encoding the discrete actions by their index, and actuator saturation can be dealt with by simply discretizing within the operating ranges. Other authors showed interest in coarsely-discretized control for NCS, e.g. [17].

B. Optimistic planning for deterministic systems

To introduce the algorithm, in this section we focus on a particular state x where it must be applied, and by convention set the current time to 0, so that $x_0 = x$. Of course, the procedure works at any time step.

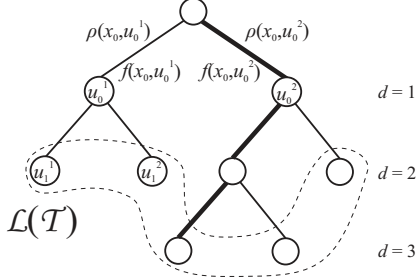


Fig. 1. Illustration of an OPD tree \mathcal{T} . Nodes are labeled by actions, arcs represent transitions and are labeled by the rewards and next states resulting by applying the corresponding action. Subscripts are depths, superscripts index the M possible actions/transitions from a node (here, $M = 2$). The leaves $\mathcal{L}(\mathcal{T})$ are enclosed in a dashed line, while the thick path highlights an action sequence. Note that the root corresponds to the empty sequence.

Optimistic planning for deterministic systems (OPD) [28] explores a tree representation of the possible action sequences from the current state, as illustrated in Figure 1. OPD starts with an unlabeled root node, and iteratively expands nodes, where each expansion adds new children nodes corresponding to all the M actions u^1, \dots, u^M . Each node at some depth d is reached via a unique path through the tree, and can thus be uniquely associated to the sequence of actions $\mathbf{u}_d = (u_0, u_1, \dots, u_{d-1})$ on this path. In what follows, we will work interchangeably with sequences and paths, keeping this equivalence in mind.

For a sequence \mathbf{u}_d , we define three quantities:

$$\ell_x(\mathbf{u}_d) = \sum_{d'=0}^{d-1} \gamma^{d'} \rho(x_{d'}, u_{d'}), \quad b_x(\mathbf{u}_d) = \ell_x(\mathbf{u}_d) + \frac{\gamma^d}{1-\gamma}$$

$$v_x(\mathbf{u}_d) = \ell_x(\mathbf{u}_d) + \gamma^d V^*(x_d)$$

where the states are generated with the action sequence \mathbf{u}_d , like in (2). Subscript x indicates that the three quantities depend on the state $x = x_0$ where OPD is applied. Due to Assumption 1, the rewards (below depth d) are in $[0, 1]$, so $\ell_x(\mathbf{u}_d)$ provides a lower bound on the value of any infinite sequence that starts with \mathbf{u}_d , while $b_x(\mathbf{u}_d)$ is an upper bound. Value $v_x(\mathbf{u}_d)$ is obtained by continuing optimally after \mathbf{u}_d .

We denote the set of sequences corresponding to leaves of \mathcal{T} by $\mathcal{L}(\mathcal{T})$. OPD *optimistically* explores the space of action sequences, by always expanding further a most promising leaf sequence: one with the largest upper bound $b_x(\mathbf{u})$. At the end, a sequence that maximizes the lower bound $\ell_x(\mathbf{u})$ among the leaves is returned. Since leaves sit at varying depths d in the tree so that $\gamma^d/(1-\gamma)$ varies, maximizing $\ell_x(\mathbf{u})$ is different from maximizing $b_x(\mathbf{u})$, and can intuitively be seen as making a safe choice. Algorithm 1 summarizes the entire procedure, where function $\Delta(\cdot)$ gives the depth of a tree, and any ties among several sequences maximizing upper or lower bounds are broken arbitrarily. We allow the algorithm to terminate either after a given number of expansions, or after a node at given depth d has been expanded, leading

Algorithm 1 Optimistic planning for deterministic systems

Input: state x , budget n or depth d (set the other to ∞)
1: initialize tree: $\mathcal{T} \leftarrow \{\text{root}\}$, $i = 0$
2: **repeat**
3: find optimistic sequence: $\mathbf{u}^\dagger \in \arg \max_{\mathbf{u} \in \mathcal{L}(\mathcal{T})} b_x(\mathbf{u})$
4: add children $u^j, j = 1, \dots, M$ to the node of \mathbf{u}^\dagger
5: $i \leftarrow i + 1$
6: **until** $i = n$ or $\Delta(\mathcal{T}) = d + 1$
7: $n \leftarrow i$; $d \leftarrow \Delta(\mathcal{T}) - 1$
Output: $\mathbf{u}^* \in \arg \max_{\mathbf{u} \in \mathcal{L}(\mathcal{T})} \ell_x(\mathbf{u})$, d , n

to $\Delta(\mathcal{T}) = d + 1$. Sometimes a sequence of length $\Delta(\mathcal{T})$ may be returned, in which case the last action is removed for uniformity of analysis. The computational budget is measured by the number of expansions, since an expansion takes M calls to the model f and to the reward function ρ , and for nonlinear systems computing f dominates the execution time. Other tree operations (such as computing b-values or traversing the tree to find the optimistic sequence) are significantly cheaper, but can be bounded between $O(n \log n)$ and $O(n^2)$, depending on the branching factor $\kappa(x)$ defined in the next section.

C. Theoretical guarantees

To characterize the complexity of finding the optimal sequence from a given state x , we use the branching factor $\kappa(x)$ (average number of children per node) of the near-optimal subtree. This subtree contains only nodes that, given the rewards obtained down to them in the tree, cannot be ruled out as belonging to optimal sequences. In general, exploring these nodes is unavoidable, and $\kappa(x)$ is in this sense necessary to characterize the problem. OPD *only* explores the near-optimal subtree, leading to a priori guarantees on the relation between computation, sequence length, and near-optimality. Since $\kappa(x)$ is generally unknown, actual values for e.g. near-optimality cannot be determined in advance. Nevertheless, the analysis provides confidence that OPD automatically adapts to the complexity of the problem at state x , described by $\kappa(x)$. We return to detail these properties after the formal development is in place.

The near-optimal subtree is defined as $\mathcal{T}^*(x) = \{\mathbf{u}_d \mid d \geq 0, V^*(x) - v_x(\mathbf{u}_d) \leq \gamma^d/(1-\gamma)\}$. Let $\mathcal{T}_d^*(x)$ be the set of nodes at depth d on $\mathcal{T}^*(x)$ and $|\cdot|$ denote set cardinality, then the asymptotic branching factor is defined as $\kappa(x) = \limsup_{d \rightarrow \infty} |\mathcal{T}_d^*(x)|^{1/d}$.

A sequence \mathbf{u}_d is said to be ε -optimal when $V^*(x) - v_x(\mathbf{u}_d) \leq \varepsilon$. The upcoming theorem is a consequence of the analysis in [28], [39]. It is given here in a form that brings out the role of the sequence length, useful for the NCS application in the sequel. Part (i) of the theorem shows that OPD returns a long and near-optimal sequence, and parts (ii), (iii) show that sequence length and near-optimality are closely related to the computation budget, via branching factor $\kappa(x)$.

Theorem 1: Let $x \in X$. When OPD is called at x :

- (i) The length of the sequence \mathbf{u}^* returned is $d = \Delta(\mathcal{T}) - 1$. Denoting $\varepsilon(x) = V^*(x) - \ell_x(\mathbf{u}^*)$, we have $\varepsilon(x) \leq \frac{\gamma^d}{1-\gamma}$.

- (ii) When OPD is called with large target depth d : • If $\kappa(x) > 1$ it will require a number of expansions¹ $n(x) = O(\kappa(x)^d)$. • If $\kappa(x) = 1$, $n(x) = O(d)$.
- (iii) When OPD is called with large budget n : • If $\kappa(x) > 1$ it will reach a depth of $d(x) = \Omega(\frac{\log n}{\log \kappa(x)})$, and $\varepsilon(x) = O(n^{-\frac{\log 1/\gamma}{\log \kappa(x)}})$. • If $\kappa(x) = 1$, $d(x) = \Omega(n)$ and $\varepsilon(x) = O(\gamma^{c(x)n})$, where $c(x)$ is a constant. \square

Proof: (sketch) Item (i) follows from the proof of Theorem 2 in [28], (ii) from the proof of Theorem 3 in [28], and (iii) from the proofs of Theorems 2 and 3 in [28]. \blacksquare

The sequence returned is also $\varepsilon(x)$ -optimal, since $V^*(x) - v_x(\mathbf{u}^*) \leq V^*(x) - \ell_x(\mathbf{u}^*) \leq \varepsilon(x)$ in view of part (i); the second inequality here is stronger than $\varepsilon(x)$ -optimality.

These results rely on the core property that OPD only expands nodes in $\mathcal{T}^*(x)$, although it uses solely reward information from the current tree [28], [39]. To build more intuition on $\mathcal{T}^*(x)$ and $\kappa(x)$, note that $\mathcal{T}^*(x)$ contains sequences for which it is impossible to tell, from their rewards down to d , whether or not they are part of an optimal solution, because their near-optimality is smaller than the amount of reward $\gamma^d/(1-\gamma)$ they might accumulate below depth d . Usually only some sequences have this property, therefore $\mathcal{T}^*(x)$ is smaller than the complete tree and $\kappa(x)$ is smaller than the number of actions M . The smaller $\kappa(x)$, the more easily near-optimal sequences can be distinguished, and the better OPD does. The best case is $\kappa(x) = 1$, obtained e.g. when a single sequence always obtains rewards of 1, and all the other rewards are 0. In this case the algorithm must only develop this sequence, and suboptimality decreases exponentially. In the worst case, $\kappa(x) = M$, obtained e.g. when all the sequences have the same value, the algorithm must explore the complete tree in a uniform fashion, expanding nodes in order of their depth.

III. OPD FOR DETERMINISTIC NETWORKED CONTROL SYSTEMS

A. Setting

We now focus on a networked-control setting, in which actuation and state signals are exchanged over a network that must be efficiently utilized. To this end, the controller should only communicate with the plant when needed. OPD is well equipped to handle this case, since it guarantees that it will return *long and near-optimal sequences* of actions.

We envision the following setup, see Figure 2. The sequence of transmission instants is denoted by k_i , $i \in \{0, 1, 2, \dots\}$, and it will either be fixed by the user or defined by the controller itself. At each k_i , the controller receives the state's measurement and generates a sequence of control actions which is sent as a single packet to the actuators' buffer, like in [2], [14], [25], [43], [44]. The actuators then apply the k' -th component of the sequence to the plant at step $k_i + k'$, until the full sequence has been used. Afterwards, the new state's measurement is sent to the controller and the procedure is repeated. The number of transmissions is reduced, since the

channel is only used at intervals equal to the sequence lengths.

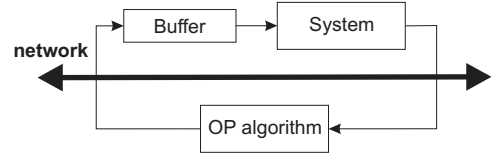


Fig. 2. NCS architecture in the deterministic case.

B. Algorithms

Algorithm 1 and Theorem 1 suggest two ways in which OPD could be exploited for NCS. The first possibility is to impose a desired sequence length (planning depth) d at every controller execution step, and then send to the plant either the full sequence or an initial subsequence thereof. Denoting the length of the sent subsequence by $d' \leq d$, this means the communication between the controller and the plant is set to occur with a period d' . Applying OPD in this way is novel. Since length d and the controller execution interval $d' \leq d$ are freely selected, this first strategy is called Clock-triggered OP (COP); it is summarized in Algorithm 2. The

Algorithm 2 COP: Clock-triggered optimistic planning

Input: initial state x_0 , target depth d , subsequence length d'

- 1: $k \leftarrow 0$
 - 2: **loop**
 - 3: measure current state x_k
 - 4: apply OPD(x_k, d), obtaining a sequence \mathbf{u}_d
 - 5: send initial subsequence $\mathbf{u}_{d'}$ to plant
 - 6: $k \leftarrow k + d'$, wait d' steps
 - 7: **end loop**
-

Algorithm 3 STOP: Self-triggered optimistic planning

Input: initial state x_0 , budget n , subsequence fraction α

- 1: $k \leftarrow 0$
 - 2: **loop**
 - 3: measure current state x_k
 - 4: apply OPD(x_k, n), obtaining a sequence $\mathbf{u}_{d(x)}$
 - 5: send initial subsequence $\mathbf{u}_{\lceil \alpha d(x) \rceil}$ to plant
 - 6: $k \leftarrow k + \lceil \alpha d(x) \rceil$, wait $\lceil \alpha d(x) \rceil$ steps
 - 7: **end loop**
-

second possibility is to impose the computation budget n , like in the classical application of OPD, and let the algorithm find the longest sequence it can within this budget. Then, different from classical OPD which sends just one action, we send again either the whole sequence or a subsequence. The returned sequence length depends in addition to n also on the current state, through the planning complexity as expressed by branching factor $\kappa(x)$. Therefore, the algorithm is self-triggered and we call it Self-Triggered OP (STOP); it is summarized as Algorithm 3. To allow sending subsequences, STOP is parameterized by the fraction $\alpha \in (0, 1]$, so that if a sequence of length d is returned by OPD, only the first $\lceil \alpha d \rceil$ actions are actually sent and applied, where $\lceil \cdot \rceil$ denotes the ceiling operator.

¹Let $g, h : (0, \infty) \rightarrow \mathbb{R}$. 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 $\forall t \leq t_0$.

C. Analysis

An algorithm is called ε -optimal if it applies in closed loop a sequence \mathbf{u}_∞ satisfying $V^*(x_0) - V^{\mathbf{u}_\infty}(x_0) \leq \varepsilon$. Consider first COP.

Theorem 2: For any d and $d' \leq d$, the following hold. (a) COP is $\frac{\gamma^d}{1-\gamma}$ -optimal. (b) For large d , at every state x where it is called, COP requires: $\bullet n(x) = O(\kappa(x)^d)$ expansions if $\kappa(x) > 1$; $\bullet n(x) = O(d)$ expansions if $\kappa(x) = 1$, with $\kappa(x)$ the branching factor of Section II-C. \square

Proof: The second part of the theorem is a consequence of Theorem 1(ii). To prove part (a), denote by \mathbf{u}^0 the sequence returned by OPD when applied at x_0 , and recall that $\varepsilon(x_0) = V^*(x_0) - \ell_{x_0}(\mathbf{u}^0)$. If the full sequence is applied, then no matter what actions are taken afterwards at least value $\ell_{x_0}(\mathbf{u}^0)$ is obtained, so COP is $\varepsilon(x_0)$ -optimal.

Now, consider applying a subsequence \mathbf{u}'^0 strictly shorter than \mathbf{u}^0 , and then reexecuting OPD in the resulting state x^1 to obtain \mathbf{u}^1 , see Figure 3. Denote by \mathbf{u}'^0 the leftover subsequence from \mathbf{u}^0 . For arbitrary sequences \mathbf{u} and $\tilde{\mathbf{u}}$, let $(\mathbf{u}, \tilde{\mathbf{u}})$ denote their concatenation.

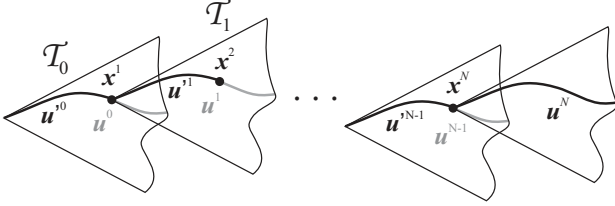


Fig. 3. Using OPD with subsequences. Different from Figure 1, the trees are now oriented horizontally.

When applied from x^1 , OPD builds the tree \mathcal{T}_1 by expanding nodes in the exactly the same order as it would have expanded, when applied from x_0 , nodes in the subtree of \mathcal{T}_0 having x^1 at root. That is, for any sequence $\tilde{\mathbf{u}}^1$ in \mathcal{T}_1 , the following b-value relationship holds by definition: $b_{x_0}(\mathbf{u}^0, \tilde{\mathbf{u}}^1) = \ell_{x_0}(\mathbf{u}^0) + \gamma^{d_1} b_{x^1}(\tilde{\mathbf{u}}^1)$, where d_1 is the depth of x^1 in \mathcal{T}_0 . So, maximizing $b_{x^1}(\tilde{\mathbf{u}}^1)$ is the same as maximizing $b_{x_0}(\mathbf{u}^0, \tilde{\mathbf{u}}^1)$ with respect to $\tilde{\mathbf{u}}^1$. Because OPD is applied with the same setting in x^1 as in x_0 , it will expand more nodes and so \mathbf{u}'^0 is inside \mathcal{T}_1 . Since \mathbf{u}^1 maximizes ℓ_{x^1} on \mathcal{T}_1 , we have $\ell_{x^1}(\mathbf{u}^1) \geq \ell_{x^1}(\mathbf{u}'^0)$, which means the composite sequence satisfies: $\ell_{x_0}(\mathbf{u}^0, \mathbf{u}^1) = \ell_{x_0}(\mathbf{u}^0) + \gamma^{d_1} \ell_{x^1}(\mathbf{u}^1) \geq \ell_{x_0}(\mathbf{u}^0) + \gamma^{d_1} \ell_{x^1}(\mathbf{u}'^0) = \ell_{x_0}(\mathbf{u}^0)$ where d_1 is the depth of x^1 .

Continuing in a similar fashion, for any N , applying $N-1$ shorter sequences followed by the full N th sequence achieves $\ell(\mathbf{u}^0, \mathbf{u}^1, \dots, \mathbf{u}^{N-1}, \mathbf{u}^N) \geq \ell_{x_0}(\mathbf{u}^0)$, see again Figure 3. Thus the same is true of the limit as $N \rightarrow \infty$, and since this limit is the value $V^{\mathbf{u}_\infty}(x_0)$ of the overall closed-loop sequence, we have obtained $V^*(x_0) - V^{\mathbf{u}_\infty}(x_0) \leq V^*(x_0) - \ell_{x_0}(\mathbf{u}^0) \leq \varepsilon(x_0)$. To obtain the final result, notice that by Theorem 1(i), $\varepsilon(x_0) \leq \frac{\gamma^d}{1-\gamma}$. \blacksquare

Thus, the quality of the solution grows with the imposed sequence length d , and the computation requirements to reach this length are bounded and characterized using $\kappa(x)$. Specifically, computation grows exponentially in d , with base $\kappa(x)$ – unless $\kappa(x) = 1$, in which case it grows linearly in d . Next, we move on to STOP.

Theorem 3: Take any large budget n and any $\alpha \in (0, 1]$. (a) The near-optimality of STOP is: $\bullet O(n^{-\frac{\log 1/\gamma}{\log \kappa(x_0)}})$ if $\kappa(x_0) > 1$, and $\bullet O(\gamma^{cn})$ if $\kappa(x_0) = 1$. (b) At every state x where it is called, STOP produces a sequence of length: $\bullet d(x) = \Omega(\frac{\log n}{\log \kappa(x)})$ if $\kappa(x) > 1$, and $\bullet d(x) = \Omega(n)$ if $\kappa(x) = 1$. \square

Proof: It directly follows by reapplying the proof of Theorem 2(b) that STOP is $\varepsilon(x_0)$ -optimal, and using the expressions of $\varepsilon(x_0)$ from Theorem 1(iii) completes the first part. The second part follows directly from Theorem 1(iii). \blacksquare

The performance guarantee of STOP depends only on the planning difficulty at the initial state x_0 : it is a negative power of n when $\kappa(x_0) > 1$, and exponential (better) in n when it is $\kappa(x_0) = 1$. The sequence length grows fast, in a way that is characterized using $\kappa(x)$, and which basically ‘inverts’ the relationship between computation and length in COP.

It must be emphasized that the analysis is performed under the assumption that the model is correct. This is the main reason for which the subsequence length (represented by d' in COP and α in STOP) does not affect the near-optimality guarantee: there is no loss, whether the loop is closed sooner or later. Also, the full initial sequence could be applied and followed by arbitrary actions, while still guaranteeing $\frac{\gamma^d}{1-\gamma}$ -optimality. No predictive algorithm can do better in general without increasing the horizon, because the rewards are not assumed to be smooth so they may change unfavorably beyond the horizon explored at the first step. Of course, in practice uncertainty is always present, as model errors or disturbances, which means the sequences cannot be too long and the loop must be closed fairly often.

Even when the model is correct, some nontrivial relations arise between shorter and longer sequences: applying shorter sequences – closing the loop more often – may achieve better or worse performance, depending on the problem. The following result characterizes this behavior, in a general way that applies to both COP and STOP.

Theorem 4: Let $x \in X$ and denote by \mathbf{u}_d the sequence returned by OPD at x . Let $\mathbf{u}_{d'}$ be an initial subsequence of \mathbf{u}_d and \mathbf{u}_{d_1} be obtained by replanning after $\mathbf{u}'_{d'}$ (see Figure 4). Define similarly $\mathbf{u}_{d''}$ and \mathbf{u}_{d_2} with $d'' > d'$. Then:

$$v_x(\mathbf{u}_{d'}, \mathbf{u}_{d_1}) \geq v_x(\mathbf{u}_{d''}, \mathbf{u}_{d_2}) - \frac{\gamma^{d'+d_1}}{1-\gamma}$$

Furthermore, if the budget or target depth of OPD are held constant, then the bound is tight in a worst-case sense. \square

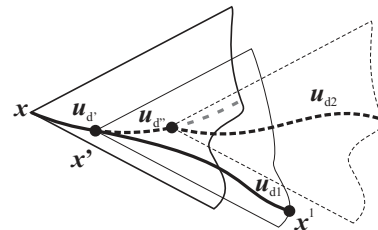


Fig. 4. Shorter versus longer subsequences.

Proof: Denote by x' and x'' the states reached by \mathbf{u}'_d and \mathbf{u}''_d , respectively. The inequality is shown as follows:

$$\begin{aligned} v_x(\mathbf{u}'_d, \mathbf{u}_{d_1}) &= \ell_x(\mathbf{u}'_d) + \gamma^{d'} v_{x'}(\mathbf{u}_{d_1}) \\ &= \ell_x(\mathbf{u}'_d) + \gamma^{d'} V^*(x') - \gamma^{d'} [V^*(x') - v_{x'}(\mathbf{u}_{d_1})] \\ &\geq v_x(\mathbf{u}'_d) - \frac{\gamma^{d'+d_1}}{1-\gamma} \geq v_x(\mathbf{u}''_d, \mathbf{u}_{d_2}) - \frac{\gamma^{d'+d_1}}{1-\gamma} \end{aligned}$$

where the first step follows from the definition of the v -value, the second just adds and subtracts an extra term, the third follows from Theorem 1(i) when applied at x' , and finally the last step is true because v cannot increase if more actions are added to the sequence.

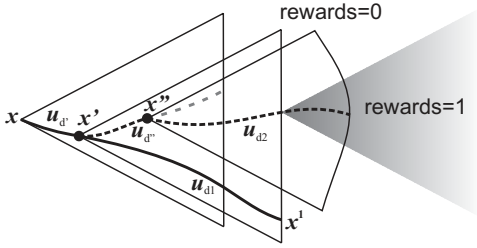


Fig. 5. Constructing an example where the bound is tight.

To show tightness, a worst-case example is provided where the bound holds with equality. Construct a problem, in the form of a tree, where all rewards are 0 except for one subtree placed below x'' at depth $d' + d_1$, in which they are all 1, see Figure 5. Note that due to the zero rewards until $d' + d_1$, up until this depth all trees will be expanded uniformly. When OPD is applied to find u_d and u_{d_1} , it cannot discriminate between sequences since they all have a lower bound ℓ equal to 0, so OPD must choose one arbitrarily. We take the arbitrary sequence u_{d_1} so that it does not contain x'' , leading to a value $v_x(\mathbf{u}'_d, \mathbf{u}_{d_1}) = 0$.

When OPD is called at x'' , it starts expanding nodes uniformly, and since this state is at depth $d'' > d'$ and OPD has the same budget or target depth as at x' , it will expand at least a node at depth $d' + d_1$. We simply place the subtree with rewards of 1 under this node, thereby ensuring that the algorithm discovers it and that the sequence $(\mathbf{u}''_d, \mathbf{u}_{d_2})$ has the optimal value $\frac{\gamma^{d'+d_1}}{1-\gamma}$. So the bound is tight. ■

The theorem says that applying a shorter sequence and then replanning may lose some performance, but not too much: the maximum loss is given by the accuracy of the *entire composite sequence* $(\mathbf{u}'_d, \mathbf{u}_{d_1})$, i.e., $\frac{\gamma^{d'+d_1}}{1-\gamma}$. Further, from the worst-case example it is clear that the same loss can be incurred even if the loop is closed again sooner than d_1 or d_2 . The following examples provide more insight into this issue, using COP as it allows to directly control the (sub)sequence length.

Example 1: Shorter sequences can perform better. Consider an MDP with state space $\{1, 2, \dots, 5\}$, two actions $-1, 1$ (“left” and “right”), and additive dynamics $x_{k+1} = \max(1, \min(5, x_k + u_k))$. The rewards obtained upon reaching each of the five states are, respectively, 0.8, 0.7, 0.5, 0.8, 0, and the discount factor is 0.8, see Figure 6.

When applied from $x_0 = 4$ with $d = 2$ and $d' = 1$, COP replans in $x_1 = 3$, which allows it to detect the larger rewards

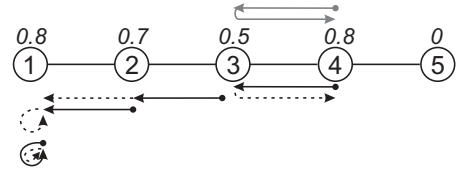


Fig. 6. A five-state MDP and two COP solutions. States are shown in circles, and rewards in italics above them. The solution from $x_0 = 4$ with $d' = d = 2$ is shown in gray on top of the figure, while the one for $d' = 1$, $d = 2$ is shown in black on the bottom. Solutions are shown as sequences of actions, where the bullets mark the states in which planning is run, and unapplied sequence tails are shown in dashed lines.

to the left. It eventually reaches state 1 and remains there, achieving the optimal return of 3.62. However, when d' is increased to 2, COP exploits the rewards of states 4 and 5 and cycles between these states forever, obtaining a suboptimal return of 3.17. □

Example 2: Longer sequences can perform better. A similar MDP is taken but now with state space $\{1, 2, \dots, 7\}$ and the rewards shown in Figure 7. The discount factor is the same.

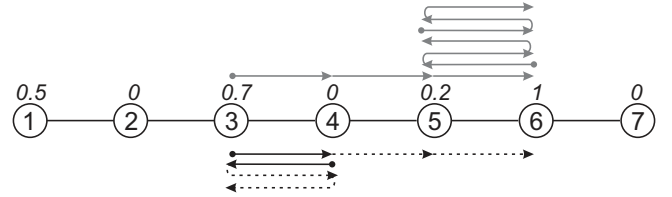


Fig. 7. A seven-state MDP and two COP solutions, for $d' = d = 3$ (top, gray) and $d' = 1$, $d = 3$ (bottom, black).

Now, when applied from $x_0 = 3$ with $d' = d = 3$, COP discovers the large reward in state 6 and controls towards this state, cycling afterwards between 5 and 6 for a return of 2.22. When $d' = 1$ however, replanning from state 4 misleads the algorithm into a shorter-horizon cycle that focuses on the reward 0.7, achieving only a suboptimal return of 1.56. □

D. Simulation results for a DC motor

We study the behavior of COP and STOP in simulations with a DC motor. See also [11] for a nonlinear example. Discretizing in time a first-principles model of the DC motor, with the zero-order-hold method and $T_s = 0.01$ s, we obtain:

$$f(x, u) = Ax + Bu, \quad A \approx \begin{bmatrix} 1 & 0.0095 \\ 0 & 0.9100 \end{bmatrix}, \quad B \approx \begin{bmatrix} 0.0084 \\ 1.6618 \end{bmatrix} \quad (3)$$

where $x_1 = \alpha$ is the shaft angle, $x_2 = \dot{\alpha}$ the angular velocity, and u the voltage. Moreover, the states and actions are restricted using saturation to $\alpha \in [-\pi, \pi]$ rad, $\dot{\alpha} \in [-15\pi, 15\pi]$ rad/s, $u \in [-30, 30]$ V, in order to represent physical limitations in the system. The goal is to stabilize the system around $x = 0$, and is described by the reward function:

$$\rho(x, u) = -x^\top Qx - u^\top Ru, \quad Q = \text{diag}(5, 0.001), \quad R = 0.01 \quad (4)$$

with discount factor $\gamma = 0.9$. State and action saturation ensure bounded rewards, and these rewards are then rescaled into $[0, 1]$. The actions are discretized into the set $U = \{-10, -3, 0, 3, 10\}$.

We apply the two algorithms from $x_0 = [2\pi/3, \pi]^\top$, setting $d = 10$ for COP and $n = 300$ for STOP. Figure 8 shows the solutions obtained when the complete returned sequences are applied, that is, when $d' = 10$ and respectively $\alpha = 1$. It is interesting to see the evolution of the planning complexity along the trajectory. This is shown in COP by the changing computation number n of expansions required to reach the desired sequence lengths, where the practical effects of Theorem 2(b) are seen; and in STOP by the lengths produced, illustrating Theorem 3(b). Complexity is generally smaller in states closer to the equilibrium (fewer expansions/longer sequences), although the evolution is not always monotonic. STOP especially requires only three controller executions and transmissions, thanks to a very long last sequence.

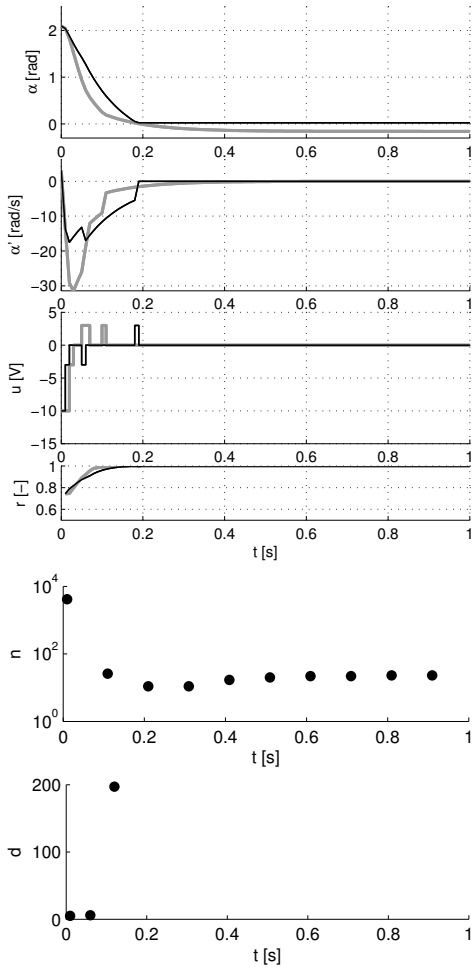


Fig. 8. Comparison between COP and STOP when applying complete sequences. The top graphs are the controlled trajectories, with COP in gray and STOP in black. The bottom two graphs show, at every controller execution instant: for COP, the computation n spent; and for STOP: the length d of the sequences found. The horizontal coordinates of the points in these graphs are also the transmission times.

To investigate the effect of applying shorter sequences, we vary for COP $d' = 1, 2, \dots, 10$ and for STOP $\alpha = 0.1, 0.2, \dots, 1$; the returns obtained are shown in Figure 9. The suboptimality of each return obeys the upper bounds of Theorem 2(a) for COP and Theorem 3(a) for STOP; e.g., the COP bound is $0.9^{10}/(1-0.9) \approx 3.49$. Although the loss from Theorem 4 is unavoidable in general, in this problem shorter

sequences are indeed better, e.g. STOP gains significantly more return when α is below 0.5.

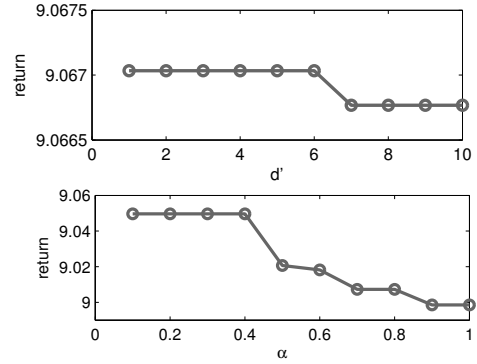


Fig. 9. Returns obtained by COP (top) and STOP (bottom) as the length of the applied subsequence varies.

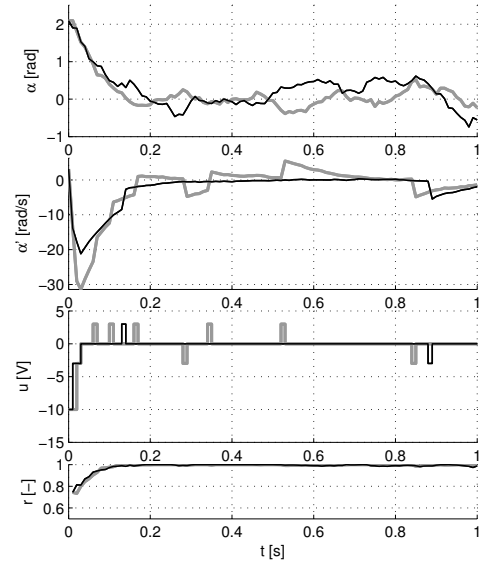


Fig. 10. Performance with a noisy system (COP in gray and STOP in black).

Finally, the resilience of the deterministic-case algorithms to random disturbances is empirically evaluated, by adding a Gaussian disturbance to the discrete-time transitions (3). The disturbance is zero-mean and has covariance matrix $\text{diag}(0.01, 0.01)$. COP and STOP employ the original deterministic model during planning. Representative trajectories are shown in Figure 10, illustrating that the algorithms work reasonably despite the rather large disturbance amplitudes. In addition to these results, in the remainder of the paper we provide a detailed analysis of stochastic extensions of COP and STOP for a class of discrete uncertainty.

IV. STOCHASTIC CASE: BACKGROUND

In the second part of the paper, we provide methods that allow a class of stochastic uncertainties.

A. Control problem and optimistic planning algorithm

In the stochastic case, the dynamics change to [3]:

$$x_{k+1} \sim \tilde{f}(x_k, u_k, \cdot) \quad (5)$$

where the transition probability function \tilde{f} provides for each pair (x, u) the probability distribution $\tilde{f}(x, u, x')$ over next states x' . The reward function is extended to also allow a dependence on the next state: $r_{k+1} = \tilde{\rho}(x_k, u_k, x_{k+1})$. We still require discrete actions and bounded rewards, as in Assumption 1. Moreover, we focus on uncertainties that can be modeled by a *finite* number of outcomes with known probabilities.

Assumption 2: For any pair (x, u) , the number of next states reachable with nonzero probability is at most integer $N > 0$.

This class of problems disallows continuous uncertainty, such as the Gaussian disturbance empirically studied in Section III-D. Nevertheless, it is highly relevant, including many discrete-event systems [13, Ch. 7,9] such as Markov jump systems [16], and with important applications in power systems [6], fault detection [35], building automation [37], etc., see also our application to stochastic network delays in Section VI. This is also the general form of MDPs typically studied in artificial intelligence [42] and operations research [41], so the variant of OP for these stochastic systems is called OP-MDP [9]. Of course, \tilde{f} may also be the discretization of an originally continuous distribution, a procedure related to scenario generation in stochastic programming [30].

Before stating the optimal control objective, some preparatory steps are necessary. Like OPD, OP-MDP works at the current system state, conventionally denoted x_0 . It explores iteratively an infinite tree that represents all possible stochastic evolutions of the system starting from x_0 . Denote a state node by s , labeled by an actual state x . The planning tree \mathcal{T}_∞ , of which Figure 11 only shows a few top nodes, is defined recursively as follows. First, the root node s_0 is labeled by the current state x_0 , and then each node s is expanded by adding, for any state x' for which $\tilde{f}(x, u, x') > 0$ for some u , a new child node s' labeled by x' . So each node has at most NM children, corresponding to all possible states reachable by applying all possible actions. Note that Figure 11 explicitly includes also the action nodes.

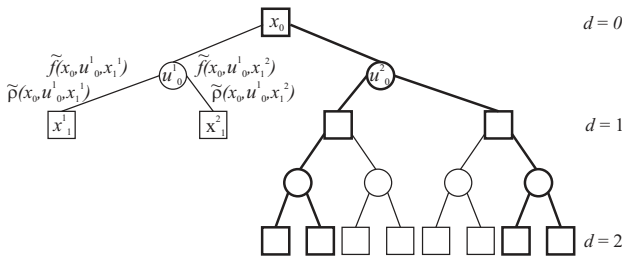


Fig. 11. Illustration of OP-MDP tree for $N = M = 2$. The squares are state nodes labeled by states x , and the actions u are explicitly included as circle nodes. Transition arcs to next states are labeled by probabilities \tilde{f} and rewards $\tilde{\rho}$. Superscripts index the possible actions and state outcomes, while subscripts are depths, which only increase with the state node levels. The thick subtree highlights a tree policy.

The open-loop action sequences from OPD would be sub-optimal in the stochastic case, since they cannot react to the realization of the random transitions. Instead, a closed-loop local solution concept called a *tree policy* is needed. At depth d , this tree policy is an assignment of actions to all state outcomes under the previous action choices, thereby selecting

only some nodes of \mathcal{T}_∞ :

$$\mathcal{T}_0 = \{s_0\}, \text{ and for any } d \geq 0: h_d: \mathcal{T}_d \rightarrow U, \\ \mathcal{T}_{d+1} = \{s' \in \mathcal{T}_\infty | s' \text{ is a child of } s \text{ along action } h_d(s)\}$$

where h_d assigns actions as desired. Then, the overall selected tree is $\mathcal{T}_{h_\infty} = \bigcup_{d \geq 0} \mathcal{T}_d$, and the policy itself is $h_\infty: \mathcal{T}_\infty \rightarrow U$, $h_\infty(s) = h_{d(s)}(s)$ where $d(s)$ gives the depth of s .

The objective is then to find, locally at x_0 , a policy h_∞ maximizing the *expected* return:

$$V^{h_\infty}(x_0) = \mathbb{E}_{h_\infty} \left\{ \sum_{k=0}^{\infty} \gamma^k r_{k+1} \right\} \quad (6)$$

where the expectation is taken over all trajectories in \mathcal{T}_{h_∞} . Finally, denote the optimal value $V^*(x_0) = \sup_{h_\infty} V^{h_\infty}(x_0)$.

OP-MDP works of course with finite tree policies, denoted simply by h and exemplified in Figure 11. These policies must correspond to well-defined subtrees \mathcal{T}_h at the top of \mathcal{T}_∞ , so that any node is either fully expanded or not at all. The leaves of \mathcal{T}_h are denoted by \mathcal{L}_h . We will treat policies h and their corresponding trees \mathcal{T}_h interchangeably. Similarly to OPD, define three values:

$$\ell_x(h) = \sum_{s \in \mathcal{L}_h} P(s) R(s) \\ b_x(h) = \sum_{s \in \mathcal{L}_h} P(s) \left[R(s) + \frac{\gamma^{d(s)}}{1-\gamma} \right] \quad (7) \\ v_x(h) = \sum_{s \in \mathcal{L}_h} P(s) \left[R(s) + \gamma^{d(s)} V^*(x(s)) \right]$$

where $R(s)$ is the discounted return accumulated along the path from the root to s , and $P(s)$ is the probability of reaching leaf s , equal to the product of the individual transition probabilities along the path. So, $\ell_x(h)$ is the expected partial return accumulated by h , and is a lower bound on the expected returns of all complete, infinite policies h_∞ starting with h ; $b_x(h)$ is an upper bound on these expected returns; and $v_x(h)$ is the expected return when continuing optimally below h . It is important to note that $b_x(h) = \ell_x(h) + \sum_{s \in \mathcal{L}_h} P(s) \frac{\gamma^{d(s)}}{1-\gamma}$. We denote the sum in this expression by $\text{diam}(h)$, called the diameter of h ; and $c(s) = P(s) \frac{\gamma^{d(s)}}{1-\gamma}$, the contribution of node s to the diameter. Since the lower and upper bounds on the values of policies starting with h are separated by $\text{diam}(h)$, this diameter is an uncertainty on $v_x(h)$, and $c(s)$ quantifies the contribution of s to this uncertainty.

OP-MDP builds a subtree \mathcal{T} of \mathcal{T}_∞ by refining at each iteration an optimistic policy that maximizes b_x ; and at the end, it returns a policy maximizing ℓ_x . Thus the approach is similar to OPD, with the major difference that now a policy has multiple leaf nodes so a choice between them must be made. This is resolved by selecting for expansion a node with maximal contribution to the diameter. Algorithm 4 summarizes the approach. The algorithm can stop either after n node expansions or after reaching a predefined diameter for the optimistic policy δ . Value δ has a dual meaning: diameter and near-optimality, see the upcoming guarantees. Note that expanding a node takes up to N times more simulations than in the deterministic case.

Algorithm 4 Optimistic planning for MDPs

Input: state x , and budget n or desired diameter δ

- 1: initialize tree: $\mathcal{T} \leftarrow \{\text{root}\}$, $i = 0$
- 2: $h^\dagger \leftarrow$ initial empty policy, $\bar{\delta} \leftarrow \frac{1}{1-\gamma}$
- 3: **repeat**
- 4: expand node $s^\dagger \in \arg \max_{s \in \mathcal{L}_{h^\dagger}} c(s)$
- 5: find new optimistic policy $h^\dagger \in \arg \max_{h \in \mathcal{T}} b_x(h)$
- 6: $\bar{\delta} \leftarrow \min\{\bar{\delta}, \text{diam}(h^\dagger)\}$, $i \leftarrow i + 1$
- 7: **until** $i = n$ or $\text{diam}(h^\dagger) \leq \delta$
- 8: **if** input was n , **then** $\delta \leftarrow \bar{\delta}$, **else** $n = i$ **end if**

Output: $h^* \in \arg \max_{h \in \mathcal{T}} \ell_x(h)$, δ , n

B. Theoretical guarantees

While in OPD only one sequence u_d was refined at a given iteration, in OP-MDP expanding a single node refines *all* policies that reach it with a positive probability. To handle this global effect, a new complexity measure called near-optimality exponent $\beta(x)$ is required [9]. This exponent is related to other measures of complexity for stochastic optimization [8], [32], which however do not take global refinement into account, and $\beta(x)$ serves that purpose. To formalize $\beta(x)$, define the largest diameter of any (finite) tree policy to which the contribution of s is the greatest: $\bar{\delta}(s) = \sup_{h \in H(s)} \text{diam}(h)$, $H(s) = \{h \mid s \in \mathcal{L}_h, c(s) = \max_{s' \in \mathcal{L}_h} c(s')\}$. This characterizes the global impact of node s . Then, define the set of nodes that have large impact on near-optimal policies:

$$S_\varepsilon(x) = \{s \in \mathcal{T}_\infty \mid \bar{\delta}(s) \geq \varepsilon, \text{ and } \exists h_\infty \ni s \text{ s.t. } V^*(x) - V^{h_\infty}(x) \leq \bar{\delta}(s)\} \quad (8)$$

where we have made explicit the dependence of S_ε on the current, root state x . Finally, the near-optimality exponent is defined as the smallest number $\beta(x)$ so that for small² ε $|S_\varepsilon(x)| = \tilde{O}(\varepsilon^{-\beta(x)})$. This measures the complexity of the optimal control problem at x .

Similarly to the deterministic case, a policy h is ε -optimal if $V^*(x) - v_x(h) \leq \varepsilon$, and define $\varepsilon(x) = V^*(x) - \ell_x(h^*)$, immediately meaning that h^* is $\varepsilon(x)$ -optimal. The following results are a consequence of the analysis of OP-MDP in [9], [39], and are stated so as to emphasize the role of the diameter.

Theorem 5: Let $x \in X$. When OP-MDP is called at x :

- (i) The policy h^* satisfies $\text{diam}(h^*) \leq \delta$ and $\varepsilon(x) \leq \delta$, with δ the diameter returned.
- (ii) When OP-MDP is called with small near-optimality (diameter) δ :
 - If $\beta(x) > 0$ it will require $n(x) = \tilde{O}(\delta^{-\beta})$ expansions.
 - If $\beta(x) = 0$, $n(x) = O((\log 1/\delta)^{b(x)})$ with $b(x) > 0$ a constant.
- (iii) When OP-MDP is called with large n :
 - If $\beta(x) > 0$ it will obtain diameter $\delta(x) = \tilde{O}(n^{-1/\beta(x)})$.
 - If $\beta(x) = 0$, $\delta(x) = \exp[-(n/a(x))^{1/b(x)}]$, where $a(x), b(x) > 0$ are constants. \square

When $\beta(x)$ is smaller, the problem is simpler. The simplest problem, for $\beta(x) = 0$, can e.g. be obtained with a

² $f(t) = \tilde{O}(g(t))$ for small (or large) t when $\exists a, b, t_0 > 0$ so that $f(t) \leq a(\log g(t))^b g(t) \forall t \leq t_0$ (or $\forall t \geq t_0$). The logarithmic term asymptotically becomes negligible compared to $g(t)$.

deterministic system ($N = 1$) having a single policy with rewards of 1; this reduces to OPD with $\kappa(x) = 1$. In this case, the results say that computation scales logarithmically with desired diameter/near-optimality δ – or, conversely, δ shrinks exponentially with n . More generally, when the problem is deterministic, the generalized development from this section reduces to the deterministic case, by taking $\beta(x) = \log(\kappa(x))/\log(1/\gamma)$ and $\delta = \frac{\gamma^d}{1-\gamma}$. The most difficult stochastic problem is for $\beta(x) = \log(NM)/\log(1/\gamma)$, when all rewards in the tree are equal and the probabilities are uniform, making the solutions impossible to differentiate and requiring a uniform exploration of the tree.

V. OP-MDP FOR STOCHASTIC NETWORKED CONTROL SYSTEMS

A. Setting

The main idea in applying OP-MDP to uncertain NCS is similar to the deterministic case: the network is used only sporadically to measure the state, a new tree policy is computed based on this measurement, and this policy is then sent via the network to the system, where it is locally applied. However, using tree policies instead of action sequences leads to several key new elements.

First, the architecture must now include two controllers. A high-level controller sits beyond the network and implements OP-MDP, see Figure 12. This controller is assumed to be computationally powerful. Instead of the buffer κ from Figure 2, a second, local controller is introduced, which is directly connected to the system but is computationally much simpler and does not perform any optimization. The local controller applies the tree policy in closed loop, starting from the root state x_k . At each step, it applies the action the policy indicates, measures the resulting realization of the next state, compares it to all the children states of the applied action, and moves the pointer in the tree to the matching child. This local loop only costs $O(N)$ computation, to compare with the N children states. It is executed until reaching a leaf state, at which point the local controller signals that the higher-level loop must be closed via the network. The memory required by the local controller to store the tree policy varies up to $O(n)$, since it is at worst proportional to the size of the entire OP-MDP tree developed.

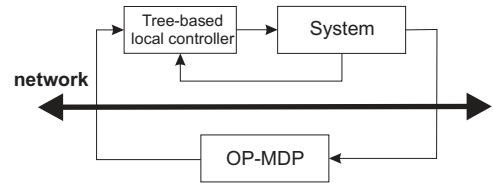


Fig. 12. NCS architecture in the stochastic case.

The second novelty is that the sequence actually applied has a random length, which must be probabilistically characterized. To this end, we define the *effective length* of a policy h , the value d so that:

$$\frac{\gamma^d}{1-\gamma} = \mathbb{E}_{s \in \mathcal{L}_h} \left\{ \frac{\gamma^{d(s)}}{1-\gamma} \right\} = \text{diam}(h) \quad (9)$$

Note that d is different from the *expected value* of the length.

Finally, it will also be interesting to apply shorter subpolicies so as to close the loop more often, e.g. to reduce the size of data packets transmitted and the memory required at the local controller. Formally, from a policy with diameter δ , one can obtain subpolicies with larger diameters δ' ; this is equivalent to changing the effective length from the initial $d = \frac{\log \delta(1-\gamma)}{\log \gamma}$ to the smaller $d' = \frac{\log \delta'(1-\gamma)}{\log \gamma}$. Our analysis is largely independent on the actual procedure to find the subpolicy, but Algorithm 5 shows one possibility. The diameter of the output policy h' will be close to, and smaller than, δ' .

Algorithm 5 Tree policy truncation

Input: \mathcal{T}_h , desired diameter δ'

- 1: **repeat** starting from $\mathcal{T}_{h'} = \{\text{root}\}$
- 2: add to $\mathcal{T}_{h'}$ all children of $s^\dagger \in \arg \max_{s \in \mathcal{L}_{h'}, s \notin \mathcal{L}_h} c(s)$
- 3: **until** $\text{diam}(\mathcal{T}_{h'}) \leq \delta'$

Output: $\mathcal{T}_{h'}$

B. Algorithms

There are two ways to adapt Algorithm 4 for NCS. A counterpart to COP is obtained by setting at every transmission a desired near-optimality δ , which is equivalent to setting a diameter and, through (9), an effective length d . Either the full policy, or a subpolicy with diameter δ' is sent via the network, and the local controller takes over. The larger δ' corresponds to a smaller effective length d' , but the actual length of the applied sequence is random and may be different from d' , so this scheme is no longer clock-triggered. It is more accurately called Diameter-triggered OP for Stochastic systems (DOPS), and shown in Algorithm 6.

Algorithm 6 DOPS

Input: initial state x_0 , target diam. δ , subpolicy diam. δ'

- 1: **loop**
 - 2: measure current state x_k
 - 3: apply OP-MDP(x_k, δ), obtaining policy h
 - 4: truncate h to δ' (e.g. with Algorithm 5), obtaining h'
 - 5: send h' to plant
 - 6: wait until policy exhausted
 - 7: **end loop**
-

The second alternative is to call OP-MDP as usual, with a fixed computation budget n , but then apply either the full policy or a subpolicy with a larger diameter. The length of the applied sequence depends stochastically on the complexity at the current state, via the diameter, see (9). We have therefore obtained Self-Triggered OP for Stochastic systems, STOPS, see Algorithm 7. Parameter $\alpha \in (0, 1]$ controls the truncation, and is chosen to give an α -times smaller effective depth of the subpolicy, leading to the diameter formula on line 4.

C. Analysis

We begin by analyzing DOPS. An algorithm is called ε -optimal if in closed-loop it generates an infinite policy h_∞ satisfying $V^*(x_0) - V^{h_\infty}(x_0) \leq \varepsilon$.

Theorem 6: Take any $\delta > 0$ and $\delta' < \delta$. (a) DOPS is δ -optimal if the full policies are applied, and $\frac{\delta}{1-\delta'(1-\gamma)}$ -optimal

Algorithm 7 STOPS

Input: initial state x_0 , budget n , diameter fraction α

- 1: **loop**
 - 2: measure current state x_k
 - 3: apply OP-MDP(x_k, n), obtaining policy h
 - 4: truncate h to diameter $\frac{[\text{diam}(h)(1-\gamma)]^\alpha}{1-\gamma}$, obtaining h'
 - 5: send h' to plant
 - 6: wait until policy exhausted
 - 7: **end loop**
-

if subpolicies are applied. (b) For small δ , at every state x where it runs planning, DOPS requires: • $n(x) = \tilde{O}(\delta^{-\beta(x)})$ expansions if $\beta(x) > 0$; • $n(x) = O((\log 1/\delta)^{b(x)})$ expansions if $\beta(x) = 0$, where $b(x)$ is a constant. \square

Proof: The computation bounds follow directly from Theorem 4, (iii). From part (i), $V^*(x_0) - \ell_{x_0}(h_0) \leq \delta$, where h_0 is the policy found at initial state x_0 . Since $\ell_{x_0}(h_0)$ is a lower-bound on the value of *any* policy after h_0 , fully applying h_0 followed by any actions maintains the bound.

The case of subpolicies is more involved. Instead of applying h_0 , a subpolicy h'_0 is used, and upon reaching any leaf $s' \in \mathcal{L}'_0$ of h'_0 , OP-MDP is applied again to find a new policy $h_1(s')$; see Figure 13 for a graphical illustration. For now consider that the full new policy is applied, and denote by h_1 the policy resulting from joining h'_0 with the new policies $h_1(s')$ at all s' . Then:

$$\begin{aligned}
 V^*(x_0) - v_{x_0}(h_1) &= \\
 &= V^*(x_0) - \sum_{s' \in \mathcal{L}'_0} P(s') \left[R(s') + \gamma^{d(s')} v_{x(s')}(h_1(s')) \right] \\
 &= V^*(x_0) - \sum_{s' \in \mathcal{L}'_0} P(s') \left[R(s') + \gamma^{d(s')} V^*(x(s')) \right] \\
 &\quad + \sum_{s' \in \mathcal{L}'_0} P(s') \gamma^{d(s')} \left[V^*(x(s')) - v_{x(s')}(h_1(s')) \right] \\
 &\leq \delta + \sum_{s' \in \mathcal{L}'_0} P(s') \gamma^{d(s')} \delta = \delta + \gamma^{d'} \delta \tag{10}
 \end{aligned}$$

where d' is the effective length corresponding to δ' . The first equality follows from the definition of v -values, the second is obtained by adding and subtracting the optimal values at s' , and the third step is due to the near-optimality of OP-MDP at both the root state and any s' .

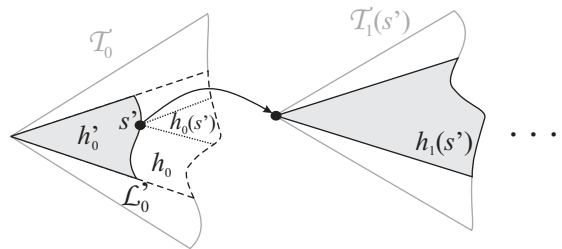


Fig. 13. Using OP-MDP with subpolicies. The full trees developed are also shown in gray outline. Note the new planning tree and policy from s' are shown displaced from their root s' for readability.

Now, if every $h_1(s')$ is truncated at δ' , a similar inequality holds for each such truncated policy; and combining this with

the derivation above, the bound $\delta + \gamma^{d'}\delta + \gamma^{2d'}\delta$ is obtained for the overall policy. Continuing recursively like this, in the limit we obtain $\frac{\delta}{1-\gamma^{d'}} = \frac{\delta}{1-\delta'(1-\gamma)}$. ■

Thus, computation depends on planning complexity, as expressed by the near-optimality exponent $\beta(x)$. The desired δ bounds the closed-loop near-optimality when full policies are applied. For subpolicies, an extra denominator $1 - \gamma^{d'}$ arises from the proof. Nevertheless, a strong intuition confirmed in experiments indicates this extra term is conservative and δ will always bound the closed-loop performance. A similar property can actually be proven for STOPS, as described next.

Theorem 7: Take any large n and any $\alpha \in (0, 1]$. (a) The near-optimality of STOPS is: • $\tilde{O}(n^{-\frac{1}{\beta(x_0)}})$ if $\beta(x_0) > 1$, and • $O(\exp[-(n/a(x_0))^{1/b(x_0)}])$ if $\beta(x_0) = 0$. (b) at every state x where it is called, STOPS returns a policy of diameter: • $\delta(x) = \tilde{O}(n^{-\frac{1}{\beta(x)}})$ if $\beta(x) > 1$, and • $\delta(x) = O(\exp[-(n/a(x))^{1/b(x)}])$ if $\beta(x) = 0$. Here, a and b are problem- and state-specific constants. □

Proof: The second part follows from Theorem 5(iii). When h_0 is fully applied ($\alpha = 1$), the first part holds as in Theorem 6.

Otherwise, examine the case when the first policy h_0 is truncated at diameter $\delta' > \text{diam}(h_0)$; and consider again the composite policy h_1 , see again Figure 13. Its ℓ -value satisfies:

$$\begin{aligned} \ell_{x_0}(h_1) &= \sum_{s' \in \mathcal{L}'_0} P(s') \left[R(s') + \gamma^{d(s')} \ell_{x(s')}(h_1(s')) \right] \\ &\geq \sum_{s' \in \mathcal{L}'_0} P(s') \left[R(s') + \gamma^{d(s')} \ell_{x(s')}(h_0(s')) \right] = \ell_{x_0}(h_0) \end{aligned}$$

where $h_0(s')$ is the part of h_0 below s' , which gets replaced by $h_1(s')$. The crucial relation is $\ell_{x(s')}(h_1(s')) \geq \ell_{x(s')}(h_0(s'))$, which holds because OP-MDP expands nodes in the same order in the subtree of s' . Since its budget is still n , when called at s' it will expand at least all the nodes on $h_0(s')$ and in the end choose a policy with at least as large an ℓ -value. We can get a similar inequality for any $h_1(s')$ that is truncated instead of being fully applied, and recursively repeating this we get in the limit that $V^{h_\infty}(x_0) \geq \ell_{x_0}(h_0)$, where h_∞ is the complete policy that would be applied in closed loop. Hence, this policy is near-optimal at least to the extent of h_0 , completing the proof. ■

Like in the deterministic-case STOP, near-optimality depends only on the planning complexity at the initial state x_0 . The diameter shrinks like a power of n when $\beta(x) > 0$, or faster, exponentially, when $\beta(x) = 0$, which implies a growth rate similar to that in STOP of the effective length d and thus of d' . More precisely, d is of the order $\frac{\log n}{\beta(x)}$, or $(n/a(x))^{1/b(x)}$ when $\beta(x) = 0$.

The closed-loop bound is independent of the subpolicy diameter in STOPS, and has a dependence on this diameter that is not believed to be very informative in DOPS, as explained after Theorem 6. So a more direct characterization of the values of subpolicies will be useful.

Theorem 8: Consider that OP-MDP returns a policy h_0 with diameter δ , which is truncated to $\delta' < \delta$, and replanning is run

from any leaf of the subpolicy, obtaining a maximal diameter δ_1 . Then, the value of the composite policy h_1 satisfies:

$$v_{x_0}(h_1) \geq v_{x_0}(h_0) - \delta' \delta_1 (1 - \gamma)$$

Furthermore, if the budget or target diameter are held constant and Algorithm 5 is used to find the subpolicy, then the bound is tight in a worst-case sense. □

Proof: The inequality is shown similarly to (10):

$$\begin{aligned} v_{x_0}(h_1) &= \sum_{s' \in \mathcal{L}'_0} P(s') \left[R(s') + \gamma^{d(s')} v_{x(s')}(h_1(s')) \right] \\ &= \sum_{s' \in \mathcal{L}'_0} P(s') \left[R(s') + \gamma^{d(s')} V^*(x(s')) \right] \\ &\quad - \sum_{s' \in \mathcal{L}'_0} P(s') \gamma^{d(s')} \left[V^*(x(s')) - v_{x(s')}(h_1(s')) \right] \\ &\geq v_{x_0}(h_0) - \sum_{s' \in \mathcal{L}'_0} P(s') \gamma^{d(s')} \delta_1 \\ &\geq v_{x_0}(h_0) - \gamma^{d'} \delta_1 = v_{x_0}(h_0) - \delta' \delta_1 (1 - \gamma) \end{aligned}$$

with the difference that $h_1(s')$ can stop at a different diameter δ_1 , and exploiting the fact that the v -value of a policy can only increase by truncation, since optimal choices are made earlier, at s' . Here, d' is again the effective depth for δ' .

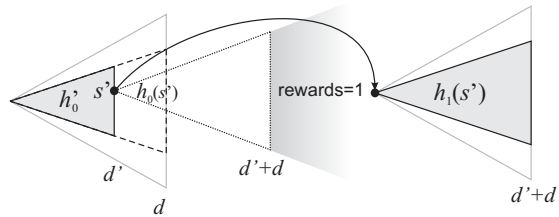


Fig. 14. Worst-case example in the stochastic case. Notation and styles are reused from Figure 13.

To construct a worst-case example, the deterministic example of Figure 5 is extended by changing all action sequences into policy trees with uniform probabilities, $\tilde{f}(x, u, x') = 1/N$. Choose some length d and take $\delta_1 = \delta = \frac{\gamma^d}{1-\gamma}$, or, if n is used, $n = \sum_{i=0}^{d-1} (NM)^i$, so as to fully expand up to depth d in a uniform tree; and take $\delta' = \frac{\gamma^{d'}}{1-\gamma}$, $d' < d$. Construct the problem in Figure 14, where all rewards are 0 except on certain subtrees at depth $d' + d$, as explained below. Due to the 0 rewards and uniform probabilities, at x_0 as well as any s' the algorithm will expand a uniform tree up to depth d , and since all policies have ℓ -value 0, it will arbitrarily choose the output policies. Then, below the composite policy h_1 we assign zero rewards, so that its overall value is 0. For each s' , we pick the parts $h_0(s')$ of arbitrary policy h_0 to be different from $h_1(s')$, and finally we assign rewards of 1 at all nodes below depth $d' + d$ downstream of $h_0(s')$. This is done for all s' , and so replanning at any leaf node of h_0 will surely discover the rewards of 1, leading to an overall closed-loop value of $\frac{\gamma^{d'+d}}{1-\gamma}$. The example is complete. ■

D. Simulation results for the inverted pendulum swingup

DOPS and STOPS are applied to swing up and balance an underactuated inverted pendulum. The states are θ (angle) and

$\dot{\theta}$, while the input voltage u is limited to $[-3, 3]$ V, insufficient to push up the pendulum in one go; instead, the pendulum needs to be swung back and forth to gather energy. Rewards are quadratic (4) with $Q = \text{diag}[1, 0.001]$ and $R = 0.01$, and $\gamma = 0.95$. The pendulum is a relevant problem due to its highly nonlinear solution and large depths to which the swings must be planned. Actions are discretized into $\{-3, 0, 3\}$, and an unreliable actuator is modeled that only applies the intended action u with probability 0.7, and with probability 0.3 it applies only $0.6u$ (when the intended action is 0 it stays 0). This leads to a discrete uncertainty with $N = 2$ values.

We apply DOPS with $d = 8$ and STOPS with $n = 2000$, from the pointing down state. We state results in terms of effective lengths, to make them easy to compare with the deterministic case. Figure 15 shows the returns for varying subpolicy lengths, illustrating the bounds of Theorems 6(a) and 7(a). Shorter subpolicies are usually better.

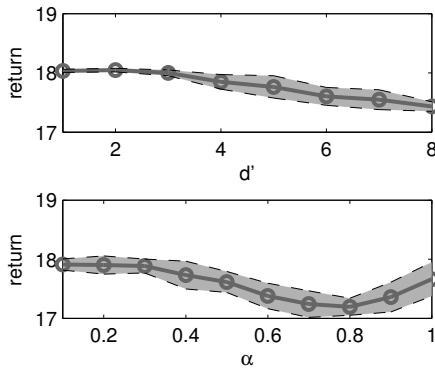


Fig. 15. Returns obtained by DOPS (top) and STOPS (bottom). The shaded area is the 95% confidence interval on the mean, from 10 experiments.

Figure 16 shows a trajectory for STOPS with $\alpha = 0.4$. The swingup is achieved, and in the bottom graph we notice that states close to the start, where the swingup must be planned, are difficult and lead to short policies/large diameters, as characterized by Theorem 7(b). The graph also illustrates the practical effects of the probabilistic relationship (9) between effective length d' and the (random) length of the actually applied sequence. The two are usually different, but d' may still be useful as a qualitative prediction of the length.

Before moving on, it is important to discuss some computational implications of discretization, in the deterministic as well as the stochastic case. When the actions are originally continuous, their discretization will usually contain a number of points M exponential in the action dimension, and similarly for the uncertainty with N discretized points. Recalling that each node expansion has complexity M or NM , OP suffers in this sense from the curse of dimensionality. This is the price to pay for the high generality of the method, recalling that it works for nonlinear, nonsmooth dynamics and general rewards. A crucial point of OP analysis is that the *number* of expansions does not directly depend on NM – but only on the complexity measures $\kappa(x)$ or $\beta(x)$. Finally, in some problems the actions or uncertainties might be natively discrete, coming e.g. from discrete phenomena in the network. We detail such a case in the next section.

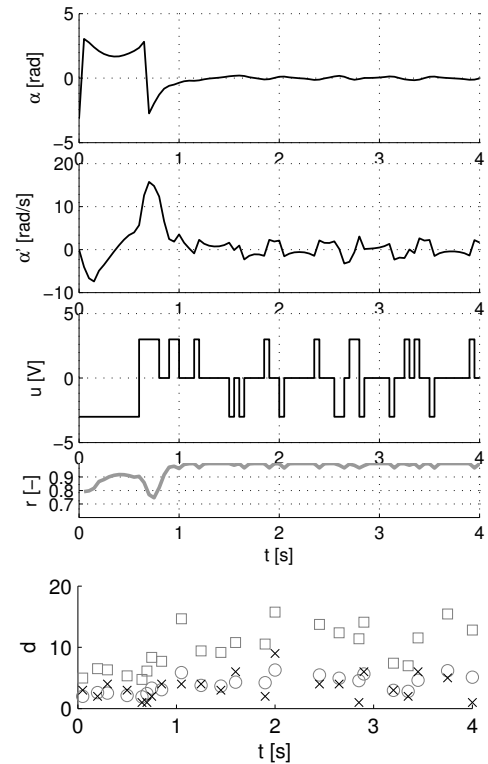


Fig. 16. Top: a controlled trajectory. Bottom: effective length of full policies (\square), of subpolicies (\circ), and real length of the applied sequences (\times).

VI. RANDOM DELAYS IN THE CONTROL CHANNEL

This section shows how our stochastic framework in Figure 12 can be applied to deal with a type of network effects: random delays in the transmission channel for control packets. Packets must arrive in the order they were sent, and the measurement channel should still be delay-free to accurately signal when OP must be rerun. Receipt of the control packets does not have to be acknowledged, since the local tree-based controller has all the information needed to react to the delay, as explained below. The delays must be a multiple of the sampling time, and are modeled by a probability distribution $p : \{0, 1, 2, \dots\} \rightarrow [0, 1]$, where $p(j)$ is the probability that the packet is received with a delay of j steps. The delay is at most J steps, and any delay up to J has nonzero probability:

Assumption 3: The distribution p is known and time-invariant. Further, $\exists J, p(j) = 0 \forall j > J$ and $p(j) > 0 \forall j \leq J$.

This is related to the setting of [48], which also applies predictive control under random delays; while that approach additionally allows delays on the measurement channel, it is limited to linear dynamics. In our setting, the controlled system is taken deterministic with dynamics f and rewards ρ , and we aim to maximize the *expected return under the random delays*. This return is usually smaller than the original optimal value under the deterministic dynamics. We consider STOPS with fully applied tree policies, and characterize its performance relative to the maximal expected return, as well as the transmission intervals.

By convention, let $k = 0$ be the current step where the control is sent, and denote u^- the previously applied action, which will be maintained as long as the new control packet

does not arrive. A stochastic MDP is defined with augmented state $\tilde{x} = [x^\top, j]^\top \in X \times \{-1, 0, \dots, J\}$, $\tilde{x}_0 = [x_0^\top, 0]^\top$, where $j = -1$ means the packet has been received, while $j \geq 0$ means a delay of j has occurred so far. The state evolves and rewards are assigned according to the previous action, and j increases, until the packet arrives, modeled by the following MDP dynamics and rewards:

$$\tilde{x}_{k+1} = \begin{cases} [f^\top(x_k, u^-), j_k + 1]^\top & \text{w.p. } 1 - q(j_k) \text{ if } j_k < J \\ [f^\top(x_k, u_k), -1]^\top & \text{w.p. } q(j_k) \text{ if } j_k < J \\ [f^\top(x_k, u_k), -1]^\top & \text{w.p. } 1 \text{ if } j_k \in \{-1, J\} \end{cases}$$

$$r_{k+1} = \begin{cases} \rho(x_k, u^-) & \text{if } j_{k+1} \geq 0 \\ \rho(x_k, u_k) & \text{if } j_{k+1} = -1 \end{cases}$$

where $q(j)$ is the probability that the packet is received at $j < J$ given that it was not received so far, $q(j) = p(j)/[1 - \sum_{i=0}^{j-1} p(i)]$, $0 \leq j < J$. The MDP defined in this way is denoted m_0 and depends on the state and default action at the current step. The entire OP-MDP analysis in Section IV-B carries through by replacing the dependence on the current state by a dependence on the entire MDP m_0 in quantities $V, v, \ell, b, S_\varepsilon$, and β , e.g. the optimal value at \tilde{x}_0 is $V_{m_0}^*(\tilde{x}_0)$. This dependence is marked in subsequent notation.

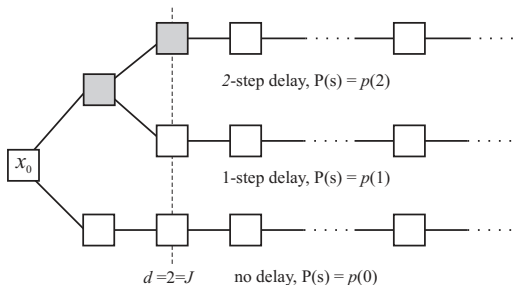


Fig. 17. A tree policy for random delays. Only the state nodes are shown.

Actions are applied by near-optimally reacting to the delay realization, using a tree policy with the structure from Figure 17. At depth 0, either the packet is received and the intended action is applied, leading to the white node at depth 1, or a delay occurs (gray node at depth 1). The branch corresponding to the first case is deterministic, while the second, delayed branch faces two similar outcomes; and so on until depth J , taken 2 in the figure. From J , all branches are deterministic. Of course, all nodes on the j -step delay branch have probability $p(j)$. The local controller determines the delay j with which the packet was received, by comparing the packet timestamp with the current time, and applies the sequence on branch j starting at depth j , thereby reacting in closed loop to the delay. The overall STOPS protocol is obtained by applying OP-MDP to the augmented MDP m at each step where it is called. At each such step, a tree policy h is found and sent using the architecture of Figure 12.

While the results of Section V could be applied off-the-shelf, the restriction to STOPS with full policies allows us to show a stronger, interesting property: that STOPS behaves in a certain sense like the deterministic STOP and so the effects of the delay are mild. Since the MDP m changes with the step where OP-MDP is applied, the optimal value also changes,

and the notion of near-optimality must be reconsidered. We therefore define near-optimality with respect to the initial MDP m_0 : an algorithm is ε -optimal if its overall policy h_∞ it applies in closed loop satisfies $V_{m_0}^*(\tilde{x}_0) - V_{m_0}^{h_\infty}(\tilde{x}_0) \leq \varepsilon$. The upcoming Theorem 9 bounds this as well as the smallest transmission interval.

Consider the *entire* OP-MDP tree \mathcal{T}_∞ , which can be imagined as in Figure 17 with an additional branching into M discrete actions at every node, see also Figure 11. Consider also the subtree $\mathcal{T}(x(s_J))$ having as root some node s_J at depth J on \mathcal{T}_∞ . The notation is justified by the fact that this subtree is, in fact, the deterministic OPD tree for state $x(s_J)$, since downstream of the random delay the problem is the time-invariant, deterministic one. Recall also branching factor $\kappa(x(s_J))$ of the near-optimal subtree $\mathcal{T}^*(x(s_J))$, see Section II-C. The performance of STOPS will then be dictated by $\kappa^*(m_0) := \max_{s_J} \kappa(x(s_J))$, i.e. the most difficult deterministic node encountered after any delay, which is intuitive since STOPS must take into account all such states. To prove this, the analysis of OP-MDP will be specialized to the particular type of MDP for random delays. The key insight is that asymptotically, for large n , the initial, stochastic tree is fully expanded, and only the behavior along the deterministic branches is important; $\kappa^*(m_0)$ dominates this behavior.

Theorem 9: In the delayed case, for large n and $\alpha = 1$:
 (a) STOPS near-optimality is $\bullet O(n^{-\frac{\log 1/\gamma}{\log \kappa^*(m_0)}})$ if $\kappa^*(m_0) > 1$, and $\bullet O(\gamma^{cn})$ if $\kappa^*(m_0) = 1$. (b) When called for any x and previous action u^- , which together give an MDP m , STOPS applies a sequence of length: $\bullet d(m) = \Omega(\frac{\log n}{\log \kappa^*(m)})$ if $\kappa^*(m) > 1$, and $\bullet d(m) = \Omega(n)$ if $\kappa^*(m) = 1$. \square

Proof: We start with proving near-optimality (a), as follows. Take a generic MDP m_0 . To achieve near-optimality ε the algorithm only expands nodes in $S_\varepsilon(m_0)$, see (8), so budget $n \leq |S_\varepsilon(m_0)|$. The main part is bounding the cardinality $|S_\varepsilon(m_0)|$ as a function of ε . Then the direct relationship between n and near-optimality follows.

Quantity $|S_\varepsilon(m_0)|$ will be bound by *excluding* nodes from $S_\varepsilon(m_0)$, and then counting all the remaining nodes. Consider the set $S_d(m_0) = \{s \in \mathcal{T}_\infty \mid d(s) \leq d, \exists h_\infty \ni s \text{ s.t. } V_{m_0}^*(\tilde{x}_0) - V_{m_0}^{h_\infty}(\tilde{x}_0) \leq \bar{\delta}(s)\}$. We characterize the impact $\bar{\delta}(s)$ of node s – see again Section V-C for the definition of impact – in the asymptotic regime, along the deterministic branches. All tree policies have the structure in Figure 17. The contribution of a node s_d^j , at depth d on branch j , is $c(s_d^j) = p(j) \frac{\gamma^d}{1-\gamma}$. The maximal-diameter policy on which $c(s_d^j)$ is largest is obtained by picking for any $j' \neq j$ a node s' with $c(s') \leq c(s_d^j)$. Therefore, $\bar{\delta}(s_d^j) \leq Jc(s_d^j) = Jp(j) \frac{\gamma^d}{1-\gamma}, \forall d, j$.

Define for convenience $v_{m_0}(s) := \sup_{h_\infty \ni s} V_{m_0}^{h_\infty}(\tilde{x}_0)$. Choose a node s_d^j , at depth $e = d - J$ on the deterministic subtree $\mathcal{T}(x(s_J))$ of some s_J , that does *not* belong to the near-optimal subtree $\mathcal{T}^*(x(s_J))$. Denote by u_e the deterministic action sequence of s_d^j on $\mathcal{T}(x(s_J))$. Then, $V^*(x(s_J)) - v_{x(s_J)}(u_e) > \frac{\gamma^e}{1-\gamma}$, where V^* and $v_{x(s_J)}$ are values in the

underlying deterministic problem, not in m_0 . Therefore:

$$\begin{aligned} V_{m_0}^*(\tilde{x}_0) - v_{m_0}(s_d^j) &\geq v_{m_0}(s_J) - v_{m_0}(s_d^j) \\ &\geq p(j)\gamma^J [V^*(x(s_J)) - v_{x(s_J)}(\mathbf{u}_e)] > p(j) \frac{\gamma^d}{1-\gamma} \end{aligned}$$

Take another node $s_{d+d'}^j$ below s_d^j ; we know $\bar{\delta}(s_{d+d'}^j) \leq Jp(j) \frac{\gamma^{d+d'}}{1-\gamma}$. By choosing $d' = \left\lceil \frac{\log J}{\log 1/\gamma} \right\rceil$, we have $J\gamma^{d'} \leq 1$ and $\bar{\delta}(s_{d+d'}^j) \leq p(j) \frac{\gamma^d}{1-\gamma} < V_{m_0}^*(\tilde{x}_0) - v_{m_0}(s_d^j)$. Since $v_{m_0}(s_d^j) \geq v_{m_0}(s_{d+d'}^j)$ and $v_{m_0}(s_{d+d'}^j)$ is in turn larger than the value of any policy h_∞ containing $s_{d+d'}^j$, there exists no such policy so that the condition $V_{m_0}^*(\tilde{x}_0) - V_{m_0}^{h_\infty}(\tilde{x}_0) \leq \bar{\delta}(s_{d+d'}^j)$ in the definition of $S_{d+d'}(m_0)$ can be satisfied, and so $s_{d+d'}^j \notin S_{d+d'}(m_0)$.

Henceforth, c_a denotes for any a an appropriately chosen constant whose value is not important to the asymptotic analysis. To bound $|S_d(m_0)|$ for any large d , we count nodes that cannot be excluded as above, up to d . In particular, applying the exclusion rule with a suboptimal node s which is a direct child of a near-optimal one, we find that only nodes up to $d' + 1$ levels below s must be counted. At depth $e = d - J$ in $\mathcal{T}(x(s_J))$, the count of such nodes is denoted $\mu_e(s_J)$ and upper bounded as follows: $\mu_e(s_J) \leq \sum_{i=0}^{d'+1} c_1 \kappa(x(s_J))^{e-i} K^i \leq c_2 \kappa(x(s_J))^e$, when $\kappa(x(s_J)) > 1$, and $\mu_e(s_J) \leq \sum_{i=0}^{d'+1} c_3 K^i = c_4$ when $\kappa(x(s_J)) = 1$.

Thus, $|S_d(m_0)|$ accumulates $\mu_e(s_J)$ for all (finitely many) s_J and any e up to $d - J$, in addition to a constant number c_5 of nodes in the overall tree up to J : $|S_d(m_0)| \leq c_5 + \sum_{s_J} \sum_{e=0}^{d-J} \mu_e(s_J)$. With some calculation, we obtain:

$$|S_d(m_0)| \leq \begin{cases} c_6 \kappa^*(m_0)^d & \text{if } \kappa^*(m_0) > 1 \\ c_7 d & \text{if } \kappa^*(m_0) = 1 \end{cases} \quad (11)$$

Returning now to $S_\varepsilon(m_0)$, note that any nodes up to d in this set belong to $S_d(m_0)$. Then, take D the smallest depth so that $J\bar{p} \frac{\gamma^D}{1-\gamma} \leq \varepsilon$, where $\bar{p} = \max_j p(j)$. For any s at $d > D$, $\bar{\delta}(s) \leq \varepsilon$ and $s \notin S_\varepsilon(m_0)$, so that $S_\varepsilon(m_0) \subseteq S_D(m_0)$. From the condition on D , $D \leq \frac{\log J\bar{p}/[\varepsilon\gamma(1-\gamma)]}{\log 1/\gamma}$, and replacing this in $|S_D(m_0)|$ from (11) we have:

$$|S_\varepsilon(m_0)| \leq \begin{cases} c_8 \varepsilon^{-\frac{\log \kappa^*(m_0)}{\log 1/\gamma}} & \text{if } \kappa^*(m_0) > 1 \\ c_9 \frac{\log 1/\varepsilon}{\log 1/\gamma} & \text{if } \kappa^*(m_0) = 1 \end{cases} \quad (12)$$

According to (8), this means $\beta(m_0) = \frac{\log \kappa^*(m_0)}{\log 1/\gamma}$ and at this point we could directly apply the guarantees of OP-MDP with this value of $\beta(m_0)$. However, we want stronger STOP-like conditions, without the logarithmic term in \tilde{O} . Hence, we continue by recalling a crucial property of OP-MDP [9]: that it only expands nodes in $S_\delta(m_0)$ where δ is the smallest impact among expanded nodes, see Algorithm 4.

Since only nodes in $S_\delta(m_0)$ are expanded, we have $n \leq |S_\delta(m_0)|$, and by using (12) for $\varepsilon = \delta$:

$$\delta \leq \begin{cases} c_{10} n^{-\frac{\log 1/\gamma}{\log \kappa^*(m_0)}} & \text{if } \kappa^*(m_0) > 1 \\ \gamma^{n/c_9} & \text{if } \kappa^*(m_0) = 1 \end{cases} \quad (13)$$

By Theorem 5(i), $\varepsilon(\tilde{x}_0) = V_{m_0}^*(\tilde{x}_0) - \ell_{m_0}(h) \leq \delta$, where h is the initial policy. STOPS will work with different MDPs m

at the leaves of h , and so on recursively, thus generating h_∞ . But since h is fully applied, $V_{m_0}^{h_\infty}(\tilde{x}_0)$ is at least as large as $\ell_{m_0}(h)$, so that $V_{m_0}^*(\tilde{x}_0) - V_{m_0}^{h_\infty}(\tilde{x}_0) \leq \delta$ and the first part of Theorem 9 is proven.

For the length guarantee (b), the shallowest branch of the tree policy h gives the minimal sequence length. For any leaf node s at d , we have $c(s) = p(j) \frac{\gamma^d}{1-\gamma} \leq \text{diam}(h) \leq \delta$. By using (13) and solving for d , we have:

$$d \geq \begin{cases} c_{11} \frac{\log n}{\log \kappa^*(m_0)} & \text{if } \kappa^*(m_0) > 1 \\ c_{12} n & \text{if } \kappa^*(m_0) = 1 \end{cases}$$

where the constants are chosen to cover for all values $p(j)$. Since the derivation holds at any m where STOPS is executed, the proof is complete. \blacksquare

VII. CONCLUSIONS

We have developed a novel approach for the optimal control of general nonlinear NCS, allowing for either time-triggered or self-triggered strategies. The strategies are directly implementable and have guaranteed near-optimality, which is placed in a tight relationship with the transmission intervals and the computation invested. A class of stochastic uncertainties was accommodated, and it was shown how this class can model random network delays in the control channel. These results were obtained by adapting optimistic planning (OP) algorithms from artificial intelligence.

Analyzing the stability of the resulting control is the main priority in future work. This will build on a general stability result for discounted optimal control, which we already achieved in [40]. Stability of COP and STOP further requires dealing with errors coming from quantization and sub-optimality. Dealing with other network effects, such as packet losses, is another interesting direction. Overall, we believe that such syncretic combinations of artificial intelligence and control ideas have a strong future.

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, Hawaii: U.S.A.*, 2012, pp. 7212–7217.
- [2] A. Bemporad, "Predictive control of teleoperated constrained systems with unbounded communication delays," in *Proceedings 37th Conference on Decision and Control*, Tampa, Florida, USA, 16–18 December 1998, pp. 2133–2138.
- [3] D. P. Bertsekas, *Dynamic Programming and Optimal Control*, 3rd ed. Athena Scientific, 2007, vol. 2.
- [4] D. P. Bertsekas and S. E. Shreve, *Stochastic Optimal Control: The Discrete Time Case*. Academic Press, 1978.
- [5] S. Bhatnagar and S. Kumar, "A simultaneous perturbation stochastic approximation-based actor-critic algorithm for markov decision processes," *IEEE Transactions on Automatic Control*, vol. 49, no. 4, pp. 592–598, 2004.
- [6] R. Billinton and R. N. Allan, *Reliability Evaluation of Power Systems*. Springer, 1996.
- [7] M. Branicky, S. Phillips, and W. Zhang, "Scheduling and feedback co-design for networked control systems," in *CDC (IEEE Conference on Decision and Control) Las Vegas, U.S.A.*, 2002, pp. 1211–1217.
- [8] S. Bubeck and R. Munos, "Open loop optimistic planning," in *Proceedings 23rd Annual Conference on Learning Theory (COLT-10)*, Haifa, Israel, 27–29 June 2010, pp. 477–489.

- [9] 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.
- [10] L. Buşoniu, R. Munos, B. De Schutter, and R. Babuška, "Optimistic planning for sparsely stochastic systems," in *Proceedings 2011 IEEE International Symposium on Adaptive Dynamic Programming and Reinforcement Learning (ADPRL-11)*, Paris, France, 11–15 April 2011, pp. 48–55.
- [11] L. Buşoniu, R. Postoyan, and J. Daafouz, "Near-optimal strategies for nonlinear networked control systems using optimistic planning," in *Proceedings American Control Conference 2013 (ACC-13)*, Washington, DC, 17–19 June 2013.
- [12] 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.
- [13] C. G. Cassandras and S. Lafortune, *Introduction to Discrete-Event Systems*. Kluwer, 1999.
- [14] A. Chaillet and A. Bicchi, "Delay compensation in packet-switching networked controlled systems," in *CDC (IEEE Conference on Decision and Control)*, Cancun, Mexico, 2008, pp. 3620–3625.
- [15] H. S. Chang, "A policy improvement method in constrained stochastic dynamic programming," *IEEE Transactions on Automatic Control*, vol. 51, no. 9, pp. 1523–1526, 2006.
- [16] O. Costa, M. Fragoso, and R. Marques, *Discrete-Time Markov Jump Linear Systems*. Springer, 2005.
- [17] C. De Persis and P. Frasca, "Robust self-triggered coordination with ternary controllers," *IEEE Transactions on Automatic Control*, vol. 58, no. 12, pp. 3024–3038, 2013.
- [18] B. De Schutter and B. De Moor, "Optimal traffic light control for a single intersection," *European Journal of Control*, vol. 4, no. 3, pp. 260–276, 1998.
- [19] A. Eqtami, D. Dimarogonas, and K. Kyriakopoulos, "Novel event-triggered strategies for model predictive controllers," in *IEEE Conference on Decision and Control and European Control Conference (CDC-ECC)*, Orlando: U.S.A., 2011, pp. 3392–3397.
- [20] E. Feinberg and A. Shwartz, "Constrained dynamic programming with two discount factors: applications and an algorithm," *IEEE Transactions on Automatic Control*, vol. 44, no. 3, pp. 628–631, 1999.
- [21] 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.
- [22] S. Gelly, Y. Wang, R. Munos, and O. Teytaud, "Modification of UCT with patterns in Monte-Carlo Go," INRIA, Tech. Rep., 2006.
- [23] T. Gommans, D. Antunes, T. Donkers, P. Tabuada, and M. Heemels, "Self-triggered linear quadratic control," *Automatica*, vol. 50, no. 4, pp. 1279–1287, 2014.
- [24] W. Heemels, K. Johansson, and P. Tabuada, "An introduction to event-triggered and self-triggered control," in *IEEE Conference on Decision and Control*, 2012, 2012, pp. 3270–3285.
- [25] E. Henriksson, D. Quevedo, H. Sandberg, and K. Johansson, "Self-triggered model predictive control for network scheduling and control," in *IFAC Symposium on Advanced Control of Chemical Processes*, Singapore, 2012, pp. 432–438.
- [26] J. Hespanha, P. Naghshtabrizi, and Y. Xu, "A survey of recent results in networked control systems," *IEEE Special Issue on Technology of Networked Control Systems*, vol. 95, no. 1, pp. 138–162, 2007.
- [27] J. Hopkins, W.E., "Exponential linear quadratic optimal control with discounting," *Automatic Control, IEEE Transactions on*, vol. 39, no. 1, pp. 175–178, 1994.
- [28] J.-F. Hren and R. Munos, "Optimistic planning of deterministic systems," in *Proceedings 8th European Workshop on Reinforcement Learning (EWRL-08)*, Villeneuve d'Ascq, France, 30 June – 3 July 2008, pp. 151–164.
- [29] 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.
- [30] M. Kaut and S. W. Wallace, "Evaluation of scenario-generation methods for stochastic programming," *Pacific Journal of Optimization*, vol. 3, no. 2, pp. 257–271, 2007.
- [31] B. Kiumarsi, F. L. 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*, 2014, appeared online.
- [32] R. Kleinberg, A. Slivkins, and E. Upfal, "Multi-armed bandits in metric spaces," in *Proceedings 40th Annual ACM Symposium on Theory of Computing (STOC-08)*, Victoria, Canada, 17–20 May 2008, pp. 681–690.
- [33] L. Kocsis and C. Szepesvári, "Bandit based Monte-Carlo planning," in *Proceedings 17th European Conference on Machine Learning (ECML-06)*, Berlin, Germany, 18–22 September 2006, pp. 282–293.
- [34] J. Lasserre, "Detecting planning horizons in deterministic infinite horizon optimal control," *IEEE Transactions on Automatic Control*, vol. 31, no. 1, pp. 70–72, 1986.
- [35] M. Mahmoud, J. Jiang, and Y. Zhang, "Stochastic stability analysis of fault-tolerant control systems in the presence of noise," *IEEE Transaction on Automatic Control*, vol. 46, no. 11, pp. 1810–1815, 2001.
- [36] C. Mansley, A. Weinstein, and M. L. Littman, "Sample-based planning for continuous action Markov decision processes," in *Proceedings 21st International Conference on Automated Planning and Scheduling*, Freiburg, Germany, 11–16 June 2011, pp. 335–338.
- [37] P.-J. Meyer, A. Girard, and E. Witrant, "Controllability and invariance of monotone systems for robust ventilation automation in buildings," in *Proceedings 52nd IEEE Conference on Decision and Control (CDC-13)*, Firenze, Italy, 10–13 December 2013, pp. 1289–1294.
- [38] L. Montestruque and P. Antsaklis, "State and output feedback control in model-based networked control systems," in *Proceedings 41st IEEE Conference on Decision and Control (CDC-2002)*, vol. 2, Las Vegas, US, 10–13 December 2002, pp. 1620–1625.
- [39] 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.
- [40] R. Postoyan, L. Buşoniu, D. Nešić, and J. Daafouz, "Stability of infinite-horizon optimal control with discounted cost," in *53rd Conference on Decision and Control (CDC-14)*, Los Angeles, USA, 15–17 December 2014.
- [41] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, 2nd ed. Wiley, 2012.
- [42] M. L. Puterman, *Markov Decision Processes—Discrete Stochastic Dynamic Programming*. Wiley, 1994.
- [43] D. E. Quevedo and D. Nešić, "Robust stability of packetized predictive control of nonlinear systems with disturbances and Markovian packet losses," *Automatica*, vol. 48, no. 8, pp. 1803–1811, 2012.
- [44] D. E. Quevedo, J. Østergaard, and D. Nešić, "Packetized predictive control of stochastic systems over bit-rate limited channels with packet loss," *IEEE Transactions on Automatic Control*, vol. 56, no. 12, pp. 2854–2868, 2011.
- [45] D. Quevedo and V. Gupta, "Sequence-based anytime control," *IEEE Transactions on Automatic Control*, vol. 58, no. 2, pp. 377–390, 2013.
- [46] Cs. Szepesvári, *Algorithms for Reinforcement Learning*. Morgan & Claypool Publishers, 2010.
- [47] H. van Ekeren, R. Negenborn, P. van Overloop, and B. De Schutter, "Time-instant optimization for hybrid model predictive control of the Rhine-Meuse delta," *Journal of Hydroinformatics*, vol. 15, no. 2, pp. 271–292, 2013.
- [48] R. Yang, G.-P. Liu, P. Shi, C. Thomas, and M. Basin, "Predictive output feedback control for networked control systems," *IEEE Transactions on Industrial Electronics*, vol. 61, no. 1, pp. 512–520, 2014.



Lucian Buşoniu received the M.Sc. degree (valedictorian) from the Technical University of Cluj-Napoca, Romania, in 2003 and the Ph.D. degree (cum laude) from the Delft University of Technology, the Netherlands, in 2009. He is an associate professor with the Department of Automation at the Technical University of Cluj-Napoca, and has previously held research positions in the Netherlands and France. His research interests include planning for nonlinear optimal control, reinforcement learning and approximate dynamic programming, multiagent systems, and robotics. He received the 2009 Andrew P. Sage Award for the best paper in the IEEE Transactions on Systems, Man, and Cybernetics.



Romain Postoyan received the M.Sc. degree in Electrical and Control Engineering from ENSEEIHT (France) in 2005. He obtained the M.Sc. by Research in Control Theory & Application from Coventry University (United Kingdom) in 2006 and the Ph.D. in Control Theory from Université Paris-Sud (France) in 2009. In 2010, he was a research assistant at the University of Melbourne (Australia). Since 2011, he is a CNRS researcher at the Centre de Recherche en Automatique de Nancy (France).



Jamal Daafouz received the Ph.D. degree in automatic control from the INSA Toulouse, in 1997. In 1998, he joined the Institut National Polytechnique de Lorraine (INPL) as an assistant professor and the Research Centre of Automatic Control (CRAN UMR 7039 CNRS). In 2005, he got the French Habilitation degree from INPL and he was engaged as a professor of automatic control at Université de Lorraine in Nancy, France. From 2010 to 2015, he was an IUF (Institut Universitaire de France) junior member. He serves as an Associate Editor at the Conference Editorial Board of the IEEE Control Systems Society and for the journals: Automatica, IEEE Transactions on Automatic Control and European Journal of Control. His research interests include hybrid and switched systems, networked control systems, robust control and applications in secure communications, metallurgy and energy management.