Optimistic minimax search for noncooperative switched control with or without dwell time ⋆

Lucian Busoniu [a], Jihene Ben Rejeb [b], Ioana Lal [a], Irinel-Constantin Morarescu [b], Jamal Daafouz [b]

[a] *Automation Department, Technical University of Cluj-Napoca, Memorandumului 28, 400114 Cluj-Napoca, Romania*
[b] *Université de Lorraine, CRAN, UMR 7039 and CNRS, CRAN, UMR 7039, 2 av. Forêt de Haye, Vandœuvre-lès-Nancy, France*

Abstract

We consider adversarial problems in which two agents control two switching signals, the first agent aiming to maximize a discounted sum of rewards, and the second aiming to minimize it. Both signals may be subject to constraints on the dwell time after a switch. We search the tree of possible mode sequences with an algorithm called *optimistic minimax search with dwell time* (OMSd), showing that it obtains a solution close to the minimax-optimal one, and we characterize the rate at which the suboptimality goes to zero. The analysis is driven by a novel measure of problem complexity, and it is first given in the general dwell-time case, after which it is specialized to the unconstrained case. We exemplify the framework for networked control systems where the minimizer signal is a discrete time delay on the control channel, and we provide extensive simulations and a real-time experiment for nonlinear systems of this type.

Key words: Switched systems; optimistic search; noncooperative optimal control; nonlinear systems.

1 Introduction

Switched systems toggle their dynamics among those in a set of dynamical modes [17,18,30]. They model real-world systems subject to known or unknown abrupt parameter changes, e.g. in the automotive, aerospace, and energy management industries. This paper considers systems with two different switching signals, which occur in e.g. smart grids [25], networks [29] or networked control systems, and have started to be considered in the literature e.g. by [3] where they were called dual switched systems. In our framework, the two signals are controlled by two adversarial agents that choose modes in turn, one of them aiming to maximize an infinite-horizon cumulative reward, and the other to minimize it. Such an infinite horizon is appropriate whenever decisions must be made repeatedly without a definite end of the control task. Either signal (or both) may be subject to dwell time constraints, so that after a switch it must be kept constant for at least an imposed number of steps. Mode dynamics are autonomous, general nonlinear, and the rewards must be bounded.

To find (an approximation of) a minimax-optimal solution, we will use optimistic methods, which combine ideas from global optimization, bandit theory for exploration in reinforcement learning, classical graph search and planning, and optimal control [20]. It turns out that applying optimism in the adversarial setting naturally leads to a best-first variant [21] of B*, a classical minimax algorithm proposed for game tree search by [1]. Since many other methods have been called "best-first search", and as we adapt the algorithm to handle infinite horizons and dwell times, we call it *optimistic minimax search with dwell time* (OMSd), keeping in mind its relation to B*. OMSd iteratively explores a tree representation of the possible sequences of max and min agent modes, by computing lower and upper bounds on the values of mode sequences passing through each node. At each iteration an optimistic leaf is expanded, by starting from the root and recursively traveling to an optimistic child, which maximizes the upper bound at max nodes or minimizes the lower bound at min nodes. OMSd is anytime: it can stop after any number of expansions, and returns the sequence of modes corresponding to a deepest expanded node. Thus it provides an adaptive-horizon solution. For closed-loop control, OMSd may be used in receding horizon.

By exploiting the optimistic framework, we develop near-optimality and convergence rate guarantees for OMSd – which to our best knowledge were missing from the literature on B* search, even in the unconstrained case. The analysis is done in the general case with dwell time constraints, and then specialized to the unconstrained case. The sequence returned by OMSd is near-optimal to the extent of the gap between the upper and lower bounds at the deepest expanded node, which can be computed *a posteriori*, once the algorithm stops. *A priori*, we characterize the rate at which the gap decreases with the computation invested

⋆ We are grateful to Rémi Munos for his contribution to the initial unconstrained method [8], and to Alexandru Codrean for his help with the experiments. This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PN-III-P1-1.1-TE-2016-0670, grant agreement no. 9/2018.

Email address: lucian@busoniu.net (Lucian Busoniu).

(number of node expansions); the rate depends on the dwell time limits of the two signals and on a measure of problem complexity. The measure is quite challenging to define when dwell time constraints are imposed, because the structure of the tree obtained after eliminating nodes that violate the dwell time conditions is quite intricate. It specializes to an intuitive meaning in the unconstrained case: the average branching factor of the subtree explored. Interestingly, in a worst-case sense the constrained algorithm converges faster than the unconstrained one.

Because the min agent chooses its mode using knowledge of the max mode, OMSd solves a Stackelberg problem [4,29], with the max agent as the leader. In fact, the unconstrained framework can be applied to any such problem where the actions of the two agents are discrete or discretized, not just when they are mode switches. Stackelberg adversarial problems appear in network security [31], pursuit-evasion [9], and advertising [10], in addition to game playing [27].

When the two mode choices are applied simultaneously, without the min agent knowing the max mode, a different type of problem ensues where the solution is a Nash equilibrium. In this case, we show that if the same algorithm is used, which still solves the problem as if it were a Stackelberg one, a relationship can nevertheless be established between the value of the sequence returned and the optimal Nash value, see also [16].

In our experiments, the min mode is a discrete time delay on the control communication channel used to send the max mode, see [26,11]. We study cases when an attacker hides behind the delay and controls it in a Stackelberg setting, with or without dwell time, and in a Nash setting solved as Stackelberg. Real-time results are provided for random delays, conservatively treated as an opponent.

In switched systems, OMSd is to our knowledge the first algorithm for optimal control under two switching signals; earlier work by [3] focused on stability, and applied to linear modes only. OMSd instead handles general nonlinear modes, and focuses on near-optimality guarantees – stability is a separate, difficult problem for the discounted costs that we use [23]. In [5], we studied max-only switched problems; the minimax case here is more challenging, and specializes to the max-only case when the min agent is removed.

In planning, besides B* [1,21] OMSd is related to other classical tree search methods such as alpha-beta pruning [14] or best-first search [15]. OMSd belongs to the optimistic class, which provides algorithms for e.g. optimization [19], optimal control of discrete-action deterministic [12] or stochastic systems [6]. A crucial feature of the analysis is the complexity measure, which is related to other measures in optimistic methods, such as the branching factor in [12], or the near-optimality dimension in [19]. Different from these however, our measure works in minimax problems. In the unconstrained case, it specializes to a branching factor also related to the branching factor of alpha-beta pruning [14,22]. However, OMSd is adaptive-depth (horizon) while alpha-beta is fixed-depth. OMSd is closer to the adaptive-depth best-first method of [15], which does not have an analysis and in fact may converge to suboptimal solutions [8].

This paper integrates and extends the earlier work [8] on unconstrained OMS, and [24] for dwell time constraints. Notable novel elements here include: (i) handling different dwell times for the two agents (in [24] they had to be the same, which is unrealistic); (ii) the study of the solution properties in the Nash case, which is important since often an agent does not have access to its opponent's action at the current step, (iii) a different and extensive experimental study, including a real-time illustration of the technique, and (iv) an integrated presentation of the framework that we show encompasses not just the unconstrained and dwell-time minimax cases, but also the single-agent case with or without dwell time [5].

Next, Sec. 2 formally states the problem, Sec. 3 gives the algorithm and Sec. 4 analyzes it. Experimental results are given in Sec. 5; Sec. 6 concludes.

2 Problem definition

2.1 Unconstrained case

Consider a switched system where a maximizer (max) agent and a minimizer (min) agent control, in turn, two different switching signals. The setting is noncooperative, i.e. the min agent acts against the max agent: while the max agent aims to maximize a cumulative reward signal, the min agent acts so as to minimize it. The max and min signals are respectively denoted u and w; there are N_u and N_w modes for these signals, and the sets of modes are denoted U and W. In our algorithm and analysis, it will often be useful to treat max and min decisions uniformly. For that purpose, denote a generic (either max or min) mode by $z \in Z := U \cup W$; and define also index h that counts all decision steps, max or min. Thus, h advances twice as fast as k, which only increases with pairs of max-min decisions. For example, we have: $z_0 = u_0, z_1 = w_0, z_2 = u_1, z_3 = w_1, \dots, z_{2k} = u_k, z_{2k+1} = w_k, \dots$. Denote an infinite sequence of modes by $\mathbf{z}_\infty = (z_0, z_1, z_2, z_3, \dots, z_{2k}, z_{2k+1}, \dots) = (u_0, w_0, u_1, w_1, \dots, u_k, w_k, \dots) \in Z^\infty$. A finite sequence of h modes is denoted $\mathbf{z}_h = (z_0, z_1, \dots, z_{h-1})$, with $\mathbf{z}_0$ the empty sequence by convention.

At each step $h \in \mathbb{N}$, the system evolves according to:

$$x_{h+1} = f(x_h, z_h) \tag{1}$$

where $x_h \in X$ is the state, $z_h \in Z$ is the (max or min) mode, and $f : X \times Z \to X$ are the (autonomous) mode dynamics. We prefer the notation $f(x, z)$ instead of the more standard $f_z(x)$ as it avoids double subscripts. A reward (negative cost) $\rho(x_h, z_h)$ is assigned, where $\rho : X \times Z \to \mathbb{R}$. Given an initial state x_0, the infinite-horizon discounted value of sequence $\mathbf{z}_\infty$ is:

$$v(\mathbf{z}_\infty) := \sum_{h=0}^{\infty} \gamma^h \rho(x_h, z_h) \tag{2}$$

where $\gamma \in (0,1)$ is the discount factor. The goal is to achieve the minimax-optimal value, defined as:

$$v^* := \lim_{k\to\infty} \left[\max_{u_0} \min_{w_0} \cdots \max_{u_k} \min_{w_k} \sum_{h=0}^{2k-1} \gamma^h \rho(x_h, z_h) \right] \tag{3}$$

Intuitively, this value is the best achievable by the max agent under the worst-case assumption that the min agent always chooses the mode that is the most detrimental for performance. Very general technical conditions under which the solution exists can be found in [4]. Note that the method we propose later will not reach this minimax value exactly or find a complete, infinite-horizon sequence; it will only get closer to the infinite-horizon value and sequence as the computation budget increases, by examining increasingly longer (but still finite) mode sequences.

Assumption 1 *The rewards $\rho(x,z)$ are in $[0,1]$ for all $x \in X, z \in Z$.*

The main role of discounting and cost boundedness is to ensure that the value (2) is in $[0, \frac{1}{1-\gamma}]$ for any sequence. This implies that for any finite sequence $\mathbf{z}_h$, $l(\mathbf{z}_h) := \sum_{j=0}^{h-1} \gamma^j \rho(x_j, z_j)$ is a lower bound on the values of all sequences $\mathbf{z}_\infty$ starting with $\mathbf{z}_h$ (with the convention that an empty sum is 0), and $b(\mathbf{z}_h) := l(\mathbf{z}_h) + \frac{\gamma^h}{1-\gamma}$ is an upper bound. Thus, $v(\mathbf{z}_\infty) \in [l(\mathbf{z}_h), b(\mathbf{z}_h)]$. Let $\delta(h) = \frac{\gamma^h}{1-\gamma}$ denote the *gap* between the two bounds, an uncertainty on the values of sequences $\mathbf{z}_\infty$ starting with $\mathbf{z}_h$. These value bounds are essential to the derivation of our algorithm below. Thus, at the cost of some restrictiveness induced by the bounded rewards and by the subunitary discount factor, we obtain a method that works for general nonlinear dynamics and nonquadratic rewards.

Note moreover that many other works in control use discounting, e.g. [13]. Bounded rewards are typical in AI methods for optimal control, such as reinforcement learning [28]. One way to achieve boundedness is by saturating a possibly unbounded original function. This changes the optimal solution, but is often sufficient in practice. On the other hand, the system may have physical limitations that naturally lead to saturation limits and a corresponding bound.

Often, the max and min mode decisions are applied simultaneously, so the dynamics and rewards are:

$$y_{k+1} = g(y_k, u_k, w_k), \quad r_{k+1} = r(y_k, u_k, w_k) \tag{4}$$

where $y_k \in Y$ is the state signal of the simultaneous-decision problem, while g and r denote the dynamics and reward function of this problem (and r_k is a particular reward at step k). The infinite-horizon value to optimize is $\sum_{k=0}^{\infty} \beta^k r(y_k, u_k, w_k)$, with $\beta \in (0,1)$ a new discount factor. Since the system can be in any combination of modes u and w, the total number of modes is $N_u \cdot N_w$.

We will represent this problem in the turn-based form (1)-(3). To this end, we define the turn-based state vector $x \in Y \times \{U \cup \{s\}\}$ so that in addition to the state of the simultaneous-decision problem, it also contains an extra state variable. This variable takes special value $s \notin U$ at max steps, and at min steps remembers the latest max mode. Recalling that index h advances twice as fast as k, we have formally that at (even, max) steps $h = 2k$, $x_h = x_{2k} = [y_k^\top, s]^\top$, while at (odd, min) steps $h = 2k+1$, $x_h = x_{2k+1} = [y_k^\top, u_k]^\top$. Using turn-based state x, g is represented by the following turn-based dynamics f in (1):

$$f(x_h, z_h) = \begin{cases} [y_k^\top, u_k]^\top & \text{if } h = 2k \\ [g(y_k, u_k, w_k)^\top, s]^\top & \text{if } h = 2k+1 \end{cases}$$

Rewards are similarly represented:

$$\rho(x_h, z_h) = \begin{cases} 0 & \text{if } h = 2k \\ r(y_k, u_k, w_k) & \text{if } h = 2k+1 \end{cases}$$

Note that f and r are time-invariant, so they cannot directly check whether h is even; however, this can be determined by examining the last, special state variable, which equals s at even steps. We have $\sum_{h=0}^{\infty} \gamma^h \rho(x_h, z_h) = \gamma \sum_{k=0}^{\infty} \gamma^{2k} r(y_k, u_k, w_k)$, so to optimize the intended objective function (2) with discount factor β, we take $\gamma = \sqrt{\beta}$.

While here we focus on switched systems, the framework can in fact be applied to any adversarial decision-making problem where the actions of the two agents are discrete or discretized. One reason for which we prefer the turn-based formalism (1)-(3) is that in this general context, it covers additional problems like two-player games.

2.2 Dwell time constraints

In switched systems, modes must often be kept constant over a minimum *dwell time*, in order to guarantee fundamental stability or performance properties, to obey actuation constraints, etc. In particular, switching signals u and w have minimum dwell-time limits $\underline{d}_u$ and $\underline{d}_w$. For the max agent the dwell-time is defined as the number of max decision steps during which the mode u remains constant after a change, and the condition requires that all dwell times along the sequence are at least as large as $\underline{d}_u$, see Fig. 1 for an example. The situation is similar for the min signal w. Dwell times therefore only increase once every two steps h (corresponding to one step k). Taking a limit equal to 1 is equivalent to not imposing a dwell-time condition for that signal, so the unconstrained problem is recovered when $\underline{d}_u = \underline{d}_w = 1$.

The objective (3) changes to reflect the constrained nature of the switching sequences. Denote by $U(\mathbf{z}_h)$ and $W(\mathbf{z}_h)$ respectively the set of all max and min modes at step h that satisfy the dwell-time constraints given prior modes $\mathbf{z}_h$. E.g., $U(\mathbf{z}_h) = U$ when $\mathbf{z}_h$ already satisfies the max dwell time condition at h, and otherwise $U(\mathbf{z}_h)$ is equal to the last max mode along sequence $\mathbf{z}_h$. Then, the constrained minimax

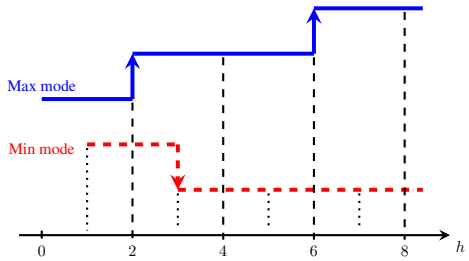

Fig. 1. Illustration of a constrained minimax sequence for $\underline{d}_u = \underline{d}_w = 2$. The max and min modes applied are shown by a blue continuous line and a red dashed line, respectively. Denote e.g. the max (blue) modes by 1, 2, 3 from bottom to top. At $k = 1$ ($h = 2k = 2$), the max agent has switched from mode 1 to mode 2; since its dwell time limit is 2, it must wait until at least $k = 1 + 2 = 3$ (equivalently, until $h = 6$) before it is allowed to select a new mode, different from 2. Here, it selects mode 3. In contrast, had the problem been unconstrained, the max mode could have already been changed at $k = 2$.

value to be achieved is:

$$v^* := \lim_{k\to\infty} \Big[\max_{u_0\in U(\mathbf{z}_0)} \min_{w_0\in W(\mathbf{z}_1)} \cdots \max_{u_k\in U(\mathbf{z}_{2k})} \min_{w_k\in W(\mathbf{z}_{2k+1})} \sum_{h=0}^{2k} \gamma^h \rho(x_h, z_h) \Big] \quad (5)$$

2.3 Nash problem

In both the unconstrained and constrained problems above, when it chooses w_k the min agent knows and can react to the max action u_k. This is a Stackelberg, leader-follower setting with the max agent as the leader, and it useful in many practical problems such as network security [31,29], pursuit-evasion problems [9], or advertising [10]. This setting was implicitly assumed throughout [8,24]. However, if the two agents act simultaneously (4) and the min agent does *not* have access to u_k at step k, then the Stackelberg solution concept is no longer appropriate, and the Nash equilibrium is usually employed (also called a saddle-point solution in the adversarial setting here). Nevertheless, as we show next, the problem can still be solved as if it were a Stackelberg one, which under certain conditions provides a bound on the Nash value – see also [16]. This bound will later be coupled with the near-optimality guarantees of our algorithm to relate the sequence it returns with the Nash value. For simplicity, we stick to unconstrained sequences in the Nash setting.

First, it will help to make the infinite-horizon Stackelberg solution more explicit. For $t \geq 0$ define the sequence of value functions $v_t : Y \to \mathbb{R}$, $\tilde{v}_t : Y \times U \times W \to \mathbb{R}$:

$$\begin{aligned} \tilde{v}_t(y,u,w) &= r(y,u,w) + \beta v_{t-1}(g(y,u,w)) \\ v_t(y) &= \tilde{v}_t(y, u_t(y), w_t(y,u_t(y))) \end{aligned} \quad (6)$$

starting from the initial function $v_0(y) = 0$. Function equalities and inequalities hold elementwise. Here, $w_t(y,u) \in \arg\min_w \tilde{v}_t(y,u,w)$ is a reaction function of the min agent to max decisions u; if there are multiple minimizer actions, a selection rule among them must be imposed. Moreover, $u_t(y) \in \arg\max_w \tilde{v}_t(y,u,w_t(y,u))$ is a Stackelberg-optimal decision of the max agent given the reaction function w_t. Then, the infinite-horizon Stackelberg value from initial state y is $v^*(y) = \lim_{t\to\infty} v_t(y)$, and is the same as (3).

In contrast, a pure (deterministic) Nash solution is defined as follows, using $v_t^{\mathrm{N}} : Y \to \mathbb{R}$, $\tilde{v}_t^{\mathrm{N}} : Y \times U \times W \to \mathbb{R}$:

$$\begin{aligned} \tilde{v}_t^{\mathrm{N}}(y,u,w) &= r(y,u,w) + \beta v_{t-1}^{\mathrm{N}}(g(y,u,w)) \\ v_t^{\mathrm{N}}(y) &= \tilde{v}_t^{\mathrm{N}}(y, u_t^{\mathrm{N}}(y), w_t^{\mathrm{N}}(y)) \end{aligned} \quad (7)$$

where $v_0^{\mathrm{N}}(y) = 0$ and $u_t^{\mathrm{N}}(y), w_t^{\mathrm{N}}(y)$ is a Nash equilibrium at stage t in state y, i.e. a pair of decisions such that $\tilde{v}_t^{\mathrm{N}}(y,u,w_t^{\mathrm{N}}(y)) \geq \tilde{v}_t^{\mathrm{N}}(y,u_t^{\mathrm{N}}(y),w_t^{\mathrm{N}}(y)) \geq \tilde{v}_t^{\mathrm{N}}(y,u_t^{\mathrm{N}}(y),w)$, $\forall u, w$. The infinite-horizon Nash value is then $v^{*\mathrm{N}}(y) = \lim_{t\to\infty} v_t^{\mathrm{N}}(y)$. Note that this definition requires the existence of a deterministic Nash equilibrium at each t, y, and if there are several such equilibria, a selection rule must be imposed to have a well-defined sequence.

Proposition 2 *The infinite-horizon Stackelberg value is larger than the Nash one:* $v^* \geq v^{*\mathrm{N}}$.

Proof: We prove the statement by induction, starting from $v_0 = v_0^{\mathrm{N}} = 0$. Take $t \geq 1$ and assume $v_{t-1} \geq v_{t-1}^{\mathrm{N}}$. Then, for any y:

$$\begin{aligned} v_t(y) &= \tilde{v}_t(y, u_t(y), w_t(y, u_t(y))) \\ &\geq \tilde{v}_t(y, u_t^{\mathrm{N}}(y), w_t(y, u_t^{\mathrm{N}}(y))) \\ &\geq \tilde{v}_t^{\mathrm{N}}(y, u_t^{\mathrm{N}}(y), w_t(y, u_t^{\mathrm{N}}(y))) \\ &\geq \tilde{v}_t^{\mathrm{N}}(y, u_t^{\mathrm{N}}(y), w_t^{\mathrm{N}}(y)) = v_t^{\mathrm{N}}(y) \end{aligned}$$

where the first inequality is true because the Stackelberg max decision $u_t(y)$ maximizes $\tilde{v}_t(y,u,w_t(y,u))$ over u; the second because $\tilde{v}_{t-1}(y,x,u) \geq \tilde{v}_{t-1}^{\mathrm{N}}(y,u,w), \forall y,u,w$, which immediately follows from $v_{t-1}(y) \geq v_{t-1}^{\mathrm{N}}(y)$; and the third inequality holds because the Nash min decision $w_t^{\mathrm{N}}(y)$ minimizes $\tilde{v}_t^{\mathrm{N}}(y, u_t^{\mathrm{N}}(y), w)$ over w. ■

3 Algorithm

We will present the optimistic minimax search (OMSd) algorithm in the dwell-time case from Sec. 2.2, which will reduce naturally to the unconstrained problem of Sec. 2.1 when $\underline{d}_u = \underline{d}_w = 1$.

OMSd explores a tree representation of the possible sequences of max and min modes. It starts with a root node corresponding to the empty sequence, and iteratively expands n nodes taking into account dwell-time conditions. Fig. 2 illustrates, with squares representing max decision nodes, and disks min decision nodes. Each node is labeled by two dwell times, for max and min switches, separated by a slash in the figure. Note that by convention both dwell time conditions are taken satisfied at $h = 0$, so the root node in the figure has dwell times $\underline{d}_u/\underline{d}_w$, namely $2/2$. A max decision node is expanded by adding children corresponding

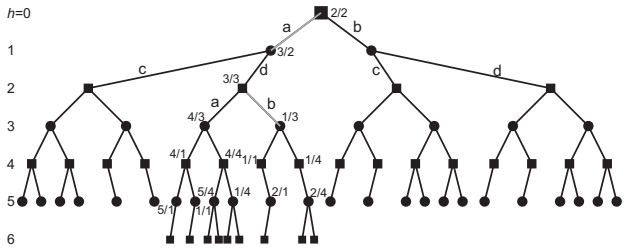

Fig. 2. Example of a minimax tree developed by the algorithm from a max root. The max agent has modes 'a' and 'b', while the min agent has modes 'c' and 'd', so that $N_u = N_w = 2$. The dwell time limits are taken $\underline{d}_u = \underline{d}_w = 2$.

to max modes, and similarly for min decision nodes. Each arc is labeled by the mode taken at the parent node to reach the child. Specifically, at max nodes, if the max dwell-time is at least $\underline{d}_u$ then N_u children nodes are created, one for every max mode; otherwise, i.e. if the max dwell-time condition is not satisfied, only the child that keeps the mode constant is added. Similarly, N_w children nodes are added at a min node if its min dwell-time is at least $\underline{d}_w$, and only the constant-mode child is added otherwise. For example, the node labeled 1/3 in the figure has max dwell time 1 because the max mode taken to reach it, 'b', is different from the previous max mode 'a' taken two levels higher (at the root), so a max switch just occurred. These two different modes are highlighted by gray arcs. Note that this particular node is not immediately affected by the non-satisfaction of the max dwell time, since it is a min decision node; indeed, both its children are created since the min dwell time is still 3, and the constraint only has an effect at the *next* depth, where the only allowed max mode is 'b'. Fig. 2 also illustrates some constrained min node expansions, from depth 5 to 6.

Each node at some depth h is reached via a unique path through the tree, and so is uniquely associated to the sequence of modes $\mathbf{z}_h$ on this path. We denote by $d_u(\mathbf{z}_h)$ and $d_w(\mathbf{z}_h)$ the current max and min dwell-times of $\mathbf{z}_h$. We will work interchangeably with sequences and nodes.

Let $\mathcal{T}$ denote the current tree, $\mathcal{L}(\mathcal{T})$ the leaf nodes of this tree, and $\mathcal{C}(\mathbf{z})$ the children of node $\mathbf{z}$, all satisfying the dwell-time constraints (so $\mathcal{C}(\mathbf{z})$ is sometimes a singleton). The algorithm computes lower and upper bounds $L(\mathbf{z})$ and $B(\mathbf{z})$ for each node. The bounds are initialized with l and b at the leaves and propagated upwards in the tree:

$$L(\mathbf{z}) = \begin{cases} l(\mathbf{z}), & \text{if } \mathbf{z} \in \mathcal{L}(\mathcal{T}) \\ \max_{\mathbf{z}' \in \mathcal{C}(\mathbf{z})} L(\mathbf{z}'), & \text{if } \mathbf{z} \text{ max node, } \mathbf{z} \notin \mathcal{L}(\mathcal{T}) \\ \min_{\mathbf{z}' \in \mathcal{C}(\mathbf{z})} L(\mathbf{z}'), & \text{if } \mathbf{z} \text{ min node, } \mathbf{z} \notin \mathcal{L}(\mathcal{T}) \end{cases}$$

$$B(\mathbf{z}) = \begin{cases} b(\mathbf{z}), & \text{if } \mathbf{z} \in \mathcal{L}(\mathcal{T}) \\ \max_{\mathbf{z}' \in \mathcal{C}(\mathbf{z})} B(\mathbf{z}'), & \text{if } \mathbf{z} \text{ max node, } \mathbf{z} \notin \mathcal{L}(\mathcal{T}) \\ \min_{\mathbf{z}' \in \mathcal{C}(\mathbf{z})} B(\mathbf{z}'), & \text{if } \mathbf{z} \text{ min node, } \mathbf{z} \notin \mathcal{L}(\mathcal{T}) \end{cases} \quad (8)$$

To better understand, consider a subtree developed by the algorithm after 4 node expansions, shown in Fig. 3 (a subtree of the tree in Fig. 2). This figure shows the bounds associated with each node, initialized to illustrative values at the leaves

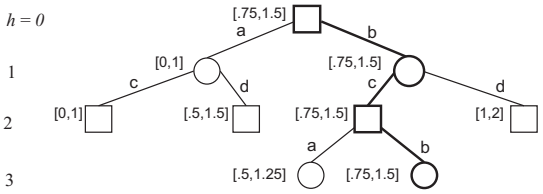

Fig. 3. Snapshot of the tree after 4 expansions. Nodes are now labeled by the interval $[L, B]$, and dwell time labels are skipped for clarity. The thick path leads to the node that the algorithm would expand next.

Algorithm 1 OMSd

Input: budget n

initialize: $\mathcal{T} \leftarrow \{\mathbf{z}_0\}$, the root
for iteration $t = 1$ to n **do**
  $\mathbf{z} \leftarrow \mathbf{z}_0$
  while $\mathbf{z} \notin \mathcal{L}(\mathcal{T})$ **do**
    $\mathbf{z} \leftarrow \begin{cases} \arg\max_{\mathbf{z}' \in \mathcal{C}(\mathbf{z})} B(\mathbf{z}'), & \text{if } \mathbf{z} \text{ max node} \\ \arg\min_{\mathbf{z}' \in \mathcal{C}(\mathbf{z})} L(\mathbf{z}'), & \text{if } \mathbf{z} \text{ min node} \end{cases}$
  $\mathbf{z}(t) \leftarrow \mathbf{z}$
  expand $\mathbf{z}(t)$, by adding its children to $\mathcal{T}$:
  if $\mathbf{z}(t)$ max node **then**
    if $d_u(\mathbf{z}(t)) \geq \underline{d}_u$, add $(\mathbf{z}(t), u)\, \forall u \in U$
    else, add one child keeping u constant
  else
    if $d_w(\mathbf{z}(t)) \geq \underline{d}_w$, add $(\mathbf{z}(t), w)\, \forall w \in W$
    else, add one child keeping w constant
  compute bounds for all $\mathbf{z} \in \mathcal{T}$ with (8)

Output: $\hat{\mathbf{z}} := \arg\max_{\mathbf{z}(t), t=1,\dots,n} h(\mathbf{z})$, $l(\hat{\mathbf{z}})$, $b(\hat{\mathbf{z}})$

and propagated with (8). For example, since the root $\mathbf{z}_0$ is a max node, its lower bound $L(\mathbf{z}_0) = 0.75$ is found as the maximum of the two children's bounds, 0 and 0.75; and similarly for the upper bound $B(\mathbf{z}_0)$.

To choose the leaf to expand at each iteration, OMSd starts from the root and constructs a path by iteratively selecting an optimistic child of the current node, where optimistic means that the child is one with the largest upper bound at max nodes, and one with the smallest lower bound at min nodes. For instance, in Fig. 3, starting from the root first the right child is selected, because the root is a max node and the right child has the largest upper bound, namely 1.5; then at this next (min) node, its left child is selected, since it has the smallest lower bound, 0.75, and so on along the thick path until a leaf is reached, which is then expanded.

After n node expansions, the algorithm stops and returns the sequence $\hat{\mathbf{z}}$ and the bounds of a deepest expanded node. Alg. 1 summarizes OMSd, where $(\cdot, \cdot)$ denotes sequence concatenation and $h(\cdot)$ yields the depth of a sequence.

When $\underline{d}_u = \underline{d}_w = 1$, since any mode is kept for at least one step, the dwell time of any node is at least 1, so the inner "else" branches never get activated, children for all modes are always created, and the algorithm solves the unconstrained problem of Sec. 2.1. The algorithm is then an infinite-horizon extension of B* search [1].

OMSd will often be used to find max modes to apply. The algorithm should then typically applied in receding horizon, calling it with the current state at max decision steps where the dwell time condition is satisfied. If the condition is not satisfied, then the max mode must be kept constant anyway so it is not useful to run OMSd. To exemplify, assume that the minimax switching sequence in Figure 1 is obtained by such a closed-loop application of OMSd. The algorithm is first called at $h = k = 0$, resulting in the first max mode, which is applied, and the min agent generates its own mode. OMSd is next called at $h = 2$, corresponding to $k = 1$, at which time it generates a different mode, and again the min agent responds. Now, since a max switch has occurred, the max mode must be kept constant for the next step, and OMSd is only called again at $h = 6$, or $k = 3$; and so on. Of course, when there is no dwell time constraint the algorithm must be called at each step. Min switches must also obey their own dwell time, and OMSd takes advantage of this information by skipping sequences that do not satisfy the min constraint. Note that the min agent can in general use any algorithm, including OMSd.

4 Analysis

4.1 Constrained case

Like the algorithm itself, the analysis will first be given in the case with dwell time constraints, and then specialized to the unconstrained case. We begin by establishing an a posteriori near-optimality bound with respect to the minimax value. Then, a complexity measure of the problem and a corresponding convergence rate of OMSd are established. In [24] the two dwell times $\underline{d}_u$, $\underline{d}_w$ had to be equal for the analysis to work. Since this is unrealistic in most cases, here we extend the analysis to different dwell times, which impacts everything from the complexity measure onward.

Define for any node $\mathbf{z}_h$ of finite depth h the minimax value $v(\mathbf{z}_h)$ among infinite sequences starting with $\mathbf{z}_h$ and obeying the dwell time constraints. Specifically:

$$v(\mathbf{z}_h) = \sum_{j=0}^{h-1} \gamma^j \rho(x_j, z_j) + \begin{cases} \max\limits_{z_h \in U(\mathbf{z}_h)} \min\limits_{z_{h+1} \in W(\mathbf{z}_{h+1})} \cdots \sum\limits_{j=h}^{\infty} \gamma^j \rho(x_h, z_h) \text{ if } \mathbf{z}_h \text{ max node} \\ \min\limits_{z_h \in W(\mathbf{z}_h)} \max\limits_{z_{h+1} \in U(\mathbf{z}_{h+1})} \cdots \sum\limits_{j=h}^{\infty} \gamma^j \rho(x_h, z_h) \text{ if } \mathbf{z}_h \text{ min node} \end{cases} \tag{9}$$

The next Lemma characterizes the subset of nodes expanded by the algorithm, which is in general not the full tree.

Lemma 3 (Lemma 3 of [24]) *At depth h in the tree, OMSd only expands nodes in the set:*

$$\mathcal{T}_h^* := \big\{ \mathbf{z}_h \;\big|\; |v^* - v(\mathbf{z}_p)| \le \delta(h), \; \forall \mathbf{z}_p \text{ on path from root to } \mathbf{z}_h \big\} \tag{10}$$

An important feature of $\mathcal{T}^*$ is the nonlocal character of the inequality in (10), which must hold for any parent and not just the expanded node.

We give now an *a posteriori* near-optimality bound, which can be computed after the algorithm stops.

Theorem 4 (Theorem 4 of [24]) *Let h^* be the largest depth of any expanded node. Then, $|v^* - v(\hat{\mathbf{z}})| \le \delta(h^*)$ and $v^* \in [L(\mathbf{z}_0), B(\mathbf{z}_0)]$.*

The analysis given so far was known from [24]. However, since unlike in [24], here $\underline{d}_u$ may be different from $\underline{d}_w$, the upcoming analysis must be updated to take this into account. To that end, let $d = \min\{\underline{d}_u, \underline{d}_w\}$ and $N = \max\{N_u, N_w\}$. To establish an *a priori* bound, we must first characterize the size of the expanded subtree $\mathcal{T}^* = \bigcup_{h \ge 0} \mathcal{T}_h^*$ via a complexity measure κ that evaluates the difficulty of the search problem. In the sequel, $|\cdot|$ denotes set cardinality.

Definition 5 *Let κ be a number so that $\exists C > 0, |\mathcal{T}_h^*| \le C\kappa^{h/d}, \forall h \ge 0$.*

Note that κ is similar with other complexity measures used in optimistic optimization and planning [20], such as the branching factor in [12], the near-optimality dimension in [19], and the near-optimality exponent in [6]. Before the final bound, we study the range of κ, starting with the largest value (the most difficult problem).

Proposition 6 *A finite κ always exists, and $\kappa \le dN$.*

Proof: It suffices to consider the case when the full tree $\mathcal{T}$ is expanded, since then $\mathcal{T}^* = \mathcal{T}$, which leads to the largest number of nodes in $\mathcal{T}_h^*$ and hence the largest κ. This occurs in a problem where all the rewards are identical. Any sequence is optimal in this case and the algorithm must explore the entire tree uniformly.

First, denote by $\overline{\mathcal{T}}$ the tree of sequences with equal dwell times for both max and min, $\underline{d}'_u = \underline{d}'_w = d$. We have $\mathcal{T} \subseteq \overline{\mathcal{T}}$. To prove this, consider any node $\mathbf{z} \in \mathcal{T}$. Since $\mathbf{z}$ satisfies the original dwell time constraint, $d_u(\mathbf{z}) \ge \underline{d}_u \ge d = \underline{d}'_u$, and $\mathbf{z}$ also satisfies the looser constraint from $\overline{\mathcal{T}}$. A similar reasoning holds for the min agent, so $\mathbf{z} \in \overline{\mathcal{T}}$.

Next, define the matrix $M_h \in \mathbb{R}^{d \times d}$, the elements of which count nodes with various max and min dwell times at depth h in $\overline{\mathcal{T}}$:

$$M_{h,ii'} = \begin{cases} \#(d_u = i,\ d_w = i'), & \text{if } i < d,\ i' < d \\ \#(d_u \ge i,\ d_w = i'), & \text{if } i = d,\ i' < d \\ \#(d_u = i,\ d_w \ge i'), & \text{if } i < d,\ i' = d \\ \#(d_u \ge i,\ d_w \ge i'), & \text{if } i = d,\ i' = d \end{cases}$$

where $\#(\cdot)$ is shorthand for "the number of nodes at h satisfying condition $\cdot$". So, the non-terminal rows $i < d$ count the number of nodes with a max dwell-time of exactly i, while the last row is special: it counts the nodes with a max dwell-time of d *or larger*. The situation is similar for the

columns, which correspond to the min dwell time. Then:

$$|\overline{\mathcal{T}}_h| = \sum_{i=1}^{d}\sum_{i'=1}^{d} M_{h,ii'}$$

Matrix M_h satisfies at all depths of the form $2jd$, $j \in \mathbb{N}$:

$$M_{2jd} \leq \begin{bmatrix} (dN)^{2j}(N-1) & \dots & (dN)^{2j}(N-1) & (dN)^{2j}(N-1) \\ \vdots & & \vdots & \vdots \\ \vdots & & \vdots & (dN)^{2j}(N-1) \\ (dN)^{2j}(N-1) & \dots & (dN)^{2j}(N-1) & (dN)^{2j} \end{bmatrix} =: \overline{M}_{2jd} \tag{11}$$

with matrix inequalities holding elementwise. The proof of this property is very technical and we do not include it here, instead referring the reader to the supplementary material at: `busoniu.net/files/papers/swminmax_suppl.pdf`. Take $j = \left\lceil \frac{h}{2d} \right\rceil$. We have $\sum_{i,i'} M_{h,ii'} \leq \sum_{i,i'} M_{2jd,ii'}$, as the number of explored nodes increases with the depth, and on the other hand $M_{2jd,ii'} \leq \overline{M}_{2jd,ii'}$ from (11). Then:

$$\begin{aligned} |\overline{\mathcal{T}}_h| &\leq \sum_{i=1}^{d}\sum_{i'=1}^{d}(dN)^{2j}(N-1) \\ &\leq d^3 N (dN)^{h/d}(N-1) = C(dN)^{\frac{h}{d}} \end{aligned}$$

with $C = d^3N(N-1)$, and since $\mathcal{T}_h^* = \mathcal{T}_h \subseteq \overline{\mathcal{T}}_h$, finally $\kappa \leq dN$. ■

Regarding the smallest value of κ, it is trivially 1, since there must be at least one node in $\mathcal{T}_h^*$: the one belonging to an optimal sequence.

Example 7 *A problem with $\kappa = 1$.* Consider the problem represented by the tree in Fig. 4. For each max node along the path on the left of the tree, one child has reward 1 and all other children have reward 0, and the same is true of their complete subtrees. The situation is reversed at min nodes. Thus, the leftmost path is minimax-optimal, with value $v^* = \gamma^0 + \gamma^2 + \ldots = \frac{1}{1-\gamma^2}$. This path is constructed such that it obeys the dwell time constraints.

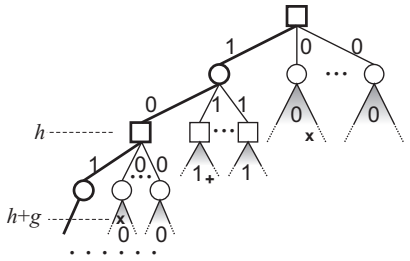

Fig. 4. A tree with $\kappa = 1$. Rewards are shown along the transitions and inside subtrees where they remain constant. The thick path is minimax-optimal and never switches.

To study $\mathcal{T}^*$, consider an arbitrary node $\mathbf{z}_{h+g}$ at depth $h+g$ that is *not* on the optimal path, but does belong to the subtree of some max node $\mathbf{z}_h$ which is on this path at an even depth h. Two examples of such nodes are shown by 'x' symbols in the figure. Then, the value of $\mathbf{z}_{h+g}$ is $v(\mathbf{z}_{h+g}) = \gamma^0 + \gamma^2 + \ldots + \gamma^{h-2} = \frac{1-\gamma^h}{1-\gamma^2}$. Since $\delta(h+g) = \frac{\gamma^{g+h}}{1-\gamma}$, node $\mathbf{z}_{h+g}$ can be excluded when: $|v^* - v(\mathbf{z}_{h+g})| > \frac{\gamma^{g+h}}{1-\gamma}$, i.e. when $\frac{\gamma^h}{1-\gamma^2} > \frac{\gamma^{g+h}}{1-\gamma}$, for which it suffices if $g > \left\lceil \frac{\log(1-\gamma^2)/(1-\gamma)}{\log 1/\gamma} \right\rceil$, a positive constant which we call G. Similarly, take now a node at depth $h+g$ on a non-optimal subtree of a *min* node at an odd h, as exemplified by a '+' in the figure. The value of such a node is $\gamma^0 + \gamma^2 + \ldots + \gamma^{h-1} + \gamma^h + \gamma^{h+1} + \gamma^{h+2} + \ldots = \gamma^0 + \gamma^2 + \ldots + \gamma^{h-1} + \gamma^{h+1} + \gamma^{h+3} + \ldots + \gamma^h + \gamma^{h+2} + \ldots = \frac{1+\gamma^h}{1-\gamma^2}$ where the intermediate step separated odd and even powers of γ at depth h and larger. Solving the exclusion condition results in the same lower bound G on g as for max nodes.

At some arbitrary depth h' the set $\mathcal{T}_{h'}^*$ contains *at most* $1 + N + N^2 + \ldots + N^G =: C$ children coming from optimal nodes at various depths $h < h'$, which cannot be excluded via the conditions above. Thus, $\mathcal{T}_{h'}^* \leq C$ and $\kappa = 1$. □

Finally, we give the *a priori* near-optimality result, in which the rates have the same mathematical form as in Theorem 7 of [24] (so that the proof there can be applied), but due to the generalized Definition 5 now also work for the case of different dwell times.

Theorem 8 *Given budget n, we have:*

$$|v^* - v(\hat{\mathbf{z}})| \leq \delta(h^*) \leq \begin{cases} O(n^{-d\frac{\log 1/\gamma}{\log \kappa}}) & \text{if } \kappa > 1 \\ O(\gamma^{n/C}) & \text{if } \kappa = 1 \end{cases} \tag{12}$$

where C is the constant from Definition 5.

Therefore, when κ is smaller (i.e. the problem is simpler), the algorithm converges faster with n, since the negative exponent of n is larger in magnitude. In particular, $\kappa = 1$ leads to exponential convergence in n. Furthermore, stronger dwell time conditions, represented by larger d, also increase the exponent, so the algorithm is also faster when the dwell time limits are larger, which makes sense since there are fewer solutions to consider. In the worst case, $\kappa = dN$ and near-optimality is $O(n^{d\frac{\log 1/\gamma}{\log dN}})$.

4.2 Unconstrained and single-agent cases

We show here that the analysis above specializes to the unconstrained setting, recovering (with minor conservatism) all the results from [8]. The analysis further encompasses the single-agent case with [5] or without dwell time [12], showing that the algorithm we provide here is a general tool for decision-making in single-agent or adversarial problems.

Consider first the unconstrained case, i.e. when $\underline{d}_u = \underline{d}_w = d = 1$. The entire analysis above holds as stated, however it simplifies in interesting ways starting from Definition 5.

This formula now only requires $|\mathcal{T}_h^*| \leq C\kappa^h$, so κ has an intuitive meaning in this case: it is the asymptotic branching factor of the near-optimal tree, i.e. the average number of children of each node on this tree, for large depths h. The a priori bound in the case $\kappa = 1$ remains the same, whereas it becomes $O(n^{-\frac{\log 1/\gamma}{\log \kappa}})$ for $\kappa > 1$. The analysis from [8] has been recovered. The range of κ is however $[1, N]$, as opposed to $[1, \sqrt{N_u N_w}]$ in [8], so some conservatism is introduced.

Introducing dwell-time constraints may either reduce or increase problem complexity, depending on whether they remove near-optimal solutions that would be difficult for the algorithm to find, leaving it with simpler ones, or the other way around. An explicit relation can be found for the case when the full trees are explored. Then, the convergence rates are $O(n^{-\frac{\log 1/\gamma}{\log N}})$ for the unconstrained problem, and $O(n^{-d\frac{\log 1/\gamma}{\log dN}})$ for the constrained one. We have $d\frac{\log 1/\gamma}{\log dN} \geq \frac{\log 1/\gamma}{\log N}$ for $d \geq 2$, so the constrained algorithm is faster.

In addition to generalizing the unconstrained minimax analysis to the dwell-time case, the analysis above also generalizes the single-agent (max-only) analysis with dwell time from [5] to the minimax setting. In particular, by taking a singleton set of min modes to effectively remove the min agent, we get the bound $O(n^{-d_u \frac{\log 1/\gamma}{\log \kappa}})$ from [5]. Furthermore, by removing the dwell-time constraint from this single-agent problem (or equivalently, the min agent from the unconstrained minimax problem), we obtain the algorithm and guarantees for optimistic planning for deterministic systems in [12], which we also applied in [5] to max-only unconstrained switches. These relationships are graphically represented in Fig. 5.

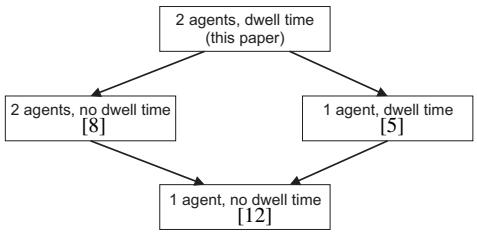

Fig. 5. Relationships between different planning settings and their analysis. The arrows point from the more general result to the specialized one.

Remark 1 Let us consider what happens when OMSd is applied to the Nash problem from Sec. 2.3. We study the return $l(\widehat{\mathbf{z}})$ of the finite-length sequence found by OMSd. The proof of Theorem 4 of [24] implies that $v^* - l(\widehat{\mathbf{z}}) \leq \delta(h(\widehat{\mathbf{z}})) = \delta(h^*)$. Coupling this with $v^* \geq v^{*\mathrm{N}}$ from Proposition 2, we have finally that $l(\widehat{\mathbf{z}}) \geq v^{*\mathrm{N}} - \delta(h^*)$, i.e. if the sequence returned were applied by the two agents, the return obtained would not be far from the Nash value.

5 Experimental evaluation

Our framework can address a wide range of noncooperative switched problems. Here, we consider networked control systems where the controller sends the chosen max modes to the system over a communication network affected by delays that are multiples of the sampling time. The delay is either controlled by an attacker who changes it so as to hinder performance, or otherwise it is random. In both cases we model the delay as a min switching signal, inspired by [11] and [26].

The attack setting is close to network security games [31,29], and in particular to [31] where the attacker can interfere with values sent along the control and measurement channels; here, the attacker attempts instead to mask its presence behind an innocuous delay signal. We give three examples in this setting. In the first example (Sec. 5.1), the max and min switching signals are unconstrained, leading to a problem of the type in Sec. 2.1. In the second example (Sec. 5.2), the delay signal must obey a minimum dwell time, as in Sec. 2.2. In the third example (Sec. 5.3), the min agent does not see the max mode before choosing the delay, illustrating the Nash problem of Sec. 2.3. Note that throughout Sec. 5.1-5.3, the delay agent explicitly acts to as to hinder the control agent, so the problem is truly noncooperative. Finally, Sec. 5.4 illustrates the method in real-time control, for a random delay signal that is still conservatively modeled as a noncooperative agent.

Before showing the results, we explain how the delay is modeled. Consider the system:

$$\tilde{y}_{k+1} = \tilde{g}(\tilde{y}_k, u_{k-w_k}), \qquad \forall k > 0 \tag{13}$$

where $\tilde{y}_k \in \mathbb{R}^p$ represents the system state at time $k \in \mathbb{Z}^+$, u_{k-w_k} is the controlled mode, and w_k is the delay (number of steps by which u is delayed) at k, which takes integer values in $\{0, 1, \ldots, m\}, m \geq 0$, so $N_w = m + 1$. A reward function $\tilde{r}(\tilde{y}_k, u_{k-w_k})$ is used that takes values in $[0, 1]$; note that the reward uses the delayed input, which means that it is generated at the system side. We will transform the problem in the minimax form (4), by defining dynamics $g(y, u, w)$ and rewards $r(y, u, w)$ that work with augmented state y. This state is, at step k:

$$y_k = [{y_k^0}^\top, y_k^1, y_k^2, \ldots, y_k^m]^\top := [\tilde{y}_k^\top, u_{k-1}, u_{k-2}, \ldots, u_{k-m}]^\top$$

Then, the augmented dynamics g that represent (13) are:

$$\begin{aligned} y_{k+1} = g(y_k, u_k, w_k) :=& [\tilde{y}_{k+1}^\top, u_k, u_{k-1}, \ldots, u_{k-m+1}]^\top \\ =& [\tilde{y}_{k+1}^\top, u_k, y_k^1, \ldots, y_k^{m-1}]^\top \end{aligned}$$

where the underlying state $\tilde{y}_{k+1}$ is computed as follows:

$$\tilde{y}_{k+1} = \begin{cases} \tilde{g}(\tilde{y}_k, u_k) = \tilde{g}(y_k^0, u_k) & \text{if } w_k = 0 \\ \tilde{g}(\tilde{y}_k, u_{k-w_k}) = \tilde{g}(y_k^0, y_k^{w_k}) & \text{if } w_k > 0 \end{cases}$$

The augmented reward function r that represents $\tilde{r}$ is:

$$r(y_k, u_k, w_k) := \begin{cases} \tilde{r}(\tilde{y}_k, u_k) = \tilde{r}(y_k^0, u_k) & \text{if } w_k = 0 \\ \tilde{r}(\tilde{y}_k, u_{k-w_k}) = \tilde{r}(y_k^0, y_k^{w_k}) & \text{if } w_k > 0 \end{cases}$$

This framework is general enough to allow any nonlinear dynamics $\tilde{g}$. In all the simulations (Sec. 5.1–5.3), we apply it to an inverted pendulum driven by a DC motor, with two states: angle α and angular velocity $\dot{\alpha}$, and a voltage input $\tilde{u}$. The continuous-time dynamics are given e.g. in [7], and are discretized via numerical integration with $T_\mathrm{s} = 0.05\,\mathrm{s}$ to obtain $\tilde{g}$. The goal is to stabilize the mass pointing upwards (corresponding to $\alpha = 0$), and the maximum voltage ($3\,\mathrm{V}$) is insufficient from some initial states to bring the mass up in one go; instead it must first be swung back and forth to accumulate energy. To perform these swing-ups, the control therefore requires large planning horizons, and adding time delays further increases the difficulty.

The reward is taken quadratic, $-(5\alpha^2 + 0.1\dot{\alpha}^2 + \tilde{u}^2)$, and normalized to $[0, 1]$ using the knowledge that $\tilde{u} \in [-3, 3]\,\mathrm{V}$, as well as the state bounds $\alpha \in [-\pi, \pi]\,\mathrm{rad}$, $\dot{\alpha} \in [-15\pi, 15\pi]\,\mathrm{rad/s}$ enforced by saturation. The reward can be written as a function r or $\tilde{r}$. We take discount factor $\beta = 0.97$, corresponding to $\gamma = \sqrt{0.97}$.[1] There are $N_u = 3$ controlled modes: modes 1 and 3 correspond to the maximum-magnitude voltage levels, namely -3 and $3\,\mathrm{V}$, while mode 2 is a stabilizing, linear state feedback $\tilde{u} = K \cdot [\alpha, \alpha]^\top$ saturated to $\pm 1.5\,\mathrm{V}$. The gains K are designed with discounted LQR on the linearized dynamics around zero, see Ch. 3 of [2]. Note that this lower-level feedback is on the system side, so it is not affected by delays. It only stabilizes the system from the neighborhood of the target equilibrium, and cannot perform the (nonlinear) swingup. The delay is either 0 or 1 steps, leading to $N_w = 2$. It should also be noted that the implementation of OMSd exploits the knowledge that rewards are 0 at max steps to compute tighter bounds than the general formulas after Assumption 1; the analysis remains conservatively valid.

5.1 Results in the unconstrained case

In the first experiment, the control and delay signals do not have dwell time constraints ($\underline{d}_u = \underline{d}_w = 1$). The max agent applies OMSd in receding horizon as explained at the end of Sec. 3. The min agent could apply any algorithm, but here it uses OMSd, also in receding horizon. For each agent, a small, medium, and large budget are tested; the min agent values are smaller overall as it is intuitively easier to hinder via delays than to solve the task. Specifically, the budget n of the max agent takes values $3000, 6000, 9000$, and for the min agent $n_w = 100, 500, 1000$. For each combination a 100-step long controlled trajectory is executed, and Fig. 6 reports the discounted returns obtained. Performance generally decreases for larger n_w, as expected, since the min agent becomes better at hindering the control. Performance is better for $n = 6000$ than for 9000; this sometimes occurs for planning algorithms, since small budgets (short horizons) can lead by luck to good long-term solutions.

[1] A numerical study in the unconstrained case of Sec. 5.1 shows that taking β (or equivalently γ) much smaller or larger than these values leads to lower performance in terms of the total, undiscounted rewards.

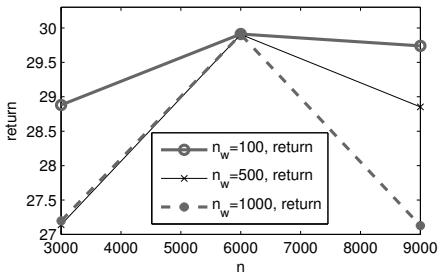

Fig. 6. Returns in the unconstrained case. Each curve shows the evolution as n varies, for a given n_w.

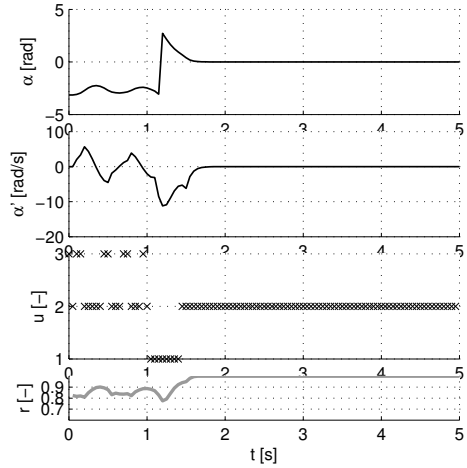

Fig. 7. A controlled trajectory. The modes 1, 2, or 3 chosen at each step are shown by 'x' markers.

Fig. 7 shows a representative trajectory, for $n = 9000$ and $n_w = 500$, selected to have medium performance. The swingup and stabilization are achieved, although the initial part with negative angles can be shortened to improve performance, by applying a more the maximum-magnitude mode 3 instead of the lower-magnitude mode 2 (which the algorithm identifies e.g. for $n = 6000$).

5.2 Results with dwell-time constraints

Next, the delay is restricted to satisfy a minimum dwell-time $\underline{d}_w$. Such a condition arises e.g. when the attacker wishes to imitate a delay signal with a bounded rate of change. Fig. 8 shows the dependence of the return on $\underline{d}_w$ for $n = 3000$, $n_w = 500$. Performance generally increases as the delay signal must obey stricter constraints, although there is a small decrease from 2 to 3, which may be due e.g. to a lucky solution of the min agent.

5.3 Results in the Nash problem

So far the attacker (min agent) could observe the max mode before it decided on the delay. In this section, we assume that it cannot, which leads to a Nash setting as in Sec. 2.3. The max agent can simply continue applying the Stackelberg algorithm, in which case Proposition 2 and the Remark of Sec. 4 hold. To allow a symmetrical reasoning to hold for the min agent, it can still apply OMSd but it has to solve a "reversed" Stackelberg problem where it is the leader and

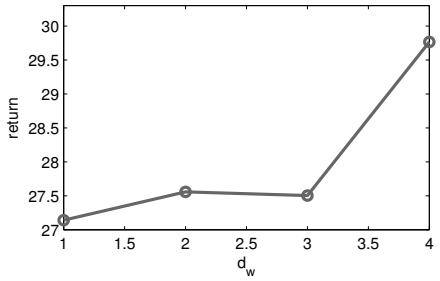

Fig. 8. Returns as the min agent dwell-time limit varies. Note that the vertical axis is kept constant in return plots to ensure they are comparable.

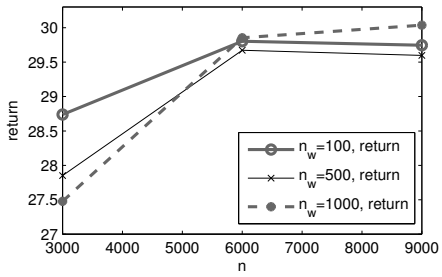

Fig. 9. Returns in the Nash case.

the max agent can react to its mode choice. The easiest way to understand this problem is to swap the two agents in problem (4), making the min agent the max one, with swapped signals $u' = w$ and $w' = u$, the same dynamics $g'(x, u', w') = g(x, u, w)$, and a reversed reward function $r'(x, u', w') = 1 - r(x, u, w)$. Then, (4) can be solved as usual, with OMSd from the perspective of the new max agent u'. Once both u and $w = u'$ have been decided, they are applied and the procedure is iterated in receding horizon.

Fig. 9 shows the returns obtained for the same budgets as in Fig. 6. The same trend of good performance with $n = 6000$ is observed, although now $n = 9000$ also works well; the effect of n_w is less clear than before.

Note that in [24], we provided simulation results for the case when the delay is an uncontrolled random signal. This illustrates that our method can be applied for robust control, where the disturbance (here, the delay) is conservatively handled as a smart min agent. We not repeat these results here, moving on instead to a real-time example where the delay is also random.

5.4 Real-time results for the rotational pendulum

We finally report a real-time control experiment, on a different, rotational pendulum, in which the pendulum link is indirectly driven via a horizontal link actuated by the motor. The states are the pendulum and horizontal joint angles, α and θ, and their angular velocities. The controlled modes u are, in order: the minimum voltage -6 V, a stabilizing controller saturated to ± 3 V, computed with discounted LQR as for the simple pendulum above, and the maximum voltage 6 V. The sampling time is 0.04 s, and the unnormalized, quadratic reward function is $\tilde{\rho}(x, u) = -15\alpha^2 - 0.05(\dot{\alpha}^2 + \theta^2 + \dot{\theta}^2 + \tilde{u}^2)$,

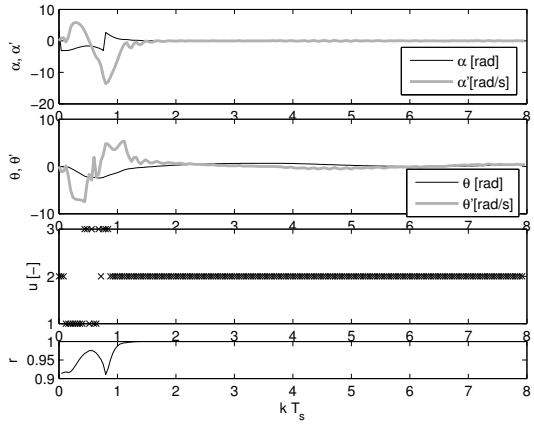

Fig. 10. Real-time trajectory of the rotational pendulum.

with $\tilde{u}$ the voltage. The delay is still 0 or 1 steps, and the discount factor $\beta = 0.95$.

The pendulum is initially in the stable downwards position, and the first control is computed as usual. Afterwards, the procedure must be changed for real-time control. We cannot wait until sampling instant $k+1$ to run the planner at x_{k+1}, since the execution time would be too long. Instead, planning is run for step $k + 1$ already during sampling interval k. However, the delay w_k only becomes known at sampling instant $k + 1$, once we get information about it along the feedback path. Thus during sampling interval k we must develop minimax trees from the predicted outcome states of *all* possible delays. Instead of using a prespecified budget, the sampling interval is divided equally among the trees developed, with a safety margin to ensure the algorithm finishes in time, and OMSd is run for each outcome state until it exhausts the allocated time. Then, at $k+1$, we select a control (in negligible computation time) from the tree that corresponds to the actual measured delay. The control is sent to the system, and depending on the delay, the system will receive the latest computed control or a previous one. In our experiment the network is simulated by artificially generating delays that are uniform random.

The computer uses an Intel Xeon E5-1620 CPU at 3.6GHz and has 16GB RAM. A C++ OMSd implementation is used. As shown in Fig. 10, the pendulum is swung up and stabilized despite the delay. A video illustrating the results can be found at `http://rocon.utcluj.ro/files/rotpend_oms.mp4`.

6 Conclusions and future work

We have studied optimistic minimax search for adversarial switched problems, handling both the cases when the two switching signals are unconstrained, and when they must obey minimum dwell times. We have shown convergence in a well-characterized way towards the optimal (constrained) minimax value, and illustrated the algorithm numerically and in real-time control.

When the uncontrolled signal evolves along a known Markov

chain, it will pay off to develop a tailored algorithm and analysis rather than conservatively assume the worst-case signal as above. Our optimistic method for stochastic Markov decision processes in [6] is suited to this problem, while the analysis will require significant changes to handle dwell time constraints.

References

[1] H. Berliner, "The B* search algorithm: A best first proof procedure," *Artificial Intelligence*, vol. 12, 1979.

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

[3] P. Bolzern, P. Colaneri, and G. D. Nicolao, "Design of stabilizing strategies for discrete-time dual switching linear systems," *Automatica*, vol. 69, pp. 93–100, 2016.

[4] M. Breton, A. Alj, and A. Haurie, "Sequential stackelberg equilibria in two-person games," *Journal of Optimization Theory and Applications*, vol. 59, no. 1, pp. 71–97, 1988.

[5] L. Buşoniu, J. Daafouz, M. C. Bragagnolo, and I.-C. Morarescu, "Planning for optimal control and performance certification in nonlinear systems with controlled or uncontrolled switches," *Automatica*, vol. 78, pp. 297–308, 2017.

[6] 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.

[7] L. Buşoniu, R. Munos, and R. Babuška, "A review of optimistic planning in Markov decision processes," in *Reinforcement Learning and Adaptive Dynamic Programming for Feedback Control*, F. Lewis and D. Liu, Eds. Wiley, 2012.

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

[9] H. Ehtamo and T. Raivio, "On applied nonlinear and bilevel programming for pursuit-evasion games," *Journal of Optimization Theory and Applications*, vol. 108, no. 1, pp. 65–96, 2001.

[10] X. He, A. Prasad, and S. P. Sethi, "Cooperative advertising and pricing in a dynamic stochastic supply chain: Feedback stackelberg strategies," *Production and Operations Management*, vol. 18, no. 1, pp. 78–94, 2009.

[11] L. Hetel, J. Daafouz, and C. Iung, "Equivalence between the lyapunov-krasovskii functional approach for discrete delay systems and the stability conditions for switched systems," *Nonlinear Analysis: Hybrid Systems*, vol. 2, no. 3, pp. 697–705, 2008.

[12] 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.

[13] 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.

[14] D. E. Knuth and R. W. Moore, "An analysis of alpha-beta pruning," *Artificial Intelligence*, vol. 6, no. 4, pp. 293–326, 1975.

[15] R. E. Korf and D. M. Chickering, "Best-first minimax search," *Artificial Intelligence*, vol. 84, no. 1–2, pp. 299–337, 1996.

[16] G. Leitmann, "On generalized stackelberg strategies," *Journal of Optimization Theory and Applications*, vol. 26, no. 4, pp. 637–643, 1978.

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

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

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

[20] ——, "From bandits to Monte Carlo tree search: The optimistic principle applied to optimization and planning," *Foundations and Trends in Machine Learning*, vol. 7, no. 1, pp. 1–130, 2014.

[21] A. J. Palay, "The B* tree search algorithm – new results," *Artificial Intelligence*, vol. 19, pp. 145–163, 1982.

[22] J. Pearl, "The solution for the branching factor of the alpha-beta pruning algorithm and its optimality," *Communications of the ACM*, vol. 25, no. 8, pp. 559–564, 1982.

[23] R. Postoyan, L. Buşoniu, D. Nešić, and J. Daafouz, "Stability analysis of discrete-time infinite-horizon optimal control with discounted cost," *IEEE Transactions on Automatic Control*, vol. 62, no. 6, pp. 2736–2749, 2017.

[24] J. B. Rejeb, L. Buşoniu, I.-C. Morarescu, and J. Daafouz, "Near-optimal control of nonlinear switched systems with non-cooperative switching rules," in *Proceedings IEEE American Control Conference (ACC-17)*, Seattle, US, 24–26 May 2017.

[25] W. Saad, Z. Han, H. V. Poor, and T. Basar, "Game-theoretic methods for the smart grid: An overview of microgrid systems, demand-side management, and smart grid communications," *IEEE Signal Processing Magazine*, vol. 29, no. 5, pp. 86–105, 2012.

[26] A. Tzes, G. Nikolakopoulos, and I. Koutroulis, "Development and experimental verification of a mobile client-centric networked controlled system," *European Journal of Control*, vol. 11, no. 3, pp. 229–241, 2005.

[27] Y. Wang and S. Gelly, "Modifications of UCT and sequence-like simulations for Monte-Carlo Go," in *Proceedings 2007 IEEE Symposium on Computational Intelligence and Games (CIG-07) USA, 1-5 April, 2007*, Honolulu, Hawaii, 1–5 April 2007, pp. 175–182.

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

[29] J. Zheng and D. A. Castañon, "Stochastic dynamic network interdiction games," in *Proceedings 2012 IEEE American Control Conference (ACC-12)*, Montreal, Canada, 27–29 June 2012, pp. 1838–1844.

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

[31] M. Zhu and S. Martinez, "Stackelberg-game analysis of correlated attacks in cyber-physical systems," in *Proceedings 2011 IEEE American Control Conference (ACC-11)*, San Francisco, US, 29 June – 1 July 2011, pp. 4063–4068.