An analysis of optimistic, best-first search for minimax sequential decision making

Lucian Buşoniu
Department of Automation
Technical University of Cluj-Napoca, Romania
Email: lucian@busoniu.net

Rémi Munos
Team SequeL
INRIA Lille, France
Email: remi.munos@inria.fr

Előd Páll
Department of Automation
Technical University of Cluj-Napoca, Romania
Email: pall.elod@gmail.com

***Abstract*—We consider problems in which a maximizer and a minimizer agent take actions in turn, such as games or optimal control with uncertainty modeled as an opponent. We extend the ideas of optimistic optimization to this setting, obtaining a search algorithm that has been previously considered as the best-first search variant of the B* method. We provide a novel analysis of the algorithm relying on a certain structure for the values of action sequences, under which earlier actions are more important than later ones. An asymptotic branching factor is defined as a measure of problem complexity, and it is used to characterize the relationship between computation invested and near-optimality. In particular, when action importance decreases exponentially, convergence rates are obtained. Throughout, examples illustrate analytical concepts such as the branching factor. In an empirical study, we compare the optimistic best-first algorithm with two classical game tree search methods, and apply it to a challenging HIV infection control problem.**

I. Introduction

The recent paradigm of optimistic optimization and planning combines ideas from global optimization, bandit theory for exploration in reinforcement learning, classical graph search algorithms, and optimal control in Markov decision processes [16]. The driving idea is optimism in the face of uncertainty, which in this context means that given partial information about possible solutions to an optimization problem, the most promising region of the solution space is refined further. In applications to different problems, the optimistic paradigm has led to some good algorithms, e.g. for optimization [15], for optimal control of discrete-actions deterministic systems [7] or stochastic Markov decision processes [3], [5], or for continuous actions [4], [14]. A core feature of optimistic methods is generality, e.g. in control they allow arbitrary nonlinear systems and nonquadratic reward functions. Another major advantage is a quantitative analysis that relates computation invested with near-optimality, where the relationship depends on a complexity measure for the problem. Related algorithms include e.g. upper confidence trees [9] and forward search sparse sampling [21].

Here, we consider the extension of optimistic ideas to sequential, adversarial decision-making problems, see e.g. [12, Ch. 10]. Two adversarial agents take discrete actions in turn, one of them aiming to maximize the infinite-horizon cumulative value of the actions, and the other to minimize it. This framework can model important classes of problems, including e.g. turn-based games such as go or chess, as well as optimal control under uncertainty, where the uncertainty is conservatively treated as the action of the opponent agent. It turns out that applying optimism in the adversarial setting naturally leads to the *best-first search* variant [17] of B*, a classical minimax algorithm proposed by [2] in 1979. Since the name "best-first search" has been used for many other methods (including A* and even two minimax techniques: fixed-depth [19] and adaptive-depth best-first search [11]), to avoid confusion we call the algorithm *optimistic minimax search* (OMS), always keeping in mind its relation to B*.

OMS explores a tree representation of the possible sequences of max and min agent actions, as do other minimax search algorithms such as alpha-beta pruning [8] or those in [19], [11]. At each leaf node, OMS requires lower and upper bounds on the values of action sequences passing through that node, and it propagates these bounds upwards in the tree by maximization or minimization according to the type of node. The next leaf to expand is selected optimistically, by starting from the root and recursively moving to a child that maximizes the upper bound at max nodes, or minimizes the lower bound at min nodes. OMS can stop after any number of iterations, after which it returns the deepest expanded node. Thus it is an adaptive-depth, anytime algorithm.

By exploiting the optimistic framework, we are able to develop theoretical performance guarantees for OMS – which to our best knowledge were missing from the literature on B* search. Specifically, we provide conditions under which OMS is guaranteed to approach the minimax-optimal solution as the budget of node expansions increases. These conditions impose structure on the value function so that earlier decisions are more important than later ones, and require this structure to be reflected in the bounds. *A posteriori*, OMS is then near-optimal to the extent of the gap between the upper and lower bounds at the deepest expanded node. To obtain an *a priori* bound, we characterize the size of the subset of nodes that OMS expands by its asymptotic branching factor, and use this factor to provide a tight relationship between computation invested and near-optimality. In particular, when the gaps decrease exponentially with the depth, the convergence rate is directly characterized. Throughout the paper, we illustrate the theoretical framework in several classes of problems, including function optimization, games, and optimal control under uncertainty. In these examples, we study the value of the branching factor, illustrating that it is a meaningful measure of problem complexity. An empirical study illustrates the analytical properties of OMS, and also includes the control problem of optimal treatment of HIV infection under uncertainty on drug effectiveness.

The importance of the effective branching factor in the analysis of minimax algorithms was understood as early as [8],

Acknowledgement: This work was supported by a grant of the Romanian National Authority for Scientific Research, CNCS-UEFISCDI, project number PNII-RU-TE-2012-3-0040.

[18], where it was applied to alpha-beta pruning, see also [10]. However, OMS is adaptive-depth and behaves quite differently from fixed-depth methods like alpha-beta. It is closer to the adaptive-depth best-first method of [11], which does not have an analysis and in fact may converge to suboptimal solutions, as we will show in an example. Here we provide a general analysis of OMS near-optimality and branching factor, placing them in direct connection with (smoothness) properties of the value function – something that is largely missing in works analyzing classical minimax methods. From this perspective our branching factor is closer to other complexity measures in optimistic methods, such as the branching factor in [7], the near-optimality dimension in [15], and the near-optimality exponent in [5]. Different from these however, it works in minimax problems and filters nodes using a nontrivial, nonlocal property, which must hold for the entire path to the node. Finally, it must be noted that the B* search algorithm, of which OMS is a special case, aims only to find the optimal action at the root, whereas OMS as applied here further refines the value at the root even after the first action is clear, which is useful in optimization.

Next, Section II introduces our formal framework, with examples, Section III gives the algorithm, Section IV provides its analysis, again with examples, and Section V gives illustrative experiments where OMS is compared with alpha-beta pruning [8] and adaptive-depth best-first search [11]. Section VI concludes.

II. Problem definition

Consider an adversarial, sequential decision-making problem where a maximizer (max) and a minimizer (min) agent take actions in turn. The max and min actions are respectively denoted u and w, and belong to action spaces U and W. We assume that U and W contain finitely many elements, N_U and N_M respectively. A generic action is denoted $z \in Z := U \cup W$, and can be either a max or min action. Denote an infinite sequence of actions by $\mathbf{z}_\infty = (z_0, z_1, z_2, z_3, \dots) = (u_0, w_0, u_1, w_1, \dots) \in (U \times W)^\infty$, and a finite sequence of h actions by $\mathbf{z}_h = (z_0, z_1, \dots, z_{h-1})$, with $\mathbf{z}_0$ the empty sequence by convention. The truncation of $\mathbf{z}_\infty$ to h initial elements is denoted $\mathbf{z}_{\infty|h}$. Finally, define a sequence of reward functions $\rho_h : (U \times W)^{\lfloor h \rfloor} \times U^{\lceil h \rceil} \to \mathbb{R},\ h \geq 1$ were notation $\lfloor h \rfloor$ means the result of the integer division of h by 2 and $\lceil h \rceil$ the remainder. Here, the convention is that a set to power 0 is omitted. The meaning of $\rho_h(\mathbf{z}_h)$ is that of immediate reward following a sequence of h decisions. Then, the overall infinite-horizon value of sequence $\mathbf{z}_\infty$ is:

$$v(\mathbf{z}_\infty) := \sum_{h=1}^{\infty} \rho_h(\mathbf{z}_{\infty|h}) \tag{1}$$

The goal is to find the minimax-optimal value, defined as:

$$v^* := \lim_{k\to\infty} \left[\max_{u_0} \min_{w_0} \cdots \max_{u_{k-1}} \min_{w_{k-1}} \sum_{h=1}^{2k} \rho_h(\mathbf{z}_h) \right] \tag{2}$$

when this limit exists.[1] This problem is similar e.g. to the one in [12, Ch. 10].

[1]Decisions u, w, and index k are used when the max and min actions are regarded separately; otherwise, we use generic decision z and index h.

Define $\mathcal{Z}(\mathbf{z}_h) = \left\{ \mathbf{z}_\infty \,\middle|\, \mathbf{z}_{\infty|h} = \mathbf{z}_h \right\}$, the set of sequences starting with $\mathbf{z}_h$. The following requirement sits at the core of our approach.

Assumption 1: There exist functions l and b and a decreasing sequence $\{\delta(h)\}_{h\geq 0}$ of positive real numbers so that for any action sequence $\mathbf{z}_h$:

$$l(\mathbf{z}_h) \leq v(\mathbf{z}_\infty) \leq b(\mathbf{z}_h), \forall \mathbf{z}_\infty \in \mathcal{Z}(\mathbf{z}_h) \tag{3}$$

$$b(\mathbf{z}_h) - l(\mathbf{z}_h) \leq \delta(h) \tag{4}$$

Thus, (3) says that l and b are lower and upper bounds on values of sequences starting with $\mathbf{z}_h$. Our algorithm will require access to such bounds. Equation (4) intuitively restricts to problems where later decisions matter less than earlier ones. We will also call $\delta(h)$ the *gap* (between the two bounds).

Example 1: Adversarial optimization. Our first example is academic, and will later provide important insight into the behavior of the algorithm. Consider a function $f(x, y)$, $f : [0, 1] \times [0, 1] \to \mathbb{R}$. Both agents take binary decisions, $U = W = \{0, 1\}$, with the following meaning. The max agent takes the domain $[0, 1] \times [0, 1]$ and splits it in half along dimension x, selecting the first half if $u = 0$ and the second if $u = 1$. The min agent then takes the resulting set and similarly splits it in half along dimension y. The max agent takes over and splits along x, and so on, see Figure 1. An infinite sequence $\mathbf{z}_\infty$ corresponds to a point and its value is $v(\mathbf{z}_\infty) = f(x, y)$, assuming f can be decomposed in the form (1).

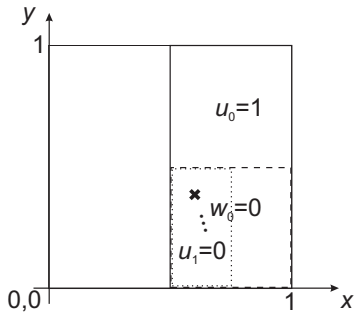

Fig. 1. Adversarial optimization. The max agent takes action 1 choosing the continuous-outline box, the min agent 0 choosing the dashed box, and the max agent then applies 0 to choose the dotted box. Any infinite sequence of decisions is uniquely associated to a point.

Take, for example, function $f(x, y) = x + y$, which satisfies this property. For this function, upper and lower bounds can be easily found as follows. Each finite sequence $\mathbf{z}_h$ corresponds to a box $(X, Y, \Delta_x, \Delta_y)$ where X, Y are the lower-left coordinates and Δ_x, Δ_y the lengths of the sides. Then, $l(\mathbf{z}_h) = X + Y$, $b(\mathbf{z}_h) = X + Y + \Delta_x + \Delta_y$. Further, $\Delta_x = 2^{-\lfloor h+1 \rfloor}, \Delta_y = 2^{-\lfloor h \rfloor}$, so that $b(\mathbf{z}_h) - l(\mathbf{z}_h) \leq 2 \cdot 2^{-h/2+1} \leq 4 \cdot (1/\sqrt{2})^h =: \delta(h)$. The minimax-optimal value is $v^* = \max_x \min_y f(x, y) = 1$ and the corresponding minimax solution is the lower-right corner of the domain. □

Example 2: Two-player games with discount. Consider a turn-based game such as go, where the state of the board is represented by vector x. At turn $k \geq 0$, the player takes decision $u_k = z_{2k}$ and the opponent responds with $w_k = z_{2k+1}$. These decisions affect the board according to

a transition function, $x_{h+1} = f(x_h, z_h)$, and the player attains rewards $\tilde{\rho}(x_h, z_h, x_{h+1})$, e.g., in go related to the territory and the number of pieces taken. The goal is to achieve discounted, minimax-optimal play:

$$\lim_{k\to\infty}\left[\max_{u_0}\min_{w_0}\cdots\max_{u_{k-1}}\min_{w_{k-1}}\sum_{h=0}^{2k}\gamma^h\tilde{\rho}(x_h, z_h, x_{h+1})\right]$$

This is modeled in our framework by taking $\rho_h(\mathbf{z}_h) := \gamma^{h-1}\tilde{\rho}(x_{h-1}, z_{h-1}, x_h)$, while noting that the dependence of the rewards on the sequence of previous actions is collapsed into the state signal.

To ensure Assumption 1, we impose the following:

Assumption 2: Rewards are bounded to the unit interval, $\tilde{\rho} : X \times Z \times X \to [0, 1]$.

This may require rescaling the original, nonunit rewards. Since all rewards after applying $\mathbf{z}_h$ are in $[0, 1]$, we have $l(\mathbf{z}_h) = \sum_{j=0}^{h-1}\gamma^j\tilde{\rho}(x_j, z_j, x_{j+1})$ and $b(\mathbf{z}_h) = l(\mathbf{z}_h) + \frac{\gamma^h}{1-\gamma}$, with the convention that an empty sum is 0. Therefore, $\delta(h) = \frac{\gamma^h}{1-\gamma}$, and Assumption 1 is satisfied.

There may be terminal, game-over states, from which any transition ends up in the same state with reward 0. □

Example 3: Discounted optimal control with disturbance. Finally, take an optimal control problem for a system affected by disturbances. The dynamics at discrete-time step k are: $x_{k+1} = f(x_k, u_k, w_k)$, where u is now the applied action and w is the disturbance. A reward $r_{k+1} = \tilde{\rho}(x_k, u_k, w_k, x_{k+1})$ is obtained, and the goal is to achieve the best possible discounted return, conservatively taking into account the worst possible disturbances, as usually done in robust control:

$$\lim_{k\to\infty}\left[\max_{u_0}\min_{w_0}\cdots\max_{u_{k-1}}\min_{w_{k-1}}\sum_{j=0}^{k-1}\gamma^j\tilde{\rho}(x_j, u_j, w_j, x_{j+1})\right]$$

To place this in our framework, take:

$$\rho_h(\mathbf{z}_h) := \begin{cases}0, & \text{if } h = 2k+1\\ \gamma^k\tilde{\rho}(x_k, u_k, w_k, x_{k+1}) & \text{if } h = 2k+2\end{cases}$$

We again impose reward boundedness to the unit interval:

Assumption 3: The reward function satisfies $\tilde{\rho} : X \times U \times W \times X \to [0, 1]$.

Then, $l(\mathbf{z}_h) = \sum_{k=0}^{\lfloor h\rfloor-1}\gamma^k\tilde{\rho}(x_k, u_k, w_k, x_{k+1})$ and $b(\mathbf{z}_h) = l(\mathbf{z}_h) + \frac{\gamma^{\lfloor h\rfloor}}{1-\gamma}$, so that $\delta(h) = \frac{\gamma^{\lfloor h\rfloor}}{1-\gamma} \le \frac{\gamma}{1-\gamma}\sqrt{\gamma}^h$, and Assumption 1 is satisfied. □

It must be emphasized that in contrast to Example 1, Examples 2 and 3 comprise entire classes of practical problems.

More generally, lower and upper bounds can be derived if v is Lipschitz under a metric ℓ on the space of sequences:

$$|v(\mathbf{z}_\infty) - v(\mathbf{z}'_\infty)| \le \ell(\mathbf{z}_\infty, \mathbf{z}'_\infty)$$

and if for any set $\mathcal{Z}(\mathbf{z}_h)$, we have access to the value of a sample $\mathbf{z}_\infty \in \mathcal{Z}(\mathbf{z}_h)$. Define $\operatorname{diam}(\mathcal{Z}(\mathbf{z}_h)) := \sup_{\mathbf{z}'_\infty\in\mathcal{Z}(\mathbf{z}_h)}\ell(\mathbf{z}_\infty, \mathbf{z}'_\infty)$, then $\forall\mathbf{z}'_\infty \in \mathcal{Z}(\mathbf{z}_h)$:

$$v(\mathbf{z}'_\infty) \ge v(\mathbf{z}_\infty) - \operatorname{diam}(\mathcal{Z}(\mathbf{z}_h)) =: l(\mathbf{z}_h)$$
$$v(\mathbf{z}'_\infty) \le v(\mathbf{z}_\infty) + \operatorname{diam}(\mathcal{Z}(\mathbf{z}_h)) =: b(\mathbf{z}_h)$$

Then, $\delta(h) = 2\operatorname{diam}(\mathcal{Z}(\mathbf{z}_h))$ and the condition on $\delta(h)$ from Assumption 1 turns into a requirement on the diameters and thus on the smoothness of v.

In fact, in e.g. Example 1 the bounds follow from the Lipschitz property of $f(x, y) = x + y$ in the L_1 metric. In Example 2, a Lipschitz property of v holds for the metric $\ell(\mathbf{z}_\infty, \mathbf{z}'_\infty) = \frac{\gamma^{h(\mathbf{z}_\infty,\mathbf{z}'_\infty)}}{1-\gamma}$, where $h(\mathbf{z}_\infty, \mathbf{z}'_\infty)$ is the first index where the two sequences are different (a similar property holds for Example 3). However, the bounds in the examples were computed in a smarter way that did not require access to the exact value of a sample; indeed such a value will often be difficult to obtain since it is an infinite sum.

In general, we allow any procedure for computing the bounds (3) as long as they together with v satisfy the smoothness property (4).

III. Algorithm

Optimistic minimax search (OMS) explores a tree representation of the possible action sequences, as illustrated in Figure 2. OMS starts with a root node corresponding to the empty sequence, and iteratively expands n nodes. Expanding a node adds new children nodes corresponding to all the N_U max actions (for max nodes) or N_W min actions (for min nodes). Each node at some depth h is reached via a unique path through the tree, and is thus uniquely associated to the sequence of actions $\mathbf{z}_h = (z_0, z_1, \ldots, z_{h-1})$ on this path. In what follows, we will work interchangeably with sequences and nodes, keeping this equivalence in mind.

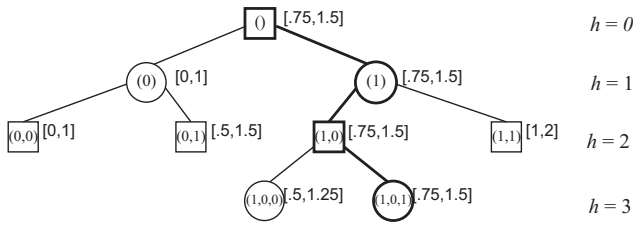

Fig. 2. Illustration of a minimax tree developed by the algorithm when applied to Example 1. Squares are max nodes, and circles min nodes. Nodes are labeled by action sequences, shown inside the node, as well as by the interval $[L, B]$, shown outside. Four nodes have been expanded, and the thick path leads to the node that the algorithm would expand at iteration five.

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 $\mathbf{z}$. The algorithm computes lower and upper bounds $L(\mathbf{z})$ and $B(\mathbf{z})$ for each node. They are initialized at the leaves using l and b from Assumption 1, and propagated upwards in the tree:

$$L(\mathbf{z}) = \begin{cases}l(\mathbf{z}), & \text{if } \mathbf{z} \in \mathcal{L}(\mathcal{T})\\ \max_{\mathbf{z}'\in\mathcal{C}(\mathbf{z})} L(\mathbf{z}'), & \text{if } \mathbf{z} \text{ max node, } \mathbf{z} \notin \mathcal{L}(\mathcal{T})\\ \min_{\mathbf{z}'\in\mathcal{C}(\mathbf{z})} L(\mathbf{z}'), & \text{if } \mathbf{z} \text{ min node, } \mathbf{z} \notin \mathcal{L}(\mathcal{T})\end{cases}$$
$$B(\mathbf{z}) = \begin{cases}b(\mathbf{z}), & \text{if } \mathbf{z} \in \mathcal{L}(\mathcal{T})\\ \max_{\mathbf{z}'\in\mathcal{C}(\mathbf{z})} B(\mathbf{z}'), & \text{if } \mathbf{z} \text{ max node, } \mathbf{z} \notin \mathcal{L}(\mathcal{T})\\ \min_{\mathbf{z}'\in\mathcal{C}(\mathbf{z})} B(\mathbf{z}'), & \text{if } \mathbf{z} \text{ min node, } \mathbf{z} \notin \mathcal{L}(\mathcal{T})\end{cases} \quad (5)$$

To choose the next leaf to expand, the algorithm starts from the root and constructs a path by recursively selecting an optimistic child for the agent at the current node. That

is, at max nodes a child with the largest upper bound is selected (optimistic for the max agent), while at min nodes the algorithm moves to a child with the smallest lower bound, which is optimistic for the min agent (it is pessimistic for the max agent).

OMS stops after n node expansions, and returns the deepest node expanded: its sequence $\hat{\mathbf{z}}$ and bounds. Algorithm 1 summarizes the entire procedure, where $(\cdot,\cdot)$ means the concatenation of the argument sequences and $h(\cdot)$ yields the depth (length) of the argument sequence. Ties in the maximizations and minimizations can be broken arbitrarily. Measuring computation by the number of node expansions is motivated by the fact that these operations are often the most expensive, such as e.g. in the control problem of Example 3, where expansion requires the simulation of the dynamics. OMS is an anytime algorithm: n does not have to be specified in advance, and the algorithm can be stopped after any number of expansions.

Algorithm 1 Optimistic minimax search

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**
    $\mathbf{z} \leftarrow \begin{cases} \arg\max_{\mathbf{z}' \in \mathcal{C}(\mathbf{z})} B(\mathbf{z}'), & \text{if } \mathbf{z} \text{ max node} \\ \arg\min_{\mathbf{z}' \in \mathcal{C}(\mathbf{z})} L(\mathbf{z}'), & \text{if } \mathbf{z} \text{ min node} \end{cases}$
  end while
  $\mathbf{z}(t) \leftarrow \mathbf{z}$
  expand $\mathbf{z}(t)$, by adding to $\mathcal{T}$ its children:
    $(\mathbf{z}(t), u)$ $\forall u \in U$, if $\mathbf{z}(t)$ max node
    or $(\mathbf{z}(t), w)$ $\forall w \in W$, if $\mathbf{z}(t)$ min node
  compute bounds for all $\mathbf{z} \in \mathcal{T}$ with (5)
end for
$\hat{\mathbf{z}} \leftarrow \arg\max_{\mathbf{z}(t), t=1,\dots,n} h(\mathbf{z})$

Output: $\hat{\mathbf{z}}$, $l(\hat{\mathbf{z}})$, $b(\hat{\mathbf{z}})$

Sometimes, OMS will be used with the intention of finding a decision to apply, rather than an approximation of the optimal value. In this case, the first action of the sequence $\hat{\mathbf{z}}$ is applied by the max agent, which then waits for the min agent's response and then reapplies OMS from the resulting situation (e.g., state). This can be seen as receding-horizon control [13]. Note that the min agent could itself apply OMS to find the actions, simply by starting with a min root node and then applying the algorithm as usual. Finally, the bounds L and B can be efficiently maintained by only updating at iteration t the path from the last expanded node $\mathbf{z}(t)$ to the root.

IV. Analysis

Let us first establish a basic property of OMS.

Lemma 4: At any iteration t, for any nodes $\mathbf{z}$, $\mathbf{z}' \in \mathcal{C}(\mathbf{z})$ on the optimistic path, we have $[L(\mathbf{z}), B(\mathbf{z})] \subseteq [L(\mathbf{z}'), B(\mathbf{z}')]$.

Proof: If $\mathbf{z}$ is a max node, $B(\mathbf{z}) = B(\mathbf{z}')$ and $L(\mathbf{z}) \geq L(\mathbf{z}')$ since $L(\mathbf{z})$ is the maximum among the children's L-values. The situation is symmetrical at min nodes. ■

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

$$v(\mathbf{z}_h) = \sum_{j=1}^{h} \rho_j(\mathbf{z}_j) + \begin{cases} \lim_{k\to\infty} [\max_{u_0} \min_{w_0} \cdots \max_{u_{k-1}} \min_{w_{k-1}} \sum_{j=1}^{2k} \rho_{h+j}((\mathbf{z}_h, \mathbf{z}_{\uparrow j}))] \\ \qquad \text{if } \mathbf{z}_h \text{ max node} \\ \lim_{k\to\infty} [\min_{w_0} \cdots \max_{u_{k-1}} \min_{w_{k-1}} \sum_{j=1}^{2k-1} \rho_{h+j}((\mathbf{z}_h, \mathbf{z}_{\downarrow j}))] \\ \qquad \text{if } \mathbf{z}_h \text{ min node} \end{cases} \tag{6}$$

again assuming that the limits exist. Here, $\mathbf{z}_{\uparrow j} = (u_0, w_0, u_{k-1}, w_{k-1})$, $\mathbf{z}_{\downarrow j} = (w_0, u_{k-1}, w_{k-1})$.

The second and final Lemma is essential to the analysis below, since it characterizes a restricted subset of nodes outside which the algorithm will never expand.

Lemma 5: At depth h in the tree, OMS only expands nodes in the set:

$$\mathcal{T}_h^* := \big\{\mathbf{z}_h \big| \, |v^* - v(\mathbf{z}_p)| \leq \delta(h), \quad \forall \mathbf{z}_p \text{ on path from root to } \mathbf{z}_h\big\} \tag{7}$$

Proof: We will show by induction from leaves to the root that:

$$v(\mathbf{z}) \in [L(\mathbf{z}), B(\mathbf{z})], \quad \forall \mathbf{z} \in \mathcal{T}$$

At any leaf, the base case holds by definition: $v(\mathbf{z}) \in [l(\mathbf{z}), b(\mathbf{z}_h)] = [L(\mathbf{z}_h), B(\mathbf{z}_h)]$. For the general case, consider an inner node $\mathbf{z}$, and assume the property is true at all its children $\mathbf{z}'$. We have by definition (6):

$$v(\mathbf{z}_p) = \begin{cases} \max_{\mathbf{z}' \in \mathcal{C}(\mathbf{z}_p)} v(\mathbf{z}') & \text{if } \mathbf{z}_p \text{ max node} \\ \min_{\mathbf{z}' \in \mathcal{C}(\mathbf{z}_p)} v(\mathbf{z}') & \text{if } \mathbf{z}_p \text{ min node} \end{cases}$$

We first show that $L(\mathbf{z}) \leq v(\mathbf{z})$. If $\mathbf{z}$ is a max node, take child $\mathbf{z}'$ so that $L(\mathbf{z}) = L(\mathbf{z}')$, then $L(\mathbf{z}) = L(\mathbf{z}') \leq v(\mathbf{z}') \leq v(\mathbf{z})$. If $\mathbf{z}$ is a min node, take child $\mathbf{z}'$ so that $v(\mathbf{z}') = v(\mathbf{z})$, therefore: $L(\mathbf{z}) \leq L(\mathbf{z}') \leq v(\mathbf{z}') = v(\mathbf{z})$. Property $B(\mathbf{z}) \geq v(\mathbf{z})$ is shown in a symmetrical way: If $\mathbf{z}$ is a max node, take child $\mathbf{z}'$ so that $v(\mathbf{z}') = v(\mathbf{z})$, therefore $B(\mathbf{z}) \geq B(\mathbf{z}') \geq v(\mathbf{z}') = v(\mathbf{z})$. If $\mathbf{z}$ is a min node, take child $\mathbf{z}'$ so that $B(\mathbf{z}) = B(\mathbf{z}')$, then $B(\mathbf{z}) = B(\mathbf{z}') \geq v(\mathbf{z}') \geq v(\mathbf{z})$.

Consider now any leaf $\mathbf{z}_h$ selected for expansion on the current tree, and any node $\mathbf{z}_p$ at depth p on the path from the root to this leaf. Applying Lemma 4 iteratively from $\mathbf{z}_p$ down to the leaf $\mathbf{z}_h$, we have $[L(\mathbf{z}_p), B(\mathbf{z}_p)] \subseteq [L(\mathbf{z}_h), B(\mathbf{z}_h)] = [l(\mathbf{z}_h), b(\mathbf{z}_h)]$. Then:

$$v(\mathbf{z}_p) \in [l(\mathbf{z}_h), b(\mathbf{z}_h)] \tag{8}$$

At the root, $v(\mathbf{z}_0) = v^*$. For any $\mathbf{z}_p$ the values v^* and $v(\mathbf{z}_p)$ are in the interval $[l(\mathbf{z}_h), b(\mathbf{z}_h)]$, which has length at most $\delta(h)$, so the property in (7) holds. This concludes the proof. ■

At this point, we can already provide an *a posteriori* bound for the algorithm that can be directly evaluated after the algorithm has run.

Theorem 6: Let h^* be the largest depth of any expanded node. Then, $|v^* - v(\hat{\mathbf{z}})| \leq \delta(h^*)$ and $v^* \in [L(\mathbf{z}_0), B(\mathbf{z}_0)]$.

Proof: Follows immediately from (8) and Algorithm 1. ■

Note again that $B(\mathbf{z}_0) - L(\mathbf{z}_0) \leq b(\hat{\mathbf{z}}) - l(\hat{\mathbf{z}}) = \delta(h^*)$. To obtain a more refined bound, which works *a priori*, we characterize the size of the expanded subset $\mathcal{T}^* = \bigcup_{h\geq 0} \mathcal{T}_h^*$. Let $|\cdot|$ denote the cardinality of the argument set.

Definition 7: Let κ be the smallest positive number so that $\exists C > 0, |\mathcal{T}_h^*| \leq C\kappa^h, \forall h \geq 0$.

The quantity κ is an asymptotic branching factor of $\mathcal{T}^*$, and it quantifies the complexity of the search problem. The smaller κ, the simpler the problem. The smallest possible value for κ is 1, when $\mathcal{T}_h^*$ contains a constant number of nodes at every h (e.g., just one minimax-optimal path), and the largest value is $\sqrt{N_U N_W}$, when $\mathcal{T}_h^*$ contains all the nodes at h, namely $N_U^{\lfloor h+1 \rfloor} N_W^{\lfloor h \rfloor}$ nodes. Below we exemplify κ in several problems. Note that κ is similar with other complexity measures used in optimistic optimization and planning [16], such as the branching factor in [7], the near-optimality dimension in [15], and the near-optimality exponent in [5]. These previous measures and algorithms are for problems involving a single decision-maker, and κ is more general since it applies to minimax problems. A crucial feature of κ and $\mathcal{T}^*$ is the nonlocal character of the inequality in (7), which must hold for any parent and not just the expanded node. This is important since it significantly reduces the size of the tree in some problems, as we will illustrate in Example 4.

Theorem 8: (i) Let $h(n)$ be the smallest depth h so that $\sum_{j=0}^{h} C\kappa^j \geq n$. Then, $|v^* - v(\hat{\mathbf{z}})| \leq \delta(h(n))$. (ii) Further, when $\exists c > 0, \beta \in (0,1)$ so that $\delta(h) \leq c\beta^h$, i.e. when the gap sequence decreases exponentially fast, then:

$$\delta(h(n)) \leq \begin{cases} c(\frac{\kappa-1}{Ck} \cdot n)^{-\frac{\log 1/\beta}{\log \kappa}} = O(n^{-\frac{\log 1/\beta}{\log \kappa}}) & \text{if } \kappa > 1 \\ c\beta^{n/C-1} = O(\beta^{n/C}) & \text{if } \kappa = 1 \end{cases} \tag{9}$$

Proof: For arbitrary h, OMS expands at most all the nodes up to h in $\mathcal{T}^*$ before expanding a node at $h+1$. Hence, since $\mathcal{T}^*$ contains at most $\sum_{j=0}^{h(n)-1} C\kappa^j$ nodes until $h(n)-1$, and the algorithm expands more nodes than this (since by assumption $n > \sum_{j=0}^{h(n)-1} C\kappa^j$). So, at least one node at $h(n)$ is expanded. From this $h^* \geq h(n)$ and since sequence $\delta(h)$ is decreasing, part (i) follows from Theorem 6.

To show part (ii), let $\kappa > 1$. Then $n \leq \sum_{j=0}^{h(n)} C\kappa^j = C\frac{\kappa^{h(n)+1}-1}{\kappa-1}$, and solving this for $h(n)$ we get $h(n) \geq \frac{\log n(\kappa-1)/C\kappa}{\log \kappa}$, which when replaced in $\delta(h(n))$ gives the desired inequality. Similarly, if $\kappa = 1$, we have $n \leq \sum_{j=0}^{h(n)} C = C(h(n)+1)$, from where $h(n) = \frac{n}{C} - 1$ which is substituted in $\delta(h(n))$. ■

Part (ii) of Theorem 8 is of practical importance, since in many problems the gap $\delta(h)$ will decrease exponentially with h, as e.g. in Examples 1–3, where β is respectively $1/\sqrt{2}$, γ, and $\sqrt{\gamma}$. The big-O expressions in (9) highlight the qualitative, asymptotic behavior of the algorithm, whereas the detailed expressions preceding them make the constants explicit. Suboptimality decreases polynomially fast with the computation n invested when $\kappa > 1$ (since the expanded tree grows exponentially), and exponentially fast with n when $\kappa = 1$ (since only a constant number of paths must be explored). Since κ is generally unknown, the near-optimality of OMS cannot be determined in advance. However, Theorem 8 provides confidence that the algorithm automatically adapts to the complexity of the problem.

Example 4: Adversarial optimization: branching factor. We will find κ for Example 1 with $f(x,y) = x + y$. For any finite sequence $\mathbf{z}_h$ the minimax-optimal value $v(\mathbf{z}_h)$ is the value of f in the lower-right corner of the corresponding box. Consider some arbitrary odd depth h; boxes $\mathbf{z}_h$ at this

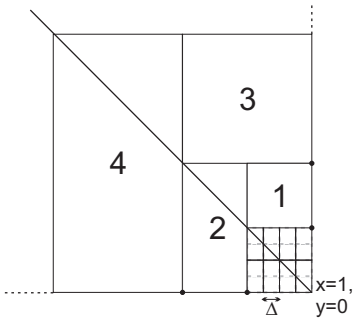

Fig. 3. Counting the nodes in $\mathcal{T}_h^*$. Some minimax-optimal points are highlighted with black disks.

depth are small tall rectangles like those shown in continuous outline at the bottom-right of Figure 3. Then the gap of these boxes is $\delta(h) = 3\Delta = 3 \cdot 2^{-(h+1)/2}$. Recall (7): if we can find a larger box containing $\mathbf{z}_h$ which is more than $\delta(h)$ away from the optimal value, then $\mathbf{z}_h$ will not be expanded. Now the suboptimality of any box is the distance between its lower-right corner and the main diagonal of the unit square. Since boxes 1 and 2 are 4Δ-away from optimum, no subbox $\mathbf{z}_h$ inside these larger boxes will be expanded. Boxes 3 and 4 are 8Δ-away so no subbox will be expanded there either, and continuing iteratively like this we can fill the entire domain *except* the lower-right corner, which contains 8 boxes. At depth $h+1$, boxes are square and have diameter 2Δ (shown in gray dotted line) so we can eliminate in a similar way all of them except the 16 in the corner. It follows that $|\mathcal{T}_h^*| \leq 16, \forall h$, and $\kappa = 1$: the problem is easy. The regret bound, including constants, is $4(\frac{1}{\sqrt{2}})^{n/16-1}$.

It must be emphasized that checking the suboptimality of *parent* boxes is crucial: if we only checked boxes for their own suboptimality $|v^* - v(\mathbf{z}_h)|$, no box with the lower-right corner on the main diagonal could be eliminated, leading to a number of boxes growing with the depth and a large branching factor: the difficulty of the problem would be misrepresented.

Note that such applications of tree search methods to minimax optimization have been studied before, see e.g. [20], although that paper uses a different theoretical framework. □

Example 5: Two-player games: branching factor. We will illustrate the meaning of κ and the regret bounds for Example 2 with discount factor γ, in two representative special cases.

Consider first that all rewards are equal, say they are all 1. Then, $v^* = 1/(1-\gamma)$ and *any* sequence has this value. So no nodes can be eliminated with the condition in (7), $\mathcal{T}^*$ contains the whole tree, and OMS will in fact explore nodes uniformly, in the order of their depth. As shown above, in this

case $\kappa = \sqrt{N_U N_W}$. Therefore, this uniform type of problem is an interesting worst case, where κ is the largest possible. From Theorem 9 with $\beta = \gamma$, near-optimality is $O(n^{-\frac{\log 1/\gamma}{\log \sqrt{N_U N_W}}})$.

Next, an example with $\kappa = 1$ is constructed, see Figure 4. At 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 $v^* = \gamma^0 + \gamma^2 + \ldots = \frac{1}{1-\gamma^2}$.

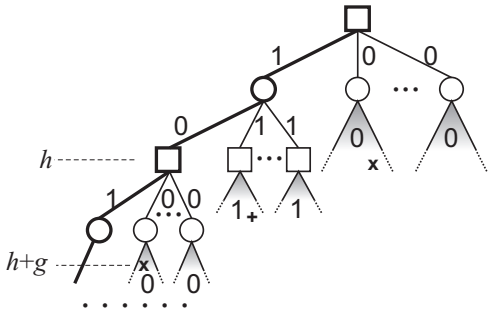

Fig. 4. A game tree with $\kappa = 1$. Rewards are shown along the transitions and inside subtrees where they remain constant. The thick path is minimax-optimal.

To study 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 $\delta(h+g) = \frac{\gamma^{g+h}}{1-\gamma}$, see Example 2, node $\mathbf{z}_{h+g}$ can be excluded when:

$$|v^* - v(\mathbf{z}_{h+g})| > \frac{\gamma^{g+h}}{1-\gamma}, \text{i.e. } \frac{\gamma^h}{1-\gamma^2} > \frac{\gamma^{g+h}}{1-\gamma}$$

which boils down to $g > \frac{\log(1-\gamma^2)/(1-\gamma)}{\log 1/\gamma}$, 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 the 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.

Defining $N = \max(N_U, N_W)$, at some arbitrary depth h' the set $\mathcal{T}^*_{h'}$ contains at most the following amount of children coming from optimal nodes at various depths $h < h'$, which cannot be excluded via the conditions above:

$$1 + N + N^2 + \ldots + N^G =: C$$

so that $\mathcal{T}^*_{h'} \leq C$ and the branching factor $\kappa = 1$. So this is an easy problem for OMS, and suboptimality is $O(\gamma^{n/C})$. □

V. Experimental study

First, in the optimization problem of Examples 1 and 4, we experimentally illustrate the practical effects of the theoretical properties studied above. Then, we show that OMS also works well in a challenging problem different from the games where it (and other minimax search algorithms) are usually applied: controlling infection with the human immunodeficiency virus (HIV), under uncertainty on the effectiveness of the drugs. This problem is in the class of Example 3.

A. Adversarial optimization

In addition to illustrating the properties of OMS, in this example we also compare it with two classical minimax search algorithms: alpha-beta pruning [8] and adaptive-depth best-first minimax search (BFMS) [11]. Alpha-beta pruning is well-known so we do not review it here. BFMS is less widely used but it is an anytime algorithm similar to OMS. It develops the tree of Figure 2, but instead of maintaining an interval at each node it uses just one value, which is initialized using a heuristic function at the leaves and then propagated upwards as in (5). At each iteration, BFMS expands the leaf of the *principal variation*, a path along which the root inherited its value. After expanding a given amount of nodes it returns the principal variation. For both alpha-beta pruning and BFMS, we use l and b as a heuristic at respectively max and min leaves.

The computational requirements of alpha-beta pruning are not directly controlled, instead it searches until a given depth in the tree. It also expands a varying amount of children per node. Thus, to keep the comparison fair, we vary the depths h in the range $3, 4, \ldots, 15$ and measure for each h the required computation, in the form of the number of nodes n_t created on the tree. Figure 5, top shows the resulting values of n_t. Then, we allow OMS and BFMS to create as many nodes. This is different from the number n of *expanded* nodes that we used in the theory above, but only up to a constant factor so there are no changes in the asymptotic behavior. We only show OMS and BFMS results corresponding $h \leq 9$, since for the other budgets upper and lower bounds can become equal in double precision, and the results are not meaningful.

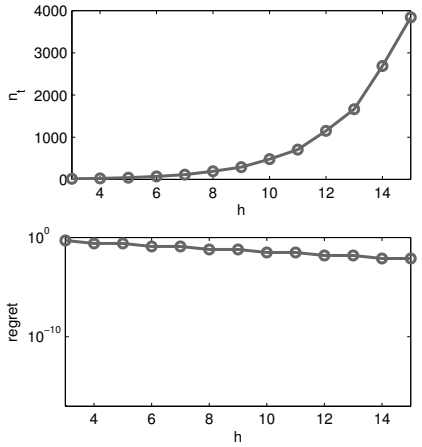

Fig. 5. Results of alpha-beta pruning for adversarial optimization.

Figure 6, top shows the depths reached by OMS and BFMS. Clearly, expanding nodes in the order of their importance is better than up to a fixed depth like in alpha-beta: the depths reached by OMS and BFMS, as well as the corresponding confidence in the solution, are much better. Finally, Figures 5 and 6, bottom show the near-optimality of

the returned solutions. This is measured here by the regret, defined for alpha-beta and BFMS as the distance between $v^* = 1$ and the value returned by the algorithm, and for OMS as half the size of the interval $[L(\mathbf{z}_0), B(\mathbf{z}_0)]$ at the root (equal to the average distance of L and B to v^*).

As expected from the analysis and the branching factor $\kappa = 1$ obtained in Example 4, OMS depths grow linearly with the computation budget and its regret shrinks exponentially with this depth. BFMS behaves surprisingly good: it often finds the optimal solution, and its depth also grows linearly with a larger slope than for OMS (in contrast, in alpha-beta n_t grows fast with h due to the exhaustive nature of the search, see again Figure 5, top). Unfortunately, unlike for OMS, a good behavior of BFMS cannot be guaranteed, and indeed BFMS is *inconsistent* over the class of problems satisfying Assumption 1, which means that for some problems it may entirely fail to converge to the optimal solution. The following counterexample illustrates this property.

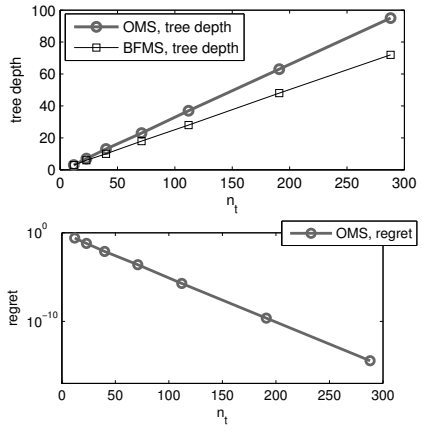

Fig. 6. Results of OMS and BFMS. BFMS regrets because they are 0 for all $n_t > 12$in this problem, so they are not shown.

Example 6: BFMS is inconsistent. Consider again adversarial optimization, Example 1, but with a different function $f(x, y)$, equal to 0.8 when $x \leq 0.5$, and $x + y$ otherwise. Take $l(\mathbf{z}) = b(\mathbf{z}) = 0.8$ for any box $\mathbf{z}$ in the left half of the domain, and use the bounds from Example 1 elsewhere. These l and b functions satisfy Assumption 1. BFMS develops the right (optimal) branch of the tree only until iteration 2, see Figure 7, and at any subsequent iteration it only expands nodes in the left branch of the root. Thus for any budget $n \geq 2$ it returns value 0.8, a constant away from the optimum $v^* = 1$. □

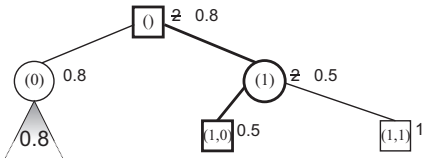

Fig. 7. BFMS tree in the counterexample. Struck-through values are those changed after iteration 1.

B. HIV infection control

We consider the HIV infection dynamics described by [1], with six state variables: T_1 and T_2 [cells/ml], the counts of healthy type 1 and type 2 target cells, T_1^{t} and T_2^{t} [cells/ml] the counts of infected type 1 and type 2 target cells, V the number of free virus copies [copies/ml], and E [cells/ml] the number of immune response cells. The system is controlled in discrete time with a sampling time of 5 days. In the strategy of structured treatment interruptions, two drugs are independently either fully administered (they are 'on'), or not at all (they are 'off'); thus there are two binary control variables u_1 and u_2, leading to $N_U = 4$. In other authors' work a one-to-one mapping was assumed between drug application and effectiveness. Here we use a variant we introduced in [6], where the effectiveness values ϵ_1 and ϵ_2 of the two drugs are, more realistically, uncertain by depending randomly on the inputs:

$$\epsilon_1 = \begin{cases} 0 & \text{w.p. 1, if } u_1 = 0 \\ 0.77 & \text{w.p. 0.5, if } u_1 = 1 \\ 0.63 & \text{w.p. 0.5, if } u_1 = 1 \end{cases}$$

$$\epsilon_2 = \begin{cases} 0 & \text{w.p. 1, if } u_2 = 0 \\ 0.33 & \text{w.p. 0.5, if } u_2 = 1 \\ 0.27 & \text{w.p. 0.5, if } u_2 = 1 \end{cases}$$

where "w.p." stands for "with probability". So depending on action u, there can be up to $N_W = 4$ possible outcomes. OMS is easy to modify for this varying-N_W case.

The system is initialized to the *unhealthy* equilibrium $x_{\mathrm{u}} = [163573, 5, 11945, 46, 63919, 24]^\top$, which represents a patient with dangerous infection levels and low immune response. STI is used to control the drugs such that the immune response of the patient is maximized and the number of virus copies is minimized, while penalizing the drugs administered due to their side-effects. We use the reward function of [1] and normalize it to $[0, 1]$. An ideal solution would drive the state to the *healthy* equilibrium $x_{\mathrm{h}} = [967839, 621, 76, 6, 415, 353108]^\top$, which represents a patient whose immune system controls the infection without the need of drugs.

OMS is applied to plan a solution in receding horizon, as explained at the end of Section III, treating the uncertainties as an opponent that aims to minimize the return like in Example 3. The budget is $n_t = 9000$, specified again as the number of created nodes, since the amount N_W of children of min nodes varies and using n would not result in a consistent computational load. The resulting trajectory is shown in Figure 8.[2] As hoped, the algorithm eventually stops administering drugs ($u_1 = u_2 = 0$), and the state reaches the healthy equilibrium x_{h} (although this particular solution has a 'lucky' disturbance realization for which the equilibrium is reached quickly).

[2]This experiment was run with a variant of OMS where the node to expand was directly selected to satisfy the interval inclusion property in Lemma 4, rather than by selecting bound extrema as in Section III.

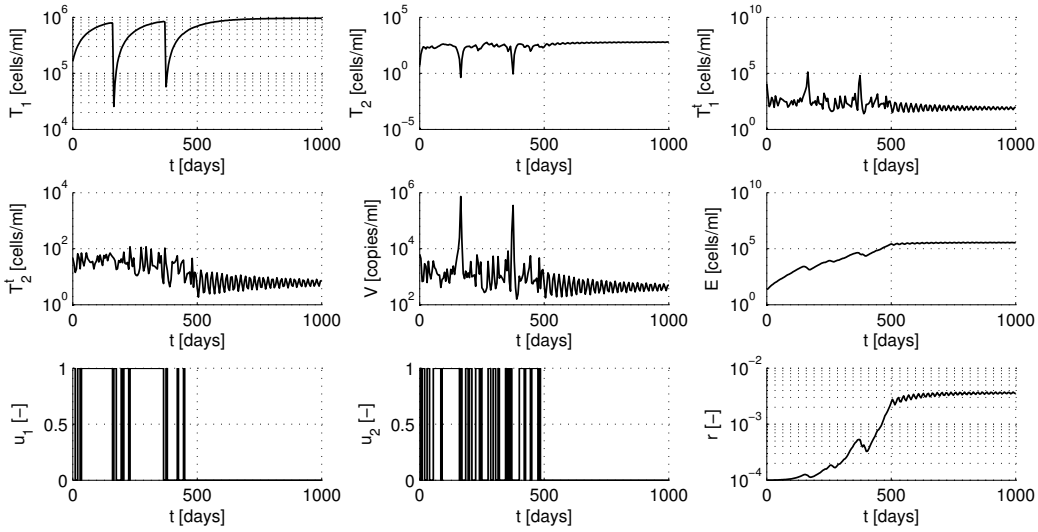

Fig. 8. HIV system controlled online with OMS. The trajectories of the six system states are shown on the top and middle rows, while the two applied actions and the (normalized) rewards obtained are shown on the bottom row.

VI. Conclusions

We have showed analytically that, under appropriate conditions, optimistic minimax search (also known as the best-first search variant of B*) converges in a well-characterized way towards the optimal minimax value, and illustrated the analysis in an empirical evaluation.

By requiring the upper and lower bounds, we have implicitly assumed knowledge about the smoothness of the value function. This can easily be obtained e.g. for discounted returns, but in general it may be too strong. Thus, the next big step is to check if it is possible to derive an algorithm that does *not* require knowing these bounds, only that they exist and get closer together as depth increases. This is similar to the idea behind simultaneous optimistic optimization in [15]. It would also be interesting to analyze other variants of B*, which performed better e.g. in the experiments of [17].

References

[1] B. Adams, H. Banks, H.-D. Kwon, and H. Tran, "Dynamic multidrug therapies for HIV: Optimal and STI control approaches," *Mathematical Biosciences and Engineering*, vol. 1, no. 2, pp. 223–241, 2004.

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

[3] 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.

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

[5] 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.

[6] 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.

[7] 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.

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

[9] 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.

[10] R. E. Korf, "Artificial intelligence search algorithms," in *Algorithms and Theory of Computation Handbook*, M. Atallah, Ed. CRC Press, 1998, pp. 1–20.

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

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

[13] J. M. Maciejowski, *Predictive Control with Constraints*. Prentice Hall, 2002.

[14] 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.

[15] 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.

[16] ——, "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.

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

[18] 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.

[19] A. Plaat, J. Schaeffer, W. Pijls, and A. de Bruin, "Best-first fixed-depth minimax algorithms," *Artificial Intelligence*, vol. 87, no. 1–2, pp. 255–293, 1996.

[20] S. Ratschan, "Search heuristics for box decomposition methods," *Journal of Global Optimization*, vol. 24, pp. 35–49, 2002.

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