Minimax Optimistic Planning for Continuous-Action Noncooperative Sequential Control

Sever Bogdan, Radu Herzal, Vineeth S. Varma, and Lucian Buşoniu, *Member, IEEE*

***Abstract*—We consider problems in which two competing agents influence in turn a dynamical system using continuous, scalar actions. One agent seeks to maximize an infinite-horizon value, while the other aims to minimize it, leading to a sequential noncooperative game. To search for the minimax-optimal solution, we introduce Minimax Optimistic Planning for Continuous actions (MOPC), an algorithm that iteratively refines the most promising regions of the space of action sequences. Our analysis shows that MOPC provides computable bounds on the solution quality, and that these bounds converge as computation grows at rates driven by a measure of problem complexity. We show how to pose in our framework – and thereby near-optimally solve – a class of problems in which opinions in a social network are influenced by two competing marketers. In experiments on such problems, MOPC outperforms uniform allocation over time.**

***Index Terms*—Optimal control, numerical algorithms, game theory.**

I. Introduction

CONSIDER a setting in which two adversarial agents influence a dynamical system in discrete time. The agents control two continuous, scalar, and bounded action signals that act in turn on the system, one of them aiming to maximize an infinite-horizon cumulative reward, and the other to minimize it. In game theory, this type of player interaction is called a sequential zero-sum game [13]. Besides board games like chess [18] or Go [25], this framework can be applied to security [20], robotics [24], cyber-physical systems [28], and training generative adversarial networks [26], etc. Closer to control, the framework applies to optimal control design under worst-case disturbances, similarly to H-infinity methods [4].

Received 17 March 2025; revised 7 May 2025; accepted 9 May 2025. Date of publication 19 May 2025; date of current version 3 June 2025. This work was supported in part by Project DECIDE, no. 57/14.11.2022, funded under the PNRR I8 Scheme by the Romanian Ministry of Research, Innovation, and Digitisation, and in part by CNCS/CCCDI-UEFISCDI project number PN-IV-P8-8.1-PRE-HE-ORG-2023-0012, within PNCDI IV. Recommended by Senior Editor C. Briat. *(Corresponding author: Lucian Buşoniu.)*

Sever Bogdan, Radu Herzal, and Lucian Buşoniu are with the Department of Automation, Technical University of Cluj-Napoca, 400114 Cluj-Napoca, Romania (e-mail: bogdan.vi.sever@student.utcluj.ro; herzal.eu.radu@student.utcluj.ro; lucian.busoniu@aut.utcluj.ro).

Vineeth S. Varma is with the Department of Automation, Technical University of Cluj-Napoca, 400114 Cluj-Napoca, Romania, and also with the Université de Lorraine, CNRS, CRAN, 54000 Vandoeuvre-les-Nancy, France (e-mail: vineeth.satheeskumar-varma@univ-lorraine.fr).

Digital Object Identifier 10.1109/LCSYS.2025.3571642

Many methods exist to solve zero-sum sequential games, and covering the full field here would be impossible. Continuous-action methods include gradient techniques for convex-concave or more general utilities [26], value iteration [12], policy iteration [6], reinforcement learning [1], and so on. Our method is different from these since it explores a tree representation of possible action sequences locally for one state, closer to [16] which was given for a different type of game. This also means that our convergence rates are different, driven by the size of the near-optimal set of sequences at the current state, rather than, e.g., by the finite state-action space size like usual sample complexity bounds in RL [27]. Faster rates can be achieved under convex-concave utilities [11], and sometimes efficient partial solvers are assumed [12]; in contrast, MOPC provides a fully implementable solution, for general Lipschitz dynamics and non-concave/convex rewards.

To approximate the minimax-optimal solution, we design a new algorithm called Minimax Optimistic Planning for Continuous actions (MOPC). MOPC iteratively partitions the set of infinite-horizon max-min action sequences, and associates to each subset (box) in the partition lower and upper bounds on the values of sequences in that box. Boxes are organized into a tree, and at each iteration, the method further refines an optimistic leaf box that maximizes upper bounds at max nodes and minimizes lower bounds at min nodes. Refinement splits the box along the dimension (action) with the largest contribution to the gap between the lower and upper bounds. The refined horizon thus grows adaptively. Our main results show that the solution returned of MOPC is close to the minimax-optimal value to an extent that (i) can directly be computed in the algorithm, as the smallest gap of any expanded node, and (ii) converges to zero at well-characterized rates as invested computation increases. These rates are modulated by a complexity measure formalized as a branching factor of a near-optimal subtree.

MOPC belongs to the class of optimistic planning (OP) methods [17], among which the most relevant here are discrete-action minimax OP [8], [10] and single-agent, max-only Continuous-action OPC [9]. MOPC essentially combines the strengths of these two techniques, to obtain a novel method able to handle continuous-action minimax problems. The key difference from [8], [10] is the overhaul to continuous actions, which alters the structure of the space and tree explored, so that, e.g., whereas before the sequence of refinements was max-min-max-min and so on, now the order is much more complicated, but still predictable. The max-only setting of [9] is a special case of the problem here, in which the min agent is removed. In graph search, MOPC is related to classical

2475-1456 © 2025 IEEE. All rights reserved, including rights for text and data mining, and training of artificial intelligence and similar technologies. Personal use is permitted, but republication/redistribution requires IEEE permission. See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Universitatea Tehnica din Cluj-Napoca. Downloaded on October 09,2025 at 06:16:22 UTC from IEEE Xplore. Restrictions apply.

methods like B* [5], alpha-beta pruning [14], and best-first search [15], which again have discrete actions. Our complexity measure is related to the branching factors in [8], [9], of alpha-beta pruning [14], and of other OP methods [17]. It arises however in a different way from the underlying continuous-action minimax problem. OP also relates to interval arithmetic for branch-and-bound optimization [2], which has been applied to minimax problems [22].

For the experiments, we consider problems in which two competing marketers influence the opinions in a social network over several campaigns. Marketing budgets chosen with MOPC outperform uniform allocation over time.

II. Problem Statement and Preliminaries

Consider a zero-sum sequential game where a max(imizer) agent and a min(imizer) agent control in turn, in discrete time, two signals: max action $u_k \in \mathbb{R}$ and min action $w_k \in \mathbb{R}$.

Assumption 1: Scalar signals u and w are bounded to $[0, 1]$.

Actions are often saturated in practice, so then rescaling them to $[0, 1]$ does not lose generality. Our method may be extended to vector actions at extra computational cost; a naive extension is exponential in the number of action elements, which can be improved by refining elements in turn, see [9] (supplementary material) for a single-agent discussion.

We usually treat max and min actions uniformly, denoting them by $z \in Z := [0, 1]$. Define also step index h that counts all decisions, max or min. Thus, h advances twice as fast as k, which only increases with pairs of max-min decisions: $z_0 = u_0, z_1 = w_0, z_2 = u_1, z_3 = w_1, \ldots, z_{2k} = u_k, z_{2k+1} = w_k, \ldots$. Denote an infinite sequence of actions by $\boldsymbol{z} = (z_0, z_1, z_2, z_3, \ldots, z_{2k}, z_{2k+1}, \ldots) = (u_0, w_0, u_1, w_1, \ldots, u_k, w_k, \ldots) \in Z^\infty$.

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 generic action, and $f : X \times Z \to X$ are the dynamics. X may be any metric space, such as $\mathbb{R}^{n_x}$ of dimension n_x. A reward $r_h = \rho(x_h, z_h)$ is assigned, where $\rho : X \times Z \to \mathbb{R}$. Given an initial state x_0, the infinite-horizon discounted value of sequence $\boldsymbol{z}$ is:

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

where $\gamma \in (0, 1)$ is the discount factor. We aim to achieve the infinite-horizon minimax-optimal value:

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

i.e., the best value of the max agent under worst-case min actions. As the strategy space of each agent is a convex subset of a linear topological space, Sion's minmax theorem guarantees solution existence [21]. Multiple sequences may achieve v^*, so, e.g., concave-convex rewards are not required.

Assumption 2: (i) Dynamics and rewards are Lipschitz, i.e., $\exists L_f, L_\rho$ so that $\forall x, x' \in X, z, z' \in Z$:

$$\begin{aligned} \|f(x, z) - f(x', z')\| &\le L_f(\|x - x'\| + |z - z'|) \\ |\rho(x, z) - \rho(x', z')| &\le L_\rho(\|x - x'\| + |z - z'|) \end{aligned}$$

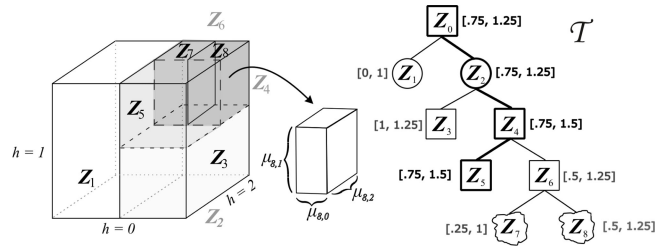

Fig. 1. Left: Example refinement of Z^∞, after 4 splits, with $M = 2$. Dimensions 4 and higher are not shown since they are not split yet. Boxes that have already been split further $(\boldsymbol{Z}_2, \boldsymbol{Z}_4, \boldsymbol{Z}_6)$ are shown in shades of gray. Right: Corresponding tree, where max nodes are square, min nodes are round, and the type of wavy-outline nodes is not yet defined. Lower and upper bounds $[L_i, B_i]$ are shown near tree nodes.

(ii) Rewards $\rho(x, z)$ are in $[0, 1]$ for all $x \in X, z \in Z$, and $\gamma L_f < 1$.

Assumption 2 is needed to make values v well-behaved; in particular it allows us to provide a bound on the difference in values between two action sequences $\boldsymbol{z}$ and $\boldsymbol{z}'$:

$$|v(\boldsymbol{z}) - v(\boldsymbol{z}')| \le L_v \sum_{h=0}^{\infty} \gamma^h |z_h - z'_h| \tag{4}$$

where the Lipschitz constant of v is $L_v = \frac{L_\rho}{1-\gamma L_f}$. Such a property was proven in [9] (suppl.) for max-only problems with $z = u$, but since we are not yet optimizing over actions, and instead simply consider two fixed sequences, the property also holds in the minimax case, in the form (4).

III. Algorithm

The goal of the algorithm is to find a finite-horizon sequence that approximately achieves v^*, where the horizon grows adaptively with computation budget n measured by the number of calls to f and ρ. To this end, we iteratively refine the infinitely-dimensional hypercube Z^∞, where each dimension h is the action at step h (hence the same notation). This set is iteratively split into smaller hyperrectangles, "boxes" for short, each of which gets a unique index i and has a finite number of refined dimensions (horizon) H_i. By convention, $\boldsymbol{Z}_0 = Z^\infty$. A box is denoted $\boldsymbol{Z}_i \subseteq Z^\infty$ and is the Cartesian product of a sequence of intervals $(\mu_{i,0}, \ldots, \mu_{i,H_i-1}, Z, Z, \ldots)$ where $\mu_{i,h} \subseteq Z$ and the largest refined dimension is $H_i - 1$; for all subsequent dimensions $\mu_{i,h} = Z$. A box is refined by splitting into $M > 1/\gamma$ equal pieces the interval of some dimension $h \le H_i$, which corresponds to refining z_h more finely. Define $\delta_{i,h}$ to be the length of the interval $\mu_{i,h}$ of box $\boldsymbol{Z}_i$, $z_{i,h}$ a sample action taken at the center of this interval, and the finite center sequence $\boldsymbol{z}_i = (z_{i,0}, \ldots, z_{i,H_i-1})$. For each box, the finite sequence of rewards $r_{i,h}$ obtained by applying $\boldsymbol{z}_i$ from x_0 is computed by simulating with f and ρ, leading to the center-sequence value truncated to the box horizon:

$$R_i = \sum_{h=0}^{H_i-1} \gamma^h r_{i,h} \tag{5}$$

Boxes are arranged in a tree $\mathcal{T}$, where a parent node is split into its children as explained above. Denote by $\mathcal{L}(\mathcal{T})$ the leaf nodes of this tree, and by $\mathcal{C}_i$ the M children of node i. Fig. 1 exemplifies a collection of boxes

Authorized licensed use limited to: Universitatea Tehnica din Cluj-Napoca. Downloaded on October 09,2025 at 06:16:22 UTC from IEEE Xplore. Restrictions apply.

(partition of Z^∞) and the corresponding tree. Take box $\boldsymbol{Z}_4$, which was obtained by splitting twice, first along dimension $h = 0$ and then along $h = 1$, so $H_4 = 2$ and the box has the following form: $(\mu_{4,0}, \mu_{4,1}, Z, Z, \dots,) = ([0.5, 1], [0.5, 1], [0, 1], [0, 1], \dots,)$. Box $\boldsymbol{Z}_4$ is then split further, along dimension 2, resulting in 2 new children: $\boldsymbol{Z}_5$ and $\boldsymbol{Z}_6$. These children inherit the intervals for all dimensions, except the one that was split, for which the two new intervals are $\mu_{5,2} = [0, 0.5]$ and $\mu_{6,2} = [0.5, 1]$. Since we are splitting along a new dimension, $H_5 = H_6 = H_4 + 1 = 3$. Importantly, whether a node i is min or max can only be determined a posteriori, depending on the dimension along which i has been split (max if h is even, min if h is odd). Since the boxes at the last depth have not been split, their type is not yet determined.

Define the infinite-horizon value v_i of sequences in box i:

$$\lim_{k\to\infty}\left[\max_{u_0\in\mu_{i,0}} \min_{w_0\in\mu_{i,1}} \cdots \max_{u_k\in\mu_{i,2k}} \min_{w_k\in\mu_{i,2k+1}} \sum_{h=0}^{2k+1} \gamma^h \rho(x_h, z_h)\right] \tag{6}$$

Note the differences from the minimax-optimal value v^*, defined over the full action space in (3), and also that definition (6) is novel to the continuous-action minimax problem, generalizing the discrete-action version in [8]. While v_i is not computable and we only use it in the analysis, we next exploit (4) to establish computable, finite-horizon lower and upper bounds on v_i at leaf boxes i.

Consider any infinite sequence $\bar{z} \in \boldsymbol{Z}_i$, and construct another infinite sequence $z_i^+ = (z_{i,0}, \dots, z_{i,H_i-1}, \bar{z}_{H_i}, \bar{z}_{H_i+1}, \dots)$ that appends to the center sequence z_i the tail of $\bar{z}$ from H_i onwards. Then, $v(\bar{z})$ is lower-bounded by:

$$\begin{aligned} L_i &:= v(z_i^+) - L_v \sum_{h=0}^{H_i-1} \gamma^h \left|\bar{z}_h - z_{i,h}\right| \\ &\geq R_i - L_v \sum_{h=0}^{H_i-1} \gamma^h \delta_{i,h}/2 \end{aligned} \tag{7}$$

where the first step follows from metric (4) and the equality of $\bar{z}$ to z_i^+ from H_i onwards; and in the second step, we truncated $v(z_i^+)$ to its first H_i terms, obtaining R_i, since all subsequent terms are positive by Assumption 2(ii). We also made $\left|\bar{z}_h - z_{i,h}\right|$ explicit, knowing that $z_{i,h}$ is at the center of its interval. Symmetrically, $v(\bar{z})$ is upper-bounded by:

$$B_i := R_i + L_v \sum_{h=0}^{H_i-1} \gamma^h \delta_{i,h}/2 + \frac{\gamma^{H_i}}{1-\gamma} \tag{8}$$

where instead of truncating $v(z_i^+)$, we replaced rewards from H_i onwards with their upper bounds 1, obtaining the geometric sum $\frac{\gamma^{H_i}}{1-\gamma}$. Since L_i and B_i bound the value of any $\bar{z} \in \boldsymbol{Z}_i$, they also bound v_i from (6) in particular. To keep derivations manageable, we use 0 and 1-reward tails for the remainder of the sequence, but these tails may be replaced by tighter lower and upper bounds on the values of the state at step H_i, leading to an improved, bootstrapping method [7].

Starting from L_i and B_i at leaves, the bounds are backed-up through the tree for inner nodes. Introduce an operator maxmin that applies max or min depending on the node type where it is called. Lower bounds are backed up with:

$$L_i = \text{maxmin}_{c\in\mathcal{C}_i} \; L_c := \begin{cases} \max_{c\in\mathcal{C}_i} L_c, & \text{if } i \text{ max} \\ \min_{c\in\mathcal{C}_i} L_c, & \text{if } i \text{ min} \end{cases} \tag{9}$$

Upper bounds are backed up using the same operator:

$$B_i = \text{maxmin}_{c\in\mathcal{C}_i} \; B_c \tag{10}$$

Note that a node may inherit L from one child and B from another, as seen for $\boldsymbol{Z}_2$ in Fig. 1.

To choose the next leaf to expand, the algorithm 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. After exhausting computation n, the algorithm stops and returns the center sequence and the bounds of a deepest expanded node. In the example tree of Fig. 1, the thick lines illustrate the optimistic path; e.g., node 5 is added to the path after 4 because 4 is a max node and $B_5 > B_6$.

To specify along which dimension a leaf box i is split after being selected, start by deriving an upper bound δ_i on the gap between B_i (8) and L_i (7):

$$\begin{aligned} B_i - L_i = L_v \sum_{h=0}^{H_i-1} \gamma^h \delta_{i,h} \\ + \frac{\gamma^{H_i}}{1-\gamma} \leq \bar{L}_v \sum_{h=0}^{\infty} \gamma^h \delta_{i,h} =: \delta_i \end{aligned} \tag{11}$$

where we used $\delta_{i,h} = 1$ for $h \geq H_i$ and for convenience we rewrote the equation as a summation of uniform terms for all dimensions, after replacing L_v from (4) with $\bar{L}_v = \max\{L_v, 1\}$. Quantity δ_i is the uncertainty on the values of sequences in box i. A box i is split along the dimension with the largest contribution to this uncertainty, denoted by $h_i^\dagger$:

$$h_i^\dagger \in \underset{h=0,\dots,H_i}{\arg\max} \; \gamma^h \delta_{i,h} \tag{12}$$

where ties are broken in favor of the smallest h. We stop at H_i because all subsequent dimensions have a smaller contribution.

This selection rule leads to a predictable, well-defined order of dimensions and therefore action types (max or min) to split, and to boxes of the same shape at a given depth d; note we may split a given dimension several times. Thus, depth d_i of a box i is *not* the horizon H_i, but the cumulative number of splits of all dimensions $h < H_i$, so $d_i > H_i$ everywhere except close to the root. Moreover, unlike in [8] where even depths had max actions and odd depths min actions, here no such simple relationship holds. E.g., in Fig. 1, to obtain $\boldsymbol{Z}_7$ and $\boldsymbol{Z}_8$, the order of splits along depths d was <u>max</u>-min-max-<u>max</u> and dimension 0 was split twice (underlined <u>max</u>es). Despite this, any sequence in any box will still alternate actions in the normal max-min order along dimensions h.

We call the new algorithm Minimax OP for Continuous actions (MOPC). It is a novel hybrid that combines the strengths of discrete-action optimistic minimax search of [8] and of the single-agent (max-only) continuous-action method of [9], so as to handle continuous-action minimax problems. Alg. 1 summarizes MOPC, optimizing bound backups so they are only performed for nodes whose bounds are impacted by the last expansion. By taking M odd so as to reuse the simulated transitions of the center sequence, expanding box i requires $(M-1)(H_i - h_i^\dagger)$ transitions when $h_i^\dagger < H_i$, or M when $h_i^\dagger = H_i$. The main output is the near-optimal sequence z_{i^*}, which can be used for control. This may often be done in receding-horizon, by applying $z_{i,0}$, observing the action of the min agent and its resulting state, and calling MOPC again

Authorized licensed use limited to: Universitatea Tehnica din Cluj-Napoca. Downloaded on October 09,2025 at 06:16:22 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1 MOPC

Input: $f, \rho, L_v, \gamma, M, n$

initialize $\mathcal{T}$ with root 0 labeled by $\mathbf{Z}_0$; $i \leftarrow 0$; $i^* \leftarrow 0$
while computation n not exhausted **do**
  while $i \notin \mathcal{L}(\mathcal{T})$ **do** ▷ find optimistic leaf
    $i \leftarrow \begin{cases} \arg\max_{c \in \mathcal{C}_i} B_c, & \text{if } z \text{ max node} \\ \arg\min_{c \in \mathcal{C}_i} L_c, & \text{if } z \text{ min node} \end{cases}$
  split i along $h_i^\dagger$, add its children c to $\mathcal{T}$ ▷ expand it
  increase n by number of newly simulated transitions
  for each new c, compute L_c and B_c with (7) and (8)
  if $d_i \geq d_{i^*}$, $i^* \leftarrow i$ ▷ update deepest expanded node
  while $i \neq 0$ **do** ▷ backup bounds up the tree
    $L_i = \text{maxmin}_{c \in \mathcal{C}_i} L_c$; $B_i = \text{maxmin}_{c \in \mathcal{C}_i} B_c$
    $i \leftarrow$ parent of i

Output: box i^* and its features (notably, z_{i^*})

there. Meanwhile, the min agent symmetrically applies MOPC or another algorithm to choose its own action.

IV. Near-Optimality of MOPC

First, we will show that MOPC only expands the boxes that are necessary in a well-defined sense, forming a near-optimal subtree $\mathcal{T}^*$. A near-optimality bound for the returned box i^* will then be provided that is readily computable in the algorithm. Finally, by introducing a branching factor of $\mathcal{T}^*$ to characterize its size, we will provide a convergence rate of the near-optimality bound to zero as computation n grows.

Lemma 1: At depth d, MOPC only expands boxes $i^\dagger$ in set:

$$\mathcal{T}_d^* := \left\{ i^\dagger \text{ at depth } d \,\middle|\, \left|v^* - v_j\right| \leq \delta_d, \; \forall j \text{ on path from root to } i^\dagger \right\} \quad (13)$$

where by a slight abuse of notation, δ_d denotes the uncertainty gap of all nodes at depth d (which all have the same shape).

Proof: Show first that (i) $v_i \in [L_i, B_i]$ for any node $i \in \mathcal{T}$ by induction from the leaves. The property holds at leaves by definition (7), (8). Take first a max inner node i. Recall $h_i^\dagger$ is the (max) dimension along which i is split. We have $\mu_{i,h_i^\dagger} = \bigcup_{c \in \mathcal{C}_i} \mu_{c,h_i^\dagger}$. Then, $\forall k > h_i^\dagger/2$:

$$\max_{u_0 \in \mu_{i,0}} \cdots \max_{z_h \in \mu_{i,h_i^\dagger}} \cdots \min_{w_k \in \mu_{i,2k+1}} \sum_{h=0}^{2k+1} \gamma^h \rho(x_h, z_h)$$
$$= \max_c \left[\max_{u_0 \in \mu_{i,0}} \cdots \max_{z_h \in \mu_{c,h_i^\dagger}} \cdots \min_{w_k \in \mu_{i,2k+1}} \sum_{h=0}^{2k+1} \gamma^h \rho(x_h, z_h) \right]$$

and therefore $v_i = \max_{c \in \mathcal{C}_i} v_c$. Take a child c so that $v_i = v_c \leq B_c \leq B_i$; and another child c' so that $L_i = L_{c'} \leq v_{c'} \leq v_i$. Note $v_c \leq B_c$ and $L_{c'} \leq v_{c'}$ are true by the induction hypothesis. Overall, $v_i \in [L_i, B_i]$. At min nodes i, the derivation is symmetrical, e.g., $v_i = \min_{c \in \mathcal{C}_i} v_c$.

Next, show that (ii) $[L_j, B_j] \subseteq [L_c, B_c]$ for the child c of j on the path to selected node $i^\dagger$ in (13). If j is max, $L_j \geq L_c$ and $B_j = B_c$ due to (9), (10). Conversely, if j is min, $L_j = L_c$ and $B_j \leq B_c$. Advancing to $i^\dagger$, we get $[L_j, B_j] \subseteq [L_{i^\dagger}, B_{i^\dagger}]$.

Now, $v^* = v_0 \in [L_0, B_0] \subseteq [L_{i^\dagger}, B_{i^\dagger}]$ by (i), (ii) in turn, and $v_j \in [L_j, B_j] \subseteq [L_{i^\dagger}, B_{i^\dagger}]$. Finally, $B_{i^\dagger} - L_{i^\dagger} \leq \delta_d$. ■

This proof extends the discrete-action of [8, Lemma 5], by accounting for the different nature of minimax values v_i of a box i. The intuition behind Lemma 1 is that MOPC will never expand a "clearly suboptimal" node for which either its own value v_i, *or the value of any of its ancestors* v_j, is further away from v^* than the uncertainty δ_d. Note that the property is surprisingly tight, in the sense that it applies to all nodes j on the path with the smallest (deepest) uncertainty δ_d, rather than with their own, larger uncertainty.

Theorem 1: The minimax-optimal value v^* (3), the value of the returned box v_{i^*} (6), and the center-sequence value R_{i^*} (5) are all in the interval $[L_{i^*}, B_{i^*}]$, which is of length δ_{i^*}.

Proof: $v_{i^*} \in [L_{i^*}, B_{i^*}]$ by definition, and $|v^* - v_{i^*}| \leq \delta_{i^*}$ by applying Lemma 1 since i^* was expanded; these relations also hold in the discrete-action case [8]. The center-sequence value R_{i^*} is a concept specific to continuous actions, so we must consider it separately; nevertheless, $R_{i^*} \in [L_{i^*}, B_{i^*}]$ by the definition of these bounds. Finally, $B_{i^*} - L_{i^*} = \delta_{i^*}$. ■

Theorem 1 provides a near-optimality bound δ_{i^*} that is directly available to the algorithm once it has run. While the infinite-horizon value v_{i^*} is unknown, R_{i^*} is an already-computed, readily accessible estimate for it.

Note next that at depth d, since sets have a similar shape to the max-only case, we get from [9] (suppl.):

$$\delta_d \leq c_\delta \sqrt{2d(\tau - 1)} \gamma^{\sqrt{2d\frac{\tau-1}{\tau^2}}} = \tilde{O}\left(\gamma^{\sqrt{2d\frac{\tau-1}{\tau^2}}}\right) \quad (14)$$

where γ is the discount factor from (2), $\tau = \lceil \frac{\log M}{\log 1/\gamma} \rceil$, c_δ is a positive constant and $\tilde{O}$ disregards asymptotically negligible constants and logarithmic factors.

Define next a measure of the size of the near-optimal tree $\mathcal{T}^*$ from Lemma 1, which dictates the complexity of the continuous-action minimax problem; $|\cdot|$ denotes cardinality.

Definition 1: The asymptotic branching factor of $\mathcal{T}^*$ is the smallest number κ so that $\exists C$ for which $\left|\mathcal{T}_d^*\right| \leq C\kappa^d, \forall d \geq 0$.

We are now ready to provide the convergence rates.

Theorem 2: As computation n increases, MOPC near-optimality approaches 0 as follows:

$$\delta_{i^*} = \begin{cases} \tilde{O}\left(\gamma^{\sqrt{\frac{2(\tau-1)\log n}{\tau^2 \log \kappa}}}\right) & \text{when } \kappa > 1 \\ \tilde{O}\left(\gamma^{n^{1/4}\sqrt{\frac{2(\tau-1)}{\tau^2\sqrt{MC}}}}\right) & \text{when } \kappa = 1 \end{cases} \quad (15)$$

Proof: To ensure i^* is at depth at least d, when $\kappa > 1$, take the smallest d so that $Cm^{d+1}Md \geq n$, where Cm^{d+1} bounds the size $\mathcal{T}^*$ up to d, and Md bounds the number of calls to f and ρ to expand a node at d. We get $d = \tilde{\Omega}(\frac{\log n}{\log \kappa})$ where $\tilde{\Omega}$ bounds from below up to logarithmic factors. Replacing this in (14) obtains – after tedious algebra – the result. When $\kappa = 1$, take d so that $Cd^2M \geq n$ since now $\mathcal{T}^*$ has up to C nodes at each depth. We get $d \geq \sqrt{n/CM}$, and the rate via (14). ■

While the proof and rates are similar to that of the max-only case [9] (suppl.), the structure of the space and tree, and the way κ arises from f and ρ, are novel and different from earlier OP methods. In the simplest problems, $\kappa = 1$, and Theorem 2 says that near-optimality converges fast to zero, roughly exponentially in $n^{1/4}$. In general, κ is non-integer, and larger in more difficult problems. Rates are slower than for discrete actions [8] due to the larger continuous-sequence space; e.g., for $\kappa = 1$, the rate in [8] was $O(\gamma^{n/a})$ with a constant.

To get insight into how κ relates to the problem at hand, construct two examples. In the first, we expect κ to be small.

Authorized licensed use limited to: Universitatea Tehnica din Cluj-Napoca. Downloaded on October 09,2025 at 06:16:22 UTC from IEEE Xplore. Restrictions apply.

Take dynamics $x_k + \ell(u_k - w_k)$, saturated to some bounded interval, and a reward function that only depends on, and increases with, x. Then, the minimax-optimal solution is $u_k = 1$, $w_k = 1$, and if the reward function has a large enough slope in x MOPC may clearly distinguish this minimax-optimal solution and eliminate most others, leading to a small κ. The second example is a difficult problem in which $\kappa = M$, obtained by setting all the rewards to be same, making solutions indistinguishable.

It is furthermore instructive to examine more deeply the relation to the max-only OPC method of [9]. To do this, we slightly deviate from Section II and create a single-valued w, meaning that the contribution of dimensions w_k to the gap δ is always 0 and w_k will never be split. At any refinement of box i along u_k at depth d, for any child c at $d+1$ immediately add a single next child at $d+2$ corresponding to the resulting next state x_{k+1} "following" the unique w. We obtain an MOPC tree that at even depths d contains the sets that the OPC tree would contain at depths $d/2$, so a branching factor of β in OPC means $\mathcal{T}_d^*$ contains $O(\beta^{d/2}) = O(\sqrt{\beta}^d)$ nodes, leading to a minimax branching factor $\kappa = \sqrt{\beta}$.

V. Application to Marketing in Duopolies

As an example, we will now instantiate our framework for a class of problems involving long-term marketing resource allocation over a social network, in a duopoly scenario. For this, like in [23], we consider a market with two companies, $m = \{1, 2\}$, competing over a social network of N consumers. To prevent confusion with the max-min agent terminology above, we will call here the consumers *nodes*, and the maximizer and minimizer *marketers* (1 and 2, respectively). The set of nodes is denoted by $\mathcal{V} = \{1, 2, \ldots, N\}$, and the network is represented by a fixed weighted directed graph $(\mathcal{V}, \mathcal{E})$, where $\mathcal{E}$ contains the edges of the graph. Each node $o \in \mathcal{V}$ has a continuous, normalized opinion assigned to it, $x_o(t) \in [0, 1]$. The state vector $x(t) = (x_1(t), x_2(t), \ldots, x_N(t))^T$ collects all opinions at continuous time t.

The marketers run a series of campaigns at equally spaced time instants, $t_0, t_2, \ldots, t_k, \ldots$, with period $T := t_{k+1} - t_k$, in principle over an infinite horizon. At each campaign, marketer 1 chooses a budget $u_k \in [0, B_1]$ to be spent over the network, to maximize a utility described by a reward signal defined below, whilst marketer 2 chooses a budget $w_k \in [0, B_2]$, with the opposite goal. Each marketer distributes these budgets over the network, according to the action vector $a_m(k) = (a_{m,1}(k), a_{m,2}(k), \ldots a_{m,N}(k))$, with $a_{m,o}(k) \geq 0$, $\sum_{o=1}^N a_{1,o}(k) \leq u_k$, and $\sum_{o=1}^N a_{2,o}(k) \leq w_k$. These action vectors are computed as a Nash equilibrium for $x(t_k)$ with the method in [23], leading to $(a_1(k), a_2(k)) = F(u_k, w_k)$. Since we consider a zero-sum game with compact and convex action spaces, this Nash solution for the per-node allocation with a given campaign budget u_k, w_k solves the minimax sequential subgame at campaign k [19]. So, once the temporal allocation is determined, per-node allocations will be identical for marketers playing sequentially or simultaneously.

Given $a_1(k), a_2(k)$, node opinions evolve as follows:

$$\begin{cases} \dot{x}(t) = -\mathtt{L}x(t) & \forall t \in \mathbb{R} \setminus \{t_k | k \geq 0\} \\ x(t_k^+) = \Phi(x(t_k), a_1(k), a_2(k)) & \forall t_k \end{cases} \tag{16}$$

where $\mathtt{L}$ is the Laplacian of the graph and $\Phi(x(t_k), a_1(k), a_2(k)) \in [0, 1]^N$ is a vector that collects the opinions immediately after a campaign, with its elements defined as:

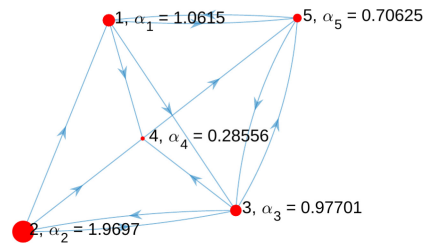

Fig. 2. A small graph example. The size of each node is proportional to its impact α_o, which is also given numerically near each node.

$$\phi\big(x_o(t_k), a_{1,o}(k), a_{2,o}(k)\big) = \frac{x_o(t_k) + a_{1,o}(t_k)}{1 + a_{1,o}(t_k) + a_{2,o}(t_k)} \tag{17}$$

The simultaneous-action dynamics are defined so that $\tilde{f}(x(t_k), u_k, w_k) =: x(t_{k+1})$ from (16), and the rewards are:

$$\tilde{\rho}\big(x(t_k^+), F(u_k, w_k)\big) = \alpha x\big(t_k^+\big) - \lambda_1 \mathbf{1}_N^T a_1(k) + \lambda_2 \mathbf{1}_N^T a_2(k)$$

where $\mathbf{1}_N$ is the column vector of N ones and $\lambda_1 \geq 0, \lambda_2 \geq 0$ weigh the advertising costs of the two marketers. Vector $\alpha = \mathbf{1}_N^T \exp(-\mathtt{L}T)$ describes the impact of the nodes on the resulting next-campaign state $x(t_{k+1})$. The objective is to find minimax-optimal sequences of actions u_k, w_k.

To rewrite the problem in our sequential setting, add like in [10] a new special state y that is -1 at max steps and remembers u at min steps, and define the dynamics and rewards in the form from Section II:

$$f\Big([x^\top, y]^\top, z\Big) = \begin{cases} [x^\top, z]^\top & \text{if } y = -1 \\ [\tilde{f}(x^\top, y, z)]^\top & \text{otherwise} \end{cases}$$

$$\rho\Big([x^\top, y]^\top, z\Big) = \begin{cases} 0 & \text{if } y = -1 \\ \text{norm}\big[\tilde{\rho}\big(x^\top, F(y, z)\big)\big] & \text{otherwise} \end{cases}$$

where 'norm' normalizes the rewards to $[0, 1]$.

We showed so far how to phrase in our framework *any* marketing problem from the class considered, so we can apply MOPC to near-optimally solve it. Next, we give numerical results for two example graphs. We start with the $N = 5$-node graph from Fig. 2, so that trajectories are easier to interpret. Both marketers are given equal maximum budgets ($B_1 = B_2 = 1$), and equal costs of advertising ($\lambda_1 = \lambda_2 = 0.8$). The initial opinion of the nodes is chosen randomly from the interval $[0, 0.3]$, favorable for marketer 2. To run MOPC, we set $\gamma = \sqrt{0.8}$, $L_v = 5$, $M = 3$, use 10 campaigns with period $T = 1$ time units, and computation effort $n = 5000$.

Fig. 3 shows the results when both marketers apply MOPC. Since initial opinions are close to zero, max applies large budgets early on, while min starts reacting with larger budgets only later, when opinions increase. As expected from [23], in the per-node allocation more influential nodes get more budget (see again node impacts in Fig. 2). The total reward across all campaigns is 18.77. We estimate the complexity measure κ of the initial state by the average branching factor on the tree developed by max for a large budget $n = 10^5$, and obtain 2.18. Thus we are on the first, large-complexity convergence rate branch of (15).

To obtain a baseline and check whether MOPC is useful, we next replace it for max with uniform allocation over time of the same total budget (still with Nash allocation over nodes), leading to the results in Fig. 4. The total reward is now 18.10

Authorized licensed use limited to: Universitatea Tehnica din Cluj-Napoca. Downloaded on October 09,2025 at 06:16:22 UTC from IEEE Xplore. Restrictions apply.

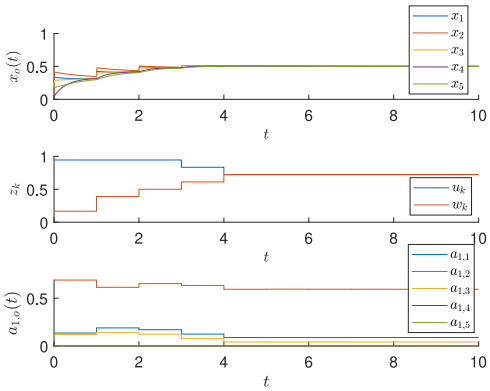

Fig. 3. Solution when both marketers apply MOPC. Top: Opinion trajectories. Middle: Budgets allocated across campaigns. Bottom: Per-node allocations of marketer 1.

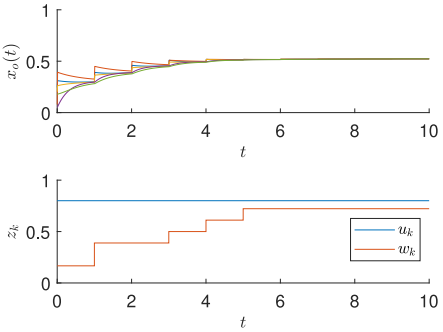

Fig. 4. Solution when marketer 1 applies a uniform budget allocation over time, while marketer 2 still applies MOPC. Allocations to nodes are skipped due to space limits.

and the opinions reach larger values later, a worse performance than with MOPC.

Finally, we illustrate MOPC for a larger graph, containing 50 nodes. The graph is generated by randomly removing 25% of the edges of a Barabási–Albert model [3], without violating connectivity. Both marketers use MOPC with $B_1 = B_2 = 10$, $n = 1000$, and other values are unchanged from above. The total reward is 181.39, better than the value 175.78 when max applies uniform allocation (trajectories not shown).

VI. Conclusion and Future Work

An optimistic planning algorithm was introduced for infinite-horizon minimax games, with near-optimality and convergence rate guarantees. We also showed how to apply the algorithm to a class of problems where two marketers aim to sway opinions in a social network, and provided numerical results in this setting. Future work will consider applications in other fields, such as control under disturbances. It may also be possible to derive a minimax version of the Lipschitz-constant-free, simultaneous variant of the max-only planner from [9] that often performs better in practice.

References

[1] A. Al-Tamimi, F. L. Lewis, and M. Abu-Khalaf, "Model-free Q-learning designs for linear discrete-time zero-sum games with application to H-infinity control," *Automatica*, vol. 43, no. 3, pp. 473–481, 2007.

[2] I. Araya and V. Reyes, "Interval branch-and-bound algorithms for optimization and constraint satisfaction: A survey and prospects," *J. Global Optim.*, vol. 65, pp. 837–866, 2016.

[3] A. Barabási, H. Jeong, Z. Néda, E. Ravasz, A. Schubert, and T. Vicsek, "Evolution of the social network of scientific collaborations," *Physica A, Stat. Mech. Appl.*, vol. 311, nos. 3–4, pp. 590–614, 2002.

[4] T. Başar and P. Bernhard, *H∞-Optimal Control and Related Minimax Design Problems: A Dynamic Game Approach*. Boston, MA, USA: Birkhauser, 1995.

[5] H. Berliner, "The B⋆ search algorithm: A best first proof procedure," *Artif. Intell.*, vol. 12, pp. 79–87, Jan. 1979.

[6] D. Bertsekas, "Distributed asynchronous policy iteration for sequential zero-sum games and minimax control," 2021, *arXiv:2107.10406*.

[7] L. Buşoniu, A. Daniels, and R. Babuška, "Online learning for optimistic planning," *Eng. Appl. AI*, vol. 55, pp. 60–72, Oct. 2016.

[8] L. Buşoniu, E. Páll, and R. Munos, "An analysis of optimistic, best-first search for minimax sequential decision making," in *Proc. IEEE Int. Symp. Adapt. Dyn. Program. Reinforcement Learn. (ADPRL)*, Dec. 2014, pp. 10–12.

[9] L. Buşoniu, E. Páll, and R. Munos, "Continuous-action planning for discounted infinite-horizon nonlinear optimal control with Lipschitz values," *Automatica*, vol. 92, pp. 100–108, Jun. 2018.

[10] L. Busoniu, J. B. Rejeb, I. Lal, I.-C. Morarescu, and J. Daafouz, "Optimistic minimax search for noncooperative switched control with or without dwell time," *Automatica*, vol. 112, Feb. 2020, Art5. no. 108632.

[11] D. Goktas and A. Greenwald, "Gradient descent ascent in min-max Stackelberg games," 2022, *arXiv:2208.09690*.

[12] D. Goktas, S. Zhao, and A. Greenwald, "Zero-sum stochastic Stackelberg games," in *Proc. 36th Conf. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 11658–11672.

[13] A. Haurie, J. B. Krawczyk, and G. Zaccour, *Games and Dynamic Games*, vol. 1. Singapore: World Sci. Publ., 2012.

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

[15] R. E. Korf and D. M. Chickering, "Best-first minimax search," *Artif. Intell.*, vol. 84, nos. 1–2, pp. 299–337, 1996.

[16] J. Marecki, G. Tesauro, and R. Segal, "Playing repeated Stackelberg games with unknown opponents," in *Proc. 11th Int. Conf. Autonomous Agents Multiagent Syst.*, Valencia, Spain, Jun. 2012, pp. 821–828.

[17] R. Munos, "From bandits to Monte Carlo tree search: The optimistic principle applied to optimization and planning," *Found. Trends Mach. Learn.*, vol. 7, no. 1, pp. 1–130, 2014.

[18] C. E. Shannon, "XXII. Programming a computer for playing chess," *London, Edinburgh, Dublin Philosophical Mag. J. Sci.*, vol. 41, no. 314, pp. 256–275, 1950.

[19] S. Simons, "Minimax theorems and their proofs," in *Minimax and Applications*. Boston, MA, USA: Springer, 1995, pp. 1–23.

[20] A. Sinha, F. Fang, B. An, C. Kiekintveld, and M. Tambe, "Stackelberg security games: Looking beyond a decade of success," in *Proc. 27th Int. Joint Conf. Artif. Intell. (IJCAI)*, Stockholm, Sweden, Jul. 2018, pp. 13–19.

[21] M. Sion, "On general minimax theorems," *Pac. J. Math.*, vol. 8, no. 1, pp. 171–176, 1958.

[22] D. Sotiropoulos, "Solving discrete minimax problems using interval arithmetic," in *Proc. Int. Conf. Optim., Techn. Appl. (ICOTA)*, 2004, pp. 1–16.

[23] V. S. Varma, I.-C. Morărescu, S. Lasaulce, and S. Martin, "Marketing resource allocation in duopolies over social networks," *IEEE Control Syst. Lett.*, vol. 2, no. 4, pp. 593–598, Oct. 2018.

[24] J. Wang, H. Hu, D. P. Nguyen, and J. F. Fisac, "MAGICS: Adversarial RL with minimax actors guided by implicit critic Stackelberg for convergent neural synthesis of robot safety," 2024, *arXiv:2409.13867*.

[25] Y. Wang and S. Gelly, "Modifications of UCT and sequence-like simulations for Monte-Carlo go," in *Proc. IEEE Symp. Comput. Intell. Games (CIG)*, Honolulu, Hawaii, Apr. 2007, pp. 175–182.

[26] Y. Wang, G. Zhang, and J. Ba, "On solving minimax optimization locally: A follow-the-ridge approach," 2019, *arXiv:1910.07512*.

[27] K. Zhang, S. M. Kakade, T. Basar, and L. F. Yang, "Model-based multi-agent RL in zero-sum Markov games with near-optimal sample complexity, *J. Mach. Learn. Res.*, vol. 24, no.175, pp. 1–53, 2023.

[28] M. Zhu and S. Martinez, "Stackelberg-game analysis of correlated attacks in cyber-physical systems," in *Proc. IEEE Am. Control Conf.*, San Francisco, CA, USA, Jul. 2011, pp. 4063–4068.

Authorized licensed use limited to: Universitatea Tehnica din Cluj-Napoca. Downloaded on October 09,2025 at 06:16:22 UTC from IEEE Xplore. Restrictions apply.