Heuristic Search Value Iteration for zero-sum Stochastic Games

Olivier Buffet, Jilles Dibangoye, Abdallah Saffidine, Vincent Thomas

Heuristic Search Value Iteration for zero-sum Stochastic Games

Olivier Buffet, Jilles Dibangoye, Abdallah Saffidine, Vincent Thomas

***Abstract*—In sequential decision-making, heuristic search algorithms allow exploiting both the initial situation and an admissible heuristic to efficiently search for an optimal solution, often for planning purposes. Such algorithms exist for problems with uncertain dynamics, partial observability, multiple criteria, or multiple collaborating agents. Here we look at two-player zero-sum stochastic games with discounted criterion, in a view to propose a solution tailored to the fully observable case, while solutions have been proposed for particular, though still more general, partially observable cases. This setting induces reasoning on both a lower and an upper bound of the value function, which leads us to proposing zsSG-HSVI, an algorithm based on Heuristic Search Value Iteration (HSVI), and which thus relies on generating trajectories. We demonstrate that, each player acting optimistically, and employing simple heuristic initializations, HSVI's convergence in finite time to an ϵ-optimal solution is preserved. An empirical study of the resulting approach is conducted on benchmark problems of various sizes.**

I. Introduction

Heuristic search techniques have been introduced to solve single-agent single-criterion optimization problems in deterministic settings, often for planning purposes, with algorithms like A*, (L)RTA*, or RBFS [1, 2]. They have then been extended to solving problems with stochastic dynamics in the MDP or POMDP framework as in (L)RTDP, (L)AO*, or HSVI [3, 4, 5], and even in multi-agent collaborative settings in MAA* or FB-HSVI [6, 7]. In all these cases, a single criterion is considered, so that guiding the exploration by being optimistic remains a rather straightforward thing to do, even when planning for multiple agents. When multiple criteria are considered (for a single agent or collaborative agents), this guidance can be adapted both when aggregating the criteria or when searching for a Pareto front [8]. Our focus here is on non-collaborative multi-agent problems. We look in particular at discounted two-player zero-sum stochastic games (zs-SGs) with perfect information and finite state and action sets. Such games can serve for instance in robust control [9], or in solving or analyzing Real-Time Strategy games (*e.g.*, μRTS [10]). While particular partially observable SGs, hence more general problems, have been solved through heuristic search [11, 12] (see next section), this work fills a gap by proposing and studying a simpler solution that is better suited to the fully observable case.

Here, discounted zs-SGs are equipped with (i) an initial state s_0, and (ii) initial upper and lower bounds (U and L) of the optimal value function, *i.e.*, admissible heuristics. Under these assumptions, we look at learning/trajectory-generating algorithmic schemes (*à la* LRTA*, (L)RTDP or HSVI), and show that trajectories can be generated by letting the two players act optimistically—one greedily w.r.t. to an upper-bound U of the optimal (Nash equilibrium) value, the other w.r.t. to a lower-bound L—while ensuring that these two bounds converge to the optimum. Using two bounds naturally leads to algorithms resembling Bounded-RTDP and HSVI [13, 5]. We take inspiration from HSVI, to derive our main contribution, *zsSG-HSVI*, a heuristic-search algorithm that provably converges in finite time to an error-bounded solution.

The paper is organized as follows. It first discusses related work on stochastic games in Section II and provides background on normal-form games, Markov decision processes (MDPs), and stochastic games in Section III. Shapley's reference algorithm, whose stopping criterion relies on the Bellman residual (BR), is called here *Shapley-BR*. It is presented alongside a variant we propose, *Shapley-Gap*, which updates both a lower and an upper bound of the optimal value function, stopping when the gap between them is small enough. This new algorithm combines properties of Shapley-BR and our contributed zsSG-HSVI. Then, Section IV presents preliminary results on upper- and lower-bounded normal-form games before proposing the key ingredients to deriving *zsSG-HSVI* and proving its convergence. Finally, Section V presents an empirical study before concluding in Section VI. Complementary discussions and benchmark problem descriptions are provided as supplementary material (referred here as appendices).

II. Related Work

Discounted zero-sum stochastic games (zs-SGs) have been addressed using approaches such as exact dynamic programming (DP) [14] (akin to Value or Policy Iteration for MDPs),

Manuscript received December 9, 2019; revised May 10, 2020; accepted June 19, 2020. This work was supported by the French National Research Agency through the "Planning and Learning to Act in Systems of Multiple Agents" Project under Grant 19-CE23-0018-01. *(Corresponding author: Olivier Buffet.)*

Olivier Buffet and Vincent Thomas are with Université de Lorraine, INRIA, CNRS, LORIA, F-54000 Nancy, France (e-mail: {olivier.buffet,vincent.thomas}@loria.fr).

Jilles Dibangoye is with Univ Lyon, INSA Lyon, INRIA, CITI, F-69621 Villeurbanne, France (e-mail: jilles-steeve.dibangoye@insa-lyon.fr).

Abdallah Saffidine is with the University of New South Wales, Sydney, Australia (e-mail: abdallahs@cse.unsw.edu.au).

Digital Object Identifier 10.1109/TG.2020.3005214

reinforcement learning (RL) [15], policy gradient [16], and approximate DP [17, 18].

The total reward criterion common in board games, *i.e.*, non-discounted zs-SGs, requires all trajectories to end in a terminal state, and has been addressed with (i) RL [15], (ii) backward induction [19, 20], *i.e.*, applying DP on reachable states while pruning irrelevant branches based on conservative estimates, (iii) sub-optimal heuristic search algorithms [21, 22], when real-time constraints have to be enforced, and (iv) Simultaneous Move MCTS [20].[1] Here, exact DP algorithms come with the strongest convergence guarantees. Among them, backward induction is a form of optimal heuristic search. The present work differs in that it addresses problems with cycles, non-zero rewards not restricted to terminal states (which may not exist), and a discounted criterion.

Recent works addressed particular cases of discounted (two-player) zero-sum partially observable stochastic games (zs-POSGs) using heuristic search: Horák et al. [11] considered One-Sided POSGs, *i.e.*, the case where one player has access to the system state, plus the action and observation of the other player, and Horák and Bošanský [12] considered POSGs with public observations, *i.e.*, the case where each Player i knows his own private state s_i and both players receive the same public observations of each player's private state, so that they have common knowledge of Player $-i$'s belief over Player i's private state. Moving from MDPs and POMDPs (as in Smith's work 2007) to these settings requires changes to the algorithm that make a different approach necessary to theoretically analyze the finite-time convergence. In contrast, the present paper looks at the simpler case of (fully observable) stochastic games, which allows for an algorithm closer to the original HSVI, as well as similar finite-time convergence results, thus suggesting different convergence behaviors compared to the aforementioned POSG versions.

To sum up, the present work thus fills a gap, both from a theoretical and empirical viewpoint, in the study of heuristic search for zero-sum games in between (fully observable) sequential games, non-discounted zs-SGs, and some discounted zs-POSGs. We now turn to presenting necessary background on game theory and heuristic search.

III. Background

In the following, upperscripts indicate players, and subscripts indicate time steps or iterations.

A. Normal-form/Matrix Games

A two-player game in *normal-form* is defined by a tuple $\Gamma \stackrel{\text{def}}{=} \langle \mathcal{A}^1, \mathcal{A}^2, v^1, v^2 \rangle$, where $\mathcal{A}^i$ is player i's ($i \in \{1, 2\}$) finite set of pure strategies, and $v^i : \mathcal{A}^1 \times \mathcal{A}^2 \mapsto \mathbb{R}$ is player i's payoff function. The objective is then, for each player, to maximize his expected payoff. Unless they have identical payoff functions, this is not about finding an optimum of some function, but an *equilibrium*. A classical solution concept is that of *Nash Equilibrium* (NE) [25], defined as a pair $(d^1, d^2) \in \Delta(\mathcal{A}^1) \times \Delta(\mathcal{A}^2)$ of mixed strategies, *i.e.*, probability distributions over pure strategies, such that no player has any incentive to deviate on his own:

$$\forall d'^1 \in \Delta(\mathcal{A}^1), \quad v^1(d'^1, d^2) \leq v^1(d^1, d^2),$$
$$\forall d'^2 \in \Delta(\mathcal{A}^2), \quad v^2(d^1, d'^2) \leq v^2(d^1, d^2).$$

In a zero-sum game, $v^1 + v^2 = 0$—*i.e.*, one player's gain is the other player's loss—, and we will typically note $v = v^1 = -v^2$. Player 1 then maximizes his expected payoff, while Player 2 minimizes it. Also, as demonstrated by von Neumann [26], such a game Γ has a single *Nash equilibrium value* $\text{NEV}(\Gamma)$ equal to both the minimax and maximin values. As a consequence, strategies forming a (joint) *Nash equilibrium strategy* (NES) can be found by solving (often as Linear Programs):

$$d^{\text{NES},1} = \arg\max_{d^1 \in \Delta(\mathcal{A}^1)} \min_{a^2 \in \mathcal{A}^2} v(d^1, a^2),$$
$$d^{\text{NES},2} = \arg\min_{d^2 \in \Delta(\mathcal{A}^2)} \max_{a^1 \in \mathcal{A}^1} v(a^1, d^2).$$

B. Stochastic Games

A discounted 2-player zero-sum stochastic (or Markov) game (zs-SG) [27, 28, 14] is specified by a tuple $\langle \mathcal{S}, \mathcal{A}^1, \mathcal{A}^2, P, r, \gamma, s_0 \rangle$, where $\mathcal{S}$ is a finite set of states, $\mathcal{A}^1$ and $\mathcal{A}^2$ are finite sets of actions (one per player), $P_{a^1,a^2}(s'|s)$ is the probability to transition from state s to s' when actions a^1 and a^2 are performed; $r(s, a^1, a^2)$ is a (scalar) reward function; $\gamma \in [0, 1)$ is a discount factor; and s_0 is the initial state. Player 1's objective is to maximize the expected γ-discounted sum of rewards $E[\sum_t \gamma^t R_t | s_0]$ while Player 2's objective is to minimize that same quantity. For convenience, we will denote: $\pi^i : \mathcal{S} \to \Delta(\mathcal{A}^i)$ a stochastic strategy for player i; and $\pi = (\pi^1, \pi^2)$ a joint strategy (or pair of strategies) for both players.

Such a game can be re-written in normal form and solved by searching for a NE (as a mixed strategy). Yet, a more satisfying solution concept here is that of *Markov perfect equilibrium* (MPE), *i.e.*, noting that a local normal-form game is faced in each state, solutions which consist in one NE per reachable state, with Markov strategies, *i.e.*, decisions depending on the current state alone, not on past information.[2] Let us thus define (i), for any $s \in \mathcal{S}$ and $V \in \mathbb{R}^{|\mathcal{S}|}$, the *Shapley matrix game* $\Gamma^s(V) \stackrel{\text{def}}{=} [r(s, \circ, \circ) + \gamma \sum_{s'} P_{\circ,\circ}(s'|s) V(s')]$ (where $(\circ, \circ)$ denotes the various action pairs (a^1, a^2) in the matrix), *i.e.*, the normal-form game faced when in state s and assuming that the expected discounted return obtained from subsequent states is specified by *value function* V, and (ii) *Shapley's optimality operator* $\mathcal{H} : V \mapsto \text{NEV}(\Gamma^\circ(V))$, which computes the NEV of each state's Shapley matrix game given V. $\mathcal{H}$ is a contraction mapping, and its unique fixed point is the value function common to all MPE strategies, denoted V^*. This implies that, in any state s, all MPE strategies induce the same expected discounted returned. *Shapley's algorithm* for solving zs-SGs Shapley [27], analogous to *value iteration (VI)* for MDPs, consists in iteratively applying this operator to generate a sequence of value functions V_k (randomly initialized) until

[1] For other criteria, refer to [23].

[2] As Shapley [27], we do not account for the pay-off relevance of states.

ϵ-convergence (details below), as depicted in Appendix A (Algorithm 3). Then, when in some state s at execution time, both players have to act according to a NES of the induced Shapley game matrix in s. Note: other exact algorithms also handle problems with no initial states [14], while the present contribution exploits one.

Stopping Criteria for Shapley's Algorithm: We first generalize Shapley's algorithm for 2-player zero-sum stochastic games by distinguishing whether moves are simultaneous or sequential, player 1 acting first or second. This simply means trading the (simultaneous) minmax operator for Nash equilibria in simultaneous normal-form games for a *min-then-max* or *max-then-min* operator. In all three cases the same algorithm converges due to the contraction property induced by the discount factor.

Whatever the operator at hand, the algorithm usually (i) updates the values asynchronously (*à la Gauss-Seidel*) and (ii) is stopped when the Bellman residual (BR) is below some threshold θ, *i.e.* $\|V_k - V_{k-1}\|_\infty \leq \theta$. Then, as for VI in the MDP setting, to ensure that the value of the resulting pair of strategies π_k is within $\epsilon > 0$ of the optimum, this algorithm (denoted *ShapleyBR*) should use $\theta = \frac{1}{2}\frac{1-\gamma}{\gamma}\epsilon$.

All three update operators are monotone, *i.e.*, (i) if $V_k \geq V^*$ $(\forall s)$, then $V_{k+1} \geq V^*$, and (ii) if $V_k \leq V^*$ $(\forall s)$, then $V_{k+1} \leq V^*$. This allows us to propose a *bounded* algorithm, *ShapleyGap*, that maintains an upper- and a lower-bound (U and L) of V^*, performing updates only at states s where the gap/width $U(s) - L(s)$ is larger than the desired threshold $\epsilon > 0$, and stopping the algorithm when $\|U - L\|_\infty < \epsilon$. The max (resp. min) player should then act "optimally" with respect to L (resp. U). The whole process behind ShapleyGap is detailed in Appendix A (Algorithm 4), and its finite-time convergence is stated further on in Corollary 2.

C. MDPs and HSVI

Markov decision processes (MDPs) [29, 30] can be seen as stochastic games in which one player—without loss of generality, (minimizing) Player 2—has a single available action, thus no decision to make. The problem is then to find a strategy, also referred to as a *policy*, for (maximizing) Player 1. Here, the NEV becomes the *optimal value function* V^*, and acting greedily with respect to the value function induces an optimal policy, so that there always exists a deterministic optimal policy.

Bounded RTDP [13] and HSVI [5] (detailed in Algorithm 1) are two algorithms that solve MDPs relying on (i) an initial state s_0 to focus on most relevant parts of the state space, (ii) upper and lower bounds (U and L) of the optimal value function V^*, and (iii) performing pointwise updates of these bounds at the states encountered while following trajectories. While HSVI was initially introduced for solving POMDPs, we focus on the generic (MDP) version presented by Smith [24] rather than BRTDP, because it comes with stronger theoretical guarantees (*e.g.*, it converges in finite time rather than in the limit). Both HSVI and BRTDP select actions greedily w.r.t. the upper bound, and stop sampling trajectories when $U(s_0) - L(s_0)$ is below a threshold $\epsilon > 0$. To select the next state given s and a, HSVI picks a state s' maximizing $excess(s', \delta) = P_a(s'|s)(U(s') - L(s') - \gamma^{-\delta}\epsilon)$, where δ is the depth of s' in the current trajectory, to focus on states whose updates are more likely to help.[3] As we shall see later, the term $\gamma^{-\delta}\epsilon$ ensures the convergence through a recursive process.

Algorithm 1: Heuristic Search Value Iteration
(in red: lines that will differ in the contributed algorithm)

```
Fct HSVI (ε)
    Initialize L and U
    while (U(s_0) − L(s_0)) > ε do
        RecursivelyTry (s_0, δ = 0)
    return L
Fct RecursivelyTry (s, δ)
    if (U(s) − L(s)) > γ^{−δ} ε then
        Update (s)
        a* ∈ arg max_{a∈A} r(s,a) + γ Σ_{s'} P_a(s'|s) U(s')
        s' ∈ arg max_{s"∈S} excess(s", δ)
        RecursivelyTry (s', δ + 1)
        Update (s)
    return
Fct Update (s)
    L ← Update (L, s)
    U ← Update (U, s)
```

IV. Algorithm and Convergence Proof

A. Bounding a Normal-form Game

As explained in the introduction, we would like to propose heuristic search algorithms (*à la* HSVI) for solving 2-player zs-SGs, *i.e.*, focusing the computational efforts on relevant parts of the state space by exploiting (i) the knowledge of the initial state s_0, and (ii) admissible heuristics for both players, used to initialize upper and lower bounds of the value function. While classical (1-player vs. nature) heuristic search requires only an upper bound of the value function to optimistically guide the search, 2-player zs-SGs shall require both an upper and a lower bound (so as (i) to be optimistic for both players, and (ii) to derive each player's executed strategy). We will thus end up computing Nash Equilibria (NEs) for both upper- and lower-bounding Shapley matrix games $\Gamma^s(U)$ and $\Gamma^s(L)$ in each encountered state s. Then, what does happen if, in some state s, both NEs are equal? As demonstrated by the next two results, in this case the obtained value is the NE value for that state.

Lemma 1. *If zs normal-form game Γ^{up}'s payoff v^{up} upper-bounds same-dimension zs normal-form game Γ^{lo}'s payoff v^{lo}, then* $\text{NEV}(\Gamma^{up}) \geq \text{NEV}(\Gamma^{lo})$.

Proof.

$$\forall a^1, a^2 \quad v^{up}(a^1, a^2) \geq v^{lo}(a^1, a^2)$$

[3] BRTDP samples a state according to a distribution which resembles a normalized version of this excess function.

$$\forall a^1 \quad \min_{a^2} v^{up}(a^1,a^2) \geq \min_{a^2} v^{lo}(a^1,a^2)$$
$$\max_{d^1}\min_{a^2}\sum_{a^1} d^1(a^1)v^{up}(a^1,a^2) \geq \max_{d^1}\min_{a^2}\sum_{a^1} d^1(a^1)v^{lo}(a^1,a^2)$$
$$\textit{i.e.,} \qquad \textsc{nev}(\Gamma^{up}) \geq \textsc{nev}(\Gamma^{lo}).$$

□

Corollary 1. *In a zs-SG, if, in some state s, the NE values of $\Gamma^s(L)$ and $\Gamma^s(U)$ are equal, then $\textsc{nev}(\Gamma^s(L)) = \textsc{nev}(\Gamma^s(U))$ is the NE value, $V^*(s)$, of the* converged *zs Shapley game in that state (for the MPE of the zs-SG).*

Proof. Applying the previous lemma in some state s, we always have in our setting: $\textsc{nev}(\Gamma^s(L)) \leq V^*(s) \leq \textsc{nev}(\Gamma^s(U))$. So, when $\textsc{nev}(\Gamma^s(L)) = \textsc{nev}(\Gamma^s(U))$, this is also the value of $V^*(s)$. □

We now turn to showing how to adapt HSVI to the zs-SG setting. While $U(s_0) - L(s_0) > \epsilon$, HSVI generates trajectories by applying, in each encountered state, (i) an action selection strategy and (ii) a next-state selection strategy. We now present these strategies and other details of the proposed algorithm, and prove that the latter finds strategies which are ϵ-optimal in s_0 in finite time.

B. Joint Action Selection

In an MDP, greedy action selection w.r.t. an optimistic bound guarantees convergence. In a zs-SG, assuming that player 2 has only one action, player 1 faces a reward-based MDP and its optimistic bound is the upper one (U). Conversely, if player 1 has only one action, player 2 faces a cost-based MDP and its optimistic bound is the lower one (L).

Hence, noting $\textsc{nes}^i(\Gamma)$ a NE strategy for player i in game Γ, the proposed exploration strategy, when in state s:

- player 1 picks stochastic action $d^{U,1}$ greedily w.r.t. to the upper bound U, *i.e.*, playing $\textsc{nes}^1(\Gamma^s(U))$; and
- player 2 picks stochastic action $d^{L,2}$ greedily w.r.t. to the lower bound L, *i.e.*, playing $\textsc{nes}^2(\Gamma^s(L))$.

We call the resulting joint decision rule $d = (d^{U,1}, d^{L,2})$ a *tentative NES* at s.

C. Next State Selection

After picking a joint strategy, the next state s' can be selected in various ways—*e.g.*, sampling a^1 from $d^1(\cdot)$, a^2 from $d^2(\cdot)$, then s' from $P_{a^1,a^2}(\cdot|s)$ (*à la* (L)RTDP). Yet, to preserve HSVI's convergence guarantees, we prefer selecting a next state s' maximizing the *expected excess* $\text{EE}(s,d^1,d^2,s',\delta) \stackrel{\text{def}}{=}$

$$\sum_{a^1,a^2} d^1(a^1)d^2(a^2)P_{a^1,a^2}(s'|s)\underbrace{\left[width(\hat{V}(s')) - \gamma^{-\delta}\epsilon\right]}_{excess(s',\delta)},$$

where $\hat{V}$ denotes the pair (L,U), $width((x,y)) \stackrel{\text{def}}{=} y - x$, and δ is the current depth. Note: In the following, operators on value functions such as $\mathcal{H}$ are naturally extended to pairs of function, *e.g.*, $\mathcal{H}(\hat{V}) = (\mathcal{H}(L), \mathcal{H}(U))$.

D. Value Update

As for classical HSVI, we use *point-based* operators to update the lower and upper bounds, *i.e.*, operators that are applied at a given point s, and define:

- a *uniformly improvable* (UI) lower bound L as respecting $\mathcal{H}L \geq L$; and
- a *strong* point-based update operator for the lower bound $\mathcal{K}^L_\circ$ as respecting, for each s where it is applied and any L, (i) $(\mathcal{K}^L_s L)(s) = (\mathcal{H}L)(s)$ and (ii) $(\mathcal{K}^L_s L)(s') \leq (\mathcal{H}L)(s')$ in any other point s'

(the same definitions apply to the upper bound U by reverting inequality operators). Uniform improvability, along with monotonicity (Sec. III-B), is necessary to ensure that lower (resp. upper) bounds remain lower (resp. upper) bounds. The strong property is required to ensure convergence to the optimal value function in the limit.

In our setting, we use a single update operator $\mathcal{K}_s$ for both bounds, which, for any state s, only updates the value for this state with the natural value updates:

$$(\mathcal{K}_s L)(s) \stackrel{\text{def}}{=} \textsc{nev}(\Gamma^s(L)),$$
$$(\mathcal{K}_s U)(s) \stackrel{\text{def}}{=} \textsc{nev}(\Gamma^s(U)),$$

and leaves L and U unchanged for other states.

As in the MDP setting, we have (for both L and U) that: (i) our proposed operator is strong, thus conservative [24, Def. 3.24, 3.25]; and (ii) any conservative update operator preserves UI [24, Th. 3.29]. We thus mainly need to ensure that our initializations induce UI bounds (see next sub-section).

E. Upper- and Lower-Bound Initializations

As for POMDPs, one way to initialize the lower and upper bounds is to compute the fixed point of an operator that lower- or upper-bounds Shapley's optimality operator $\mathcal{H}$. Table I shows the candidate operators for both bounds:

$\mathcal{G}^{[L/U]}_{\textbf{SEQ}}$**:** a sequential game approximation—also known as *serialization*—, *i.e.*, setting U as the optimal value if player 1 acts after player 2 (the latter knowing the action choice of the former), and L as the opposite;

$\mathcal{G}^{[L/U]}_{\textbf{TRIV}}$**:** trivial approximations that translate to, $\forall s$,

$$U(s) = \frac{1}{1-\gamma}R_{\max} \stackrel{\text{def}}{=} \frac{1}{1-\gamma}\max_{s',a^1,a^2} r(s',a^1,a^2) \text{ and}$$
$$L(s) = \frac{1}{1-\gamma}R_{\min} \stackrel{\text{def}}{=} \frac{1}{1-\gamma}\min_{s',a^1,a^2} r(s',a^1,a^2).$$

The former approximations require solving sequential games, possibly with Shapley- or HSVI-like algorithms.[4] The latter two values can be computed in constant time.

Uniform Improvability: Without loss of generality, let us discuss Uniform Improvability only for upper bounds. As proposed above, the initial upper bound U is obtained as the unique fixed point of some update operator $\mathcal{K}^{up}$ which always "upper-bounds" Shapley's optimality operator $\mathcal{H}$. As a consequence, $\mathcal{H}U \leq \mathcal{K}^{up}U = U$, so that U is uniformly improvable. This approach is similar to Theorem 3.20 in [24, page 71], and also applies to lower bounds.

[4]Churchill et al. [21] do so with an $\alpha\beta$-based algorithm, thus not dealing with cycles and infinite horizons.

TABLE I: Various update operators for the value function, ordered by resulting value. Top: 2 operators for initializing U. Middle: optimal Shapley operator. Bottom: 2 operators for initializing L.

$(\mathcal{G}^U_{\text{TRIV}}V)(s)$	$= \max_{s',a^1,a^2,s''} r(s',a^1,a^2) + \gamma V(s'')$	
$(\mathcal{G}^U_{\text{SEQ}}V)(s)$	$= \min_{a^2}\max_{a^1}$	$\Gamma^s(V)(a^1,a^2)$
$(\mathcal{H}V)(s)$	$= \min_{d^2}\max_{a^1}$	$\Gamma^s(V)(a^1,d^2)$
$= \text{NEV}(\Gamma^s(V))$	$= \max_{d^1}\min_{a^2}$	$\Gamma^s(V)(d^1,a^2)$
$(\mathcal{G}^L_{\text{SEQ}}V)(s)$	$= \max_{a^1}\min_{a^2}$	$\Gamma^s(V)(a^1,a^2)$
$(\mathcal{G}^L_{\text{TRIV}}V)(s)$	$= \min_{s',a^1,a^2,s''} r(s',a^1,a^2) + \gamma V(s'')$	

On-demand Initializations: The following discussion on how to handle initializations is set in the context of zs-SGs, but applies to other settings as well.

Value function initializations are often considered as part of pre-computations performed *a priori* for all states, either because all states equally need initializations (as here in ShapleyBR and ShapleyGap, but contrary to HSVI), or because the initialization process provides values for all states anyway (as here with the trivial initializations). Initializations based on serialized versions of (simultaneous move) zs-SGs can be obtained either through a serialized version of ShapleyBR or ShapleyGap (*i.e.*, with "serialized" operators), or through a serialized version of HSVI. For (simultaneous) HSVI, our algorithm will thus try to benefit from *on-demand* initializations, calling (serialized) HSVI whenever needed. Note that *a priori* initializations shall typically be used with "uniform" stopping criteria, *i.e.*, requesting the same property to be satisfied on all states, while *on demand* initializations shall satisfy a property at least at the state at hand, and possibly depending on the current depth for algorithms such as HSVI (because less precision is needed when going deeper in a trajectory). Some implementation details are provided in Section V. Note that, for the sake of clarity, Algorithm 2 does not reflect the process of on-demand initializations.

F. Trials Termination Criterion and Convergence Proof

Let us end a trial when reaching a state s at depth δ that is *NEV-finished—i.e.*, when the excess, $excess(s,\delta)$, is non-positive. Using this trial termination condition is the last element needed to ensure that the zs-SG version of HSVI converges in finite time. We essentially need to prove the following key lemma before using results from [24].

Lemma 2 (*Cf.* [24, Lemma 6.1])**.** *Let $\mathcal{K}_\circ$ be a strong update operator, let L and U be uniformly improvable value functions, let s be a state, and let $d^* = (d^{U,1}, d^{L,2})$ be the current tentative NES at s. Then*

$$width(\mathcal{K}_s\hat{V}(s)) \leq \gamma \sum_{s'} P_{d^{U,1},d^{L,2}}(s'|s)\, width(\hat{V}(s')),$$

where $P_{\circ,\circ}$ is naturally extended to mixed strategies (stochastic actions).

Proof. We have

$$\begin{aligned}
&width(\mathcal{K}_s\hat{V}(s)) \\
&= width(\mathcal{H}\hat{V}(s)) && (\mathcal{K}_s \text{ is strong}) \\
&= \mathcal{H}U(s) - \mathcal{H}L(s) && (\text{def. of } width(\cdot)) \\
&= \text{NEV}(\Gamma^s(U)) - \text{NEV}(\Gamma^s(L)) && (\text{def. of } \mathcal{H}) \\
&= \max_{d^1}\min_{a^2} v^U(s,d^1,a^2) - \min_{d^2}\max_{a^1} v^L(s,a^1,d^2) \\
&= \max_{d^1}\min_{d^2} v^U(s,d^1,d^2) - \min_{d^2}\max_{d^1} v^L(s,d^1,d^2) \\
&= \min_{d^2} v^U(s,d^{U,1},d^2) - \max_{d^1} v^L(s,d^1,d^{L,2}) \\
&\leq v^U(s,d^{U,1},d^{L,2}) - v^L(s,d^{U,1},d^{L,2}) \\
&= \gamma \sum_{s'} P_{d^{U,1},d^{L,2}}(s'|s)(U(s') - L(s')) \\
&= \gamma \sum_{s'} P_{d^{U,1},d^{L,2}}(s'|s)\, width(\hat{V}(s')).
\end{aligned}$$

□

This result is important as it upper-bounds the width at s after update by a linear combination of the widths of the states immediately reachable from s through d^*. Thus, sufficiently reducing the widths of those next states (which may change during execution) induces reducing the width at s. This serves to show both HSVI's and ShapleyGap's finite-time convergence.

Corollary 2. *ShapleyGap converges in finite time to ϵ-optimal lower and upper bounds.*

Proof. Let U_i and L_i denote the upper- and lower-bounding value functions after i iterations of ShapleyGap. Let us demonstrate by induction that $\|U_i - L_i\|_\infty \leq \gamma^i \|U_0 - L_0\|_\infty$, until the termination criterion is reached. This property trivially holds for $i = 0$. Let us assume that it holds for some $i \geq 0$. If $\|U_i - L_i\|_\infty \leq \epsilon$, then the termination criterion is reached. Otherwise, Lemma 2 can be applied at any state s such that $U_i(s) - L_i(s) > \epsilon$, so that

$$U_{i+1}(s) - L_{i+1}(s) \leq \gamma \sum_{s'} P_{d^{U,1},d^{L,2}}(s'|s)\, width(\hat{V}_i(s'))$$

(where $d^{U,1}$ and $d^{L,2}$ have been computed by the algorithm)

$$\begin{aligned}
&\leq \gamma \sum_{s'} P_{d^{U,1},d^{L,2}}(s'|s)\gamma^i \|U_0 - L_0\|_\infty \\
&\leq \gamma^{i+1} \|U_0 - L_0\|_\infty.
\end{aligned}$$

For any other state, the gap is already less than (or equal to) ϵ. The property thus holds for $i+1$. As a consequence, ShapleyGap converges in $\lceil \log_\gamma(\epsilon/\|U_0 - L_0\|_\infty) \rceil$ iterations. □

Theorem 1. *The HSVI variant for zs-SGs specified above—through selection processes, update operators and trial stopping criterion—converges in finite time.*

Proof. (sketch inspired by [24]) First, the previous lemma also holds when replacing the *width* by the *excess* [24, Lemma 6.2]. As a consequence, if all successors s' of s reachable through d^* are NEV-finished, then s' is NEV-finished as well [24, Lem. 6.3]. Also, HSVI always selects an NEV-unfinished successor if one exists [24, Lem. 6.4], and all states beyond depth $\delta_{\max} \stackrel{\text{def}}{=} \lceil \log_\gamma(\epsilon/\|U - L\|_\infty) \rceil$ are NEV-finished [24, Lem. 6.4] (*e.g.*, with U_0 and L_0 the initial bounds). A consequence is that any trial terminates

and the penultimate visited state becomes NEV-finished [24, Lem. 6.7]. From this, the tree of reachable states from s_0 has bounded depth $\delta_{\max}$ and finite branching factor β, so that s_0 will be NEV-finished (at depth 0) after at most $\frac{\beta^{\delta_{\max}}-1}{\beta-1}$ trials. □

The complete algorithm is detailed in Algorithm 2. As can be observed, the general algorithmic schema of HSVI is untouched as it essentially differs in (i) the action selection (relying on NESs) (lines 9–10), (ii) the next-state selection (lines 11), and (iii) the bound updates (relying on NEVs) (lines 16–17).

Algorithm 2: zsSG-HSVI (in red: differences with HSVI)

```
1  Fct HSVI (ε)
2      Initialize L and U
3      while width(s_0) > ε do
4          RecursivelyTry (s_0, δ = 0)
5      return L
6  Fct RecursivelyTry (s, δ)
7      if width(s) > γ^{-δ} ε then
8          Update (s)
9          d^{U,1} ← NES^1(Γ^s(U))
10         d^{L,2} ← NES^2(Γ^s(L))
11         s' ∈ arg max_{s"∈S} EE(s, d^{U,1}, d^{L,2}, s", δ)
12         RecursivelyTry (s', δ + 1)
13         Update (s)
14     return
15 Fct Update (s)
16     L ← Update (L, s) /* uses NEV(Γ^s(L))  */
17     U ← Update (U, s) /* uses NEV(Γ^s(U))  */
```

Comparison with HSVI for POMDPs and POSGs: Note that the finite branching factor, due to the finite number of reachable states and despite the infinite set of actions, allows deriving a version of HSVI with the same definition of the excess function and the same convergence result (including the bound) as HSVI for POMDPs [24]. In the contrary, the potentially infinite branching factor in POSGs considered by [11] and [12] requires adding a term to the excess function, which induces an increased $\delta_{\max}$, and relying on the Lipschitz-continuity of the optimal value function to prove finite-time convergence to an ϵ-optimal solution.[5]

G. Comparing HSVI, ShapleyGap, and ShapleyBR

ShapleyGap can be seen as in-between HSVI and ShapleyBR since this variant of the Shapley (BR) algorithm shares the use of upper and lower bounds with HSVI. We here detail these similarities that imply similarities in the convergence behaviors.

[5]One could possibly derive a bound on the number of iterations before convergence for these POSGs, but this would involve results related to hypersphere packing, as well as the volume of both an hypersphere and a regular simplex.

a) Similarities between ShapleyBR and ShapleyGap: First, ShapleyBR and ShapleyGap both ignore the initial state s_0, contrary to HSVI, and stop when a uniform criterion (satisfied by all states) is satisfied, while HSVI looks only at the gap at s_0. Also, they will both adopt the same ordering of state updates (a rarely scrutinized factor), which may benefit (or, in the contrary, disadvantage) value propagation through states, and thus speed up (resp. slow down) the convergence process. HSVI, on its side, updates states along visited trajectories, so that not all states are visited, and not all visited states attain a (local) convergence criterion. Finally, ShapleyGap and ShapleyBR both use *a priori* initializations, while HSVI uses *on demand* initializations.

b) Similarities between ShapleyGap and HSVI: Compared to ShapleyBR, ShapleyGap and HSVI have not one, but two value functions to update, so that each state update is twice as expensive. Also, ShapleyBR will keep on updating all states at each iteration until it has converged, while ShapleyGap will stop updating states at which the gap is smaller than the required threshold, and HSVI will update only visited states if their gap is beyond a depth-dependent threshold. Finally, ShapleyGap and HSVI rely on the same value function initializations, so that they are likely to be both equally "lucky" (or "unlucky") compared to ShapleyBR and its different initialization.

To sum up, for a given problem instance, the same aforementioned features may be beneficial (or conversely detrimental) either to both ShapleyGap and ShapleyBR, or to both ShapleyGap and HSVI, so that one can expect ShapleyGap's results to often lie in-between ShapleyBR and HSVI.

H. Policy Execution

In the HSVI framework,[6] a player should act greedily with respect (i) to an optimistic bound of the optimal value function for the heuristic search to converge (*cf.* the action selection presented above), and (ii) to a pessimistic bound at execution time so as to guarantee the worst-case expected return. More precisely, at execution time, if in state s, player 1 (resp. 2) should find a Nash equilibrium strategy d^L for $\Gamma^s(L)$ (resp. d^U for $\Gamma^s(U)$) and act according to $d^{L,1}$ (resp. $d^{U,2}$). That way, player 1 makes sure to get the *security level* value associated to $\Gamma^s(L)$. The following example illustrates why acting optimistically at execution time can be detrimental.

Example 1. *Let us consider the matrix game Γ^α described in Table II with its upper and lower bounds Γ^{up} and Γ^{lo}, all 3 games sharing the same NEV, 0. Yet, according to Γ^{up}, any strategy $(p, 1-p)$ of (row) Player 1 ($p \in [0,1]$) is part of a NES. If Player 1, being optimistic, plays $(p, 1-p)$ and Player 2, knowing that α is actually negative, plays $(q, 1-q)$, then the obtained value is $p\alpha q$ (≤ 0),* i.e., *a loss for player 1 if $pq \neq 0$.*

This example also shows that Γ^{up} and Γ^{lo} having identical NEV does not mean that they are equal on the support of the tentative NE, i.e., *the matrix values involved in the tentative*

[6]This discussion also holds for ShapleyGap because of the use of upper and lower bounds.

TABLE II: An example game Γ^{α} ($\alpha \in [-1,+1]$) with example upper and lower bound games Γ^{up} and Γ^{lo}, their NESs and their (NE) values.

game	matrix	NESs	value
Γ^{up}	$\begin{bmatrix} +1 & 0 \\ 0 & 0 \end{bmatrix}$	$((p,1-p),(0,1))$	0
Γ^{α}	$\begin{bmatrix} \alpha & 0 \\ 0 & 0 \end{bmatrix}$	[see text]	0
Γ^{lo}	$\begin{bmatrix} -1 & 0 \\ 0 & 0 \end{bmatrix}$	$((0,1),(q,1-q))$	0

NE. Plus, while Γ^{α} always has NEV 0, its possible NESs depend on whether α is negative, positive or null.

For player 1, this implies (i) either solving two-player normal form game $\Gamma^{lo}(s)$ when actually in state s at some point during execution, or (ii) being equipped with (a) a pre-computed decision rule in each state s visited by zsSG-HSVI, and (b) a pre-computed or on-line-computable decision rule in any other reachable state (so as to guarantee the value associated to L's initialization). Under these circumstances, player 1 has an appropriate ($\gamma^{-t}\epsilon$-rational if at time t) decision to execute even in states not actually visited by the algorithm (likewise for player 2, but relying on U), but reachable by non-trembling players. This also ensures that player 1 has appropriate—though not necessarily $\gamma^{-t}\epsilon$-optimal—decisions to execute even in states reached because player 2 acted sub-optimally, in the sense that, if player 2's hand trembles in state s at t, then player 1's expected return from s remains at least $L(s)$. Thus, zsSG-HSVI provides solutions strategies that are not near-optimal Markov perfect equilibrium, but are still robust to trembling hands.

Another consequence is that, while one could possibly exploit the fact that, if s and s' are symmetric states, then $U(s) = -L(s')$, this requirement (being able to act according to a player's pessimistic bound) prevents us from exploiting the 0 NEV of self-symmetric states in symmetric zs-SGs, *i.e.*, when both players' situations mirror each other.

V. Experiments

The experiments compare zsSG-HSVI against ShapleyBR and ShapleyGap (presented in Sec. III-B) on various problems.

A. Setup

a) Benchmark Problems: We have conducted experiments on (a) a two-player *soccer* board game [31, 17], parameterized by its dimensions $w \times h$ and the initial location of the Max player (x_0, y_0) (the Min player being placed symmetrically); (b) a router/server *flow control* problem [9, 17], parameterized by the maximum buffer size $bMax$ and initial buffer length $bInit$; (c) (discounted) *Alesia*, a prototypical war game [32], parameterized by the radius R of the field and the two players' initial numbers of units ($units$); and (d) *Alesia2*, a novel variant where, at each time step, each player gets a reward equal to its (positive or negative) progress. γ was always set to 0.95 and ϵ to 0.001. These problems are detailed in Appendix C.

b) Algorithms: HSVI, ShBR and ShGp denote the (zsSG-)HSVI, ShapleyBR and ShapleyGap algorithms for zs-SGs with trivial initializations (for ShBR: $\forall s,\ V(s) = \frac{R_{\max}+R_{\min}}{2(1-\gamma)}$).

$Alg^{1\to 2}$ (resp. $Alg^{2\to 1}$) denotes the (sequential) version of $Alg \in \{\text{HSVI, ShBR, ShGp}\}$ where player 1 (resp. 2) moves first, obtained by simply replacing the Nash update operators by sequential ones, and, for HSVI, by letting player 1 (resp. 2) act greedily w.r.t. U (resp. L). These sequential versions have the same convergence properties as the "simultaneous" and MDP ones.

Finally, $Alg+$ denotes the version of $Alg \in \{\text{HSVI, ShGp}\}$ where U (resp. L) is initialized with the upper-bound (resp. lower-bound) obtained with $Alg^{2\to 1}$ (resp. $Alg^{1\to 2}$) either *a priori* (if Alg = ShGp, with the same error ϵ), or on-demand (if Alg = HSVI, with the same error $\gamma^{-\delta}\epsilon$). Note that we do not mix algorithm schemas. Also, ShapleyGap's stopping criterion relies on the local gap at s_0, unless used as a (precomputed) initialization.

B. Results

The 3 sequential algorithms (player 1 moving first) and 3+2 simultaneous algorithms have been applied with a 3600s timeout on various instances of the 4 problems. All experiments have been conducted on a core i7 @ 1.9GHz with 4GB maximum heap size, using IBM CPlex to solve LPs. The Java source code is available at https://buffet.gitlabpages.inria.fr/HS4SG/ . Table III presents the results obtained on a subset of instances in terms of time to convergence, number of playouts (for HSVI) or iterations (ShGp/BR), and number of states visited at least once (error at s_0 is ≤ 0.001). Table IV presents results on the hard instances of the Soccer problem, adding the final error at s_0.

Note that, in HSVI variants, the *number of playouts* (P) is a number of generated trajectories, and thus should not be compared with the number of iterations in ShGp and ShBR variants (I).

1) Sequential Games: In the sequential case, HSVI$^{1\to 2}$ is often dominated by ShBR$^{1\to 2}$ and ShGp$^{1\to 2}$ on small problem instances. In the contrary, HSVI$^{1\to 2}$ appears to be dominating ShBR$^{1\to 2}$ and ShGp$^{1\to 2}$ on the largest problem instances (except on two Soccer problem instances that require visiting a large portion of the state space so that HSVI's natural pruning is not useful). An explanation is that, while only trivial initializations are used, HSVI$^{1\to 2}$ usually ignores many states, which often allows for faster convergence. On some overly large problems, this even prevents HSVI$^{1\to 2}$ from experiencing the out-of-memory errors of its competitors. In any case, *i.e.*, regardless whether HSVI$^{1\to 2}$ is better or worse than ShGp$^{1\to 2}$ and ShBR$^{1\to 2}$, ShGp$^{1\to 2}$'s convergence time is often in-between that of HSVI$^{1\to 2}$ and ShBR$^{1\to 2}$ (to put it differently, ShGp$^{1\to 2}$ is rarely better than both HSVI$^{1\to 2}$ and ShBR$^{1\to 2}$, or by a small margin), which was expected, as discussed in Sec. IV-G. As can be noted, some problem instances need few iterations to converge, whatever the algorithm: In the largest instances of Alesia, this is due to the possibility for player 1 to easily defend itself and draw.

TABLE III: Running time (in seconds), number of playouts (P), iterations (I) and visited states (N) for easy instances of the Soccer problem, and various instances of the FlowControl, Alesia, and Alesia2 problems, solved with serialized and simultaneous versions of HSVI, ShGp and ShBR, using $\gamma = 0.95$ and $\epsilon = 0.001$. In **bold**, shortest convergence times (distinguishing sequential and simultaneous games).

Algorithm →					HSVI$^{1\to2}$			ShGp$^{1\to2}$		ShBR$^{1\to2}$		HSVI			HSVI+			ShGp		ShGp+		ShBR	
Problem ↴				$\lvert\mathcal{S}\rvert$	t	P	N	t	I	t	I	t	P	N	t	P	N	t	I	t	I	t	I
w	h	x_0	y_0	**[Soccer]**																			
5	4	0	0	762	3	2148	762	1	128	**0**	5	56	656	759	16	79	223	10	26	10	25	**2**	11
5	4	0	1	762	4	2225	762	1	75	**0**	5	58	717	761	16	96	229	11	26	10	25	**2**	11
5	4	1	0	762	**0**	89	270	**0**	4	**0**	5	4	42	123	**0**	1	0	2	4	2	3	2	11
5	4	1	1	762	3	1799	762	**0**	75	**0**	5	62	704	761	17	94	228	10	25	9	24	**2**	11
5	4	2	0	762	**0**	51	156	**0**	3	**0**	5	2	27	89	**0**	1	0	1	3	1	2	2	11
5	4	2	1	762	**0**	78	233	**0**	3	**0**	5	4	47	116	**0**	1	0	1	3	1	2	2	11
5	4	3	0	762	**0**	15	50	**0**	2	**0**	5	**0**	6	14	**0**	1	0	1	2	1	1	2	11
5	4	3	1	762	**0**	20	56	**0**	2	**0**	5	**0**	8	20	**0**	1	0	1	2	1	1	2	11
5	4	4	0	762	**0**	1	2	**0**	1	**0**	5	**0**	1	2	**0**	1	0	1	1	**0**	0	2	11
5	4	4	1	762	**0**	1	2	**0**	1	**0**	5	**0**	1	2	**0**	1	0	1	1	**0**	0	2	11
10	6	0	0	7.1K	19	14036	7.1K	6	78	**0**	10	588	6169	7.1K	64	175	1.3K	118	34	116	33	**29**	18
10	6	0	2	7.1K	20	13399	7.1K	6	77	**0**	10	598	6320	7.1K	78	245	1.6K	118	34	116	33	**29**	18
10	6	4	0	7.1K	2	523	2.3K	1	6	**0**	10	89	798	3.3K	**2**	1	0	24	6	21	5	29	18
10	6	4	2	7.1K	20	10725	7.1K	4	59	**0**	10	649	6644	7.1K	72	166	1.2K	108	29	103	28	**29**	18
10	6	8	0	7.1K	**0**	15	73	**0**	2	**0**	10	**0**	6	18	**0**	1	0	11	2	7	1	29	18
10	6	8	2	7.1K	**0**	20	75	**0**	2	**0**	10	**0**	8	16	**0**	1	0	10	2	7	1	29	18
$bMax$		$bInit$		**[FlowCtrl]**																			
100		10	101	**0**	130	78	**0**	181	**0**	98	18	129	77	**0**	1	1	8	181	7	168	2	98	
100		90	101	**0**	130	78	**0**	181	**0**	98	18	129	77	**0**	1	1	8	181	7	168	2	98	
500		10	501	**0**	177	99	**0**	237	**0**	238	33	156	138	**0**	1	0	50	237	33	165	24	238	
500		90	501	**0**	177	99	**0**	237	**0**	238	33	156	138	**0**	1	0	48	237	34	165	25	238	
1000		10	1.1K	**0**	220	118	**0**	262	**0**	264	40	175	133	**1**	1	0	107	262	67	165	53	264	
1000		90	1.1K	**0**	220	118	**0**	262	**0**	264	40	175	133	**0**	1	0	106	262	68	165	53	264	
5000		10	5.1K	**0**	294	103	3	319	2	322	64	233	173	**1**	1	0	647	319	334	165	332	322	
5000		90	5.1K	**0**	294	103	3	319	2	322	64	233	173	**1**	1	0	648	319	334	165	324	322	
R		$units$		**[Alesia]**																			
2		8	405	**0**	54	135	**0**	1	**0**	3	**0**	63	126	**0**	10	15	**0**	3	**0**	2	**0**	4	
3		14	1.6K	**0**	228	702	**0**	2	**0**	6	2	297	653	**0**	23	72	3	4	3	3	3	7	
4		20	4.0K	**0**	550	2.1K	**0**	3	**0**	8	9	760	2.0K	**3**	90	310	13	6	13	5	9	10	
5		27	8.7K	2	1232	4.8K	**1**	3	2	11	30	1782	4.7K	**12**	221	931	46	8	38	7	29	13	
6		33	16K	**5**	2175	8.9K	**5**	4	8	14	75	3244	8.6K	**32**	310	1.6K	100	10	96	9	65	15	
7		40	26K	15	3598	16K	**12**	4	23	16	185	5497	16K	**76**	397	2.6K	210	11	204	10	145	18	
50		30	98K	**0**	58	1.8K	11	1	99	30	28	1424	5.7K	**0**	1	0	580	9	531	8	795	30	
60		35	157K	**0**	68	2.4K	25	1	252	35	49	1969	8.2K	**0**	1	0	1096	10	1039	9	1584	35	
70		40	238K	**0**	78	3.2K	53	1	571	40	77	2646	11K	**1**	1	0	1973	11	1930	10	2973	40	
R		$units$		**[Alesia2]**																			
2		8	405	**0**	68	138	**0**	1	**0**	3	**0**	97	161	**0**	24	42	**0**	4	**0**	2	**0**	4	
3		14	1.6K	**0**	281	708	**0**	1	**0**	6	3	417	758	**2**	145	300	5	6	3	3	3	7	
4		20	4.0K	**0**	592	1.9K	**0**	2	**0**	8	12	1092	2.1K	**9**	467	941	17	8	12	5	10	9	
5		27	8.7K	**1**	1119	4.1K	**1**	2	2	10	32	2285	4.6K	31	1015	2.4K	52	10	40	7	**27**	11	
6		33	16K	**2**	1887	7.4K	4	3	6	12	71	4021	8.1K	102	1968	4.4K	118	12	100	9	**63**	13	
7		40	26K	**5**	2800	13K	9	3	20	14	181	7006	14K	279	3372	7.9K	240	13	237	11	**139**	15	
50		30	98K	**2**	2082	8.7K	32	5	95	30	**61**	2901	8.3K	69	1625	4.6K	934	14	817	11	959	30	
60		35	157K	**5**	3046	14K	87	6	244	35	**104**	4320	13K	165	2686	7.7K	1838	16	1563	12	1940	35	
70		40	238K	**11**	4155	21K	194	6	570	40	**204**	6127	19K	332	3697	12K	3419	18	2857	13	3600	40	

TABLE IV: Running time (in seconds) with a 3600s timeout (TO), number of playouts (P), iterations (I) and visited states (N) for hard instances of the Soccer problem, solved with serialized and simultaneous versions of HSVI, ShGp and ShBR, using $\gamma = 0.95$ and $\epsilon = 0.001$. In **bold**, shortest convergence times (distinguishing sequential and simultaneous games), or timeout with smallest error in s_0 (e).

Algorithm →					HSVI$^{1\to2}$				ShGp$^{1\to2}$			ShBR$^{1\to2}$			HSVI				HSVI+				ShGp			ShGp+			ShBR		
Problem ↴				$\lvert\mathcal{S}\rvert$	t	P	e	N	t	I	e	t	I	e	t	P	e	N	t	P	e	N	t	I	e	t	I	e	t	I	e
w	h	x_0	y_0	**[Soccer]**																											
50	30	0	15	4.5M	TO	1.2M	.13	3.5M	TO	53	0.05	**1550**	50	0	TO	23.8K	1.2	138K	**TO**	1	**1.1**	0	TO	1	1.3	TO	0	1.3	TO	3	34.3
50	30	15	15	4.5M	TO	1.2M	.29	3.5M	TO	54	0.02	**1580**	50	0	TO	24.6K	1.4	174K	TO	1	1.1	0	TO	1	1.1	**TO**	1	**1.0**	TO	3	34.3
50	30	30	15	4.5M	**560**	0.2M	0	1.1M	1407	20	0	1566	50	0	TO	24.7K	1.3	176K	**964**	1	0	0	TO	1	1.0	TO	1	.98	TO	3	34.3
100	60	0	30	72M	**TO**	1.2M	**.18**	4.9M	—— *out of memory* ——						TO	24.0K	1.2	188K	**TO**	1	**1.1**	0	—— *out of memory* ——								
100	60	30	30	72M	**TO**	1.1M	**.37**	8.0M	—— *out of memory* ——						TO	24.6K	1.4	253K	**TO**	1	**1.2**	0	—— *out of memory* ——								
100	60	60	30	72M	**TO**	1.2M	**.38**	8.9M	—— *out of memory* ——						TO	24.6K	1.3	255K	**TO**	1	**1.2**	0	—— *out of memory* ——								

2) Simultaneous Games: Similar observations can be made while comparing HSVI, ShGp and ShBR, often (but not always) with comparable numbers of iterations and of visited states, though with much longer convergence times. The main difference thus seems to be due to the time consumption of the LP-based operator.

Let us now consider the versions with not-so-trivial initializations based on the sequential problems. ShGp+ usually performs a bit fewer iterations than its counterpart ShGp, but only saves time in FlowControl, and loses time in some other problems. HSVI+ may perform fewer or much fewer iterations than HSVI, sometimes a single one because the initial U and L are equal. Yet, except in these cases, the total running time can be much slower in HSVI+ than in HSVI (due to the many calls to HSVI$^{1\to2}$ and HSVI$^{2\to1}$, which do not appear in the table). The contrast can be observed by looking at results obtained on Alesia vs Alesia2. Thus, while relying on a heuristic search is clearly beneficial on large problem instances, whether to use HSVI or HSVI+ depends on the problem at hand.

Note also that no runs have been interrupted due to numerical instabilities when solving LPs, which happens when using a different LP solver (see also [33]).

VI. Conclusion and Future Work

The present paper proposes zsSG-HSVI, a heuristic search algorithm for discounted (two-player) zero-sum stochastic games with cycles, and non-zero rewards in any transition. The players' opposite criteria naturally led to algorithms with upper and lower bounds such as HSVI and BRTDP. The principle of optimism in the face of uncertainty is maintained, as well as that of acting (at execution time) according to the pessimistic bound. As the original HSVI, on which it is based, the resulting algorithm provably converges to an ϵ-optimal solution in finite time. To make the picture more complete and better analyze zsSG-HSVI, a variant of Shapley's algorithm is also introduced that, as zsSG-HSVI, maintains upper and lower bounds of the optimal value function and stops when the gap in-between them is below some threshold.

Experiments demonstrate that using zsSG-HSVI rather than Shapley-BR may be detrimental on small problem instances, but the benefit increases with the problem size. Also, Shapley-Gap's perfomance is typically in-between zsSG-HSVI and Shapley-BR, whichever of the two converges faster, which was expected as it mixes features of both other algorithms.

We now list several directions for future work.

a) Improvements: LPs could be solved faster by using oracle methods (as in DO$\alpha\beta$ [20]) and/or bootstrapping techniques (because similar LPs are solved when re-visiting a state). A still open question is how to handle heuristic initializations: which algorithms to use and with what precision (depending on whether pre- or on-demand computations are performed). Of course, other algorithmic schemes than HSVI could be employed, typically based on the FIND-and-REVISE schema [34], so that the main convergence results may be addressed jointly.

b) General-Sum SGs: A direction for future research is to extend the present work to solving general-sum SGs (with possibly more than two players). This expectedly means maintaining upper and lower bounds for each player. But a first issue is that of choosing an appropriate solution concept, since a given matrix game does not have a single NE value any more. If planning for all players at once—*i.e.*, assuming each will commit to the strategy assigned to him—, then any NE may be a valid solution. But what if planning for a single player without constraints on the other players' strategies?

c) POSGs: As mentioned in Sec. II, HSVI has been first directly applied to particular zero-sum (two-player) partially observable stochastic games (zs-POSGs): with one-sided partial observability [11] and with public observability [12]. These problems allow (i) not having to deal with nested beliefs, and (ii) reasoning about beliefs such that the optimal value function is convex/concave in belief-space, which allows defining generalizing value function approximators (upper- and lower-bounds) as in POMDPs.

In contrast, an objective of the present work is to pave the way to addressing general POSGs using a variant of HSVI and value function approximation as already done for Dec-POMDPs [7], *i.e.*, using a sufficient statistic of the planner called an *occupancy state*. In the case of a 2-player zero-sum POSG, this means addressing the problem as an occupancy zs-SG [35]. Then, the planning problem becomes deterministic, but the state and action spaces are continuous, thus infinite: one because it is a probability simplex, the other because it contains all stochastic decision rules. Taking inspiration from [11, 12], one shall be able to achieve ϵ-optimal convergence in finite time using recursive partitioning and exploiting the Lipschitz continuity of the value function [36].

Appendix A
Shapley Algorithms

The classical Shapley algorithm [27], here called ShapleyBR, is depicted in Algorithm 3, while the proposed ShapleyGap algorithm is depicted in Algorithm 4. With ShapleyBR, an ϵ-optimal joint policy is obtained when both players act greedily with respect to V. With ShapleyGap, an ϵ-optimal joint policy is obtained when (maximizing) Player 1 acts greedily w.r.t. L, and (minimizing) Player 2 acts greedily w.r.t. U.

Algorithm 3: ShapleyBR (asynchronous)

Fct `ShapleyBR` (ϵ)
  input : V initialized (randomly or otherwise)
  output: V $(\frac{1}{2}\frac{1-\gamma}{\gamma}\epsilon)$-optimal
  $\theta \leftarrow \frac{1}{2}\left(\frac{1-\gamma}{\gamma}\right)^2\epsilon$
  repeat
    $BR \leftarrow 0$
    for $s \in \mathcal{S}$ **do**
      $v \leftarrow V(s)$
      $V(s) \leftarrow$ `Update` (V, s)
      $BR \leftarrow \max\{BR, |V(s) - v|\}$
  until $BR \leq \theta$
  return V

Algorithm 4: ShapleyGap (asynchronous)

Fct `ShapleyGap` (ϵ)
  input : L and U lower and upper initializations
  output: L and U ϵ-optimal lower and upper bounds
  repeat
    $gap \leftarrow 0$
    for $s \in \mathcal{S}$ **do**
      if $U(s) - L(s) > \epsilon$ **then**
        $L(s) \leftarrow$ `Update` (L, s)
        $U(s) \leftarrow$ `Update` (U, s)
        $gap \leftarrow \max\{gap,$ `width` $(s)\}$
  until $gap \leq \epsilon$
  return (L, U)

TABLE V: Various update operators for the value function, *almost* ordered by resulting value. In the middle is the optimal Shapley operator. Above are operators for initializing the upper bound U. Below are operators for initializing the lower bound L.

$(\mathcal{G}^U_{\text{TRIV}}V)(s)$	$=$	$\max_{s',a^1,a^2,s"} r(s',a^1,a^2)+\gamma V(s")$	
$(\mathcal{G}^U_{\text{ORA}}V)(s)$	$=$	$\mathbb{E}_{a^2\sim\text{Unif}(\mathcal{A}^2)}\max_{a^1}$	$\Gamma^s(V)(a^1,a^2)$
$(\mathcal{G}^U_{\text{MDP}}V)(s)$	$=$	$max_{a^1}\mathbb{E}_{a^2\sim\text{Unif}(\mathcal{A}^2)}$	$\Gamma^s(V)(a^1,a^2)$
$(\mathcal{G}^U_{\text{SEQ}}V)(s)$	$=$	$\min_{a^2}\max_{a^1}$	$\Gamma^s(V)(a^1,a^2)$
$(\mathcal{H}V)(s)$	$=$	$\min_{d^2}\max_{a^1}$	$\Gamma^s(V)(a^1,d^2)$
$=\text{NEV}(\Gamma^s(V))$	$=$	$\max_{d^1}\min_{a^2}$	$\Gamma^s(V)(d^1,a^2)$
$(\mathcal{G}^L_{\text{SEQ}}V)(s)$	$=$	$\max_{a^1}\min_{a^2}$	$\Gamma^s(V)(a^1,a^2)$
$(\mathcal{G}^L_{\text{MDP}}V)(s)$	$=$	$\min_{a^2}\mathbb{E}_{a^1\sim\text{Unif}(\mathcal{A}^1)}$	$\Gamma^s(V)(a^1,a^2)$
$(\mathcal{G}^L_{\text{ORA}}V)(s)$	$=$	$\mathbb{E}_{a^1\sim\text{Unif}(\mathcal{A}^1)}\min_{a^2}$	$\Gamma^s(V)(a^1,a^2)$
$(\mathcal{G}^L_{\text{TRIV}}V)(s)$	$=$	$\min_{s',a^1,a^2,s"} r(s',a^1,a^2)+\gamma V(s")$	

APPENDIX B
UPPER AND LOWER BOUND INITIALIZATIONS

Table I only presents a limited selection of operators that can be used to compute upper and lower bounds of the optimal value function. For completeness, Table V shows an extended list of candidate operators for both bounds, which we also describe with words as follows:

$\mathcal{G}^{[L/U]}_{\text{SEQ}}$**:** a sequential game approximation—also known as *serialization*—, *i.e.*, U being the optimal value if player 1 acts before player 2, and L the opposite (the second player knowing the action choice of the first player);

$\mathcal{G}^{[L/U]}_{\text{MDP}}$**:** an MDP approximation, *i.e.*, U (respectively L) being the optimal value function of the MDP induced by assigning to player 2 (resp. player 1) any fixed strategy—*e.g.*, random actions;

$\mathcal{G}^{[L/U]}_{\text{ORA}}$**:** an MDP approximation with an oracle that predicts the action of the random opponent (but not nature's move); or

$\mathcal{G}^{[L/U]}_{\text{TRIV}}$**:** trivial approximations, *e.g.*, $\forall s$,
$U(s) = \frac{1}{1-\gamma}\max_{s',a^1,a^2} r(s',a^1,a^2)$ and
$L(s) = \frac{1}{1-\gamma}\min_{s',a^1,a^2} r(s',a^1,a^2)$.

Within Table V, the operators within some box (in-between two lines) return "greater or equal" functions than the operators within a lower box. Within the two boxes containing three operators, the only known ordering relations are: $\mathcal{G}^U_{\text{ORA}}V \geq \mathcal{G}^U_{\text{MDP}}V$ and $\mathcal{G}^L_{\text{MDP}}V \geq \mathcal{G}^L_{\text{ORA}}V$. Of course, one issue is whether deriving these bounding value functions is worth the computational effort. The fixed points of the $\mathcal{G}_{\text{TRIV}}$ operators can be computed in constant time (see above formulas), and can serve as initializations for computing the fixed points of the other operators, *e.g.*, again by heuristic search. The $\mathcal{G}_{\text{ORA}}$ operators are not worth considering as they would provide loser bounds than other operators at the same computational cost.

APPENDIX C
BENCHMARK PROBLEMS

The present section provides details on the benchmark problems used in the experiments. They may differ from the original versions in the choice of certain parameters, in the use of a discount factor, and in the possibility of resetting (rather than ending in a terminal state).

A. Two-Player Soccer Game

This description (of the present implementation) is largely inspired by Lagoudakis and Parr [17], itself inspired by Littman [31].

The playground is a $w \times h$ rectangular grid representing an horizontal soccer field: player 1 tries to score by getting out of the grid with the ball through its left side (and respectively through the right side for player 2). Each player is on his own cell, cell coordinates being noted $(x, y) \in \{1, \ldots, W\} \times \{1, \ldots, H\}$. The ball is always possessed by one of the 2 players. The initial location of the Max player is (x_0, y_0), the Min player being placed symmetrically.

Each player has a choice of 5 actions: up (U), down (D), left (L), right (R), and stand (S). At each time step the players decide on which actions they are going to take and then a fair coin is flipped to determine which player moves first. The players move one at a time in the order determined by the coin flip.

- If a player collides with a border or with another player during a move, the player remains in his current position.
- If the player with the ball (attacker) runs into the opponent (defender), the ball is passed to the opponent.
- Therefore, the only way for the defender to steal the ball is to be in the square into which the attacker intends to move.
- The attacker can cross the goal line and score into the defender's goal, however the players cannot score into their own goals.

Scoring for player 1 (resp. 2) results in an immediate reward of $+1$ (resp. -1) and a restart of the game. The discount factor for the problem is set to $\gamma = 0.95$, which encourages early scoring.

In Soccer, the number of states is:

$$|\mathcal{S}| = (w \cdot h) \cdot (w \cdot h - 1) \cdot 2 + 2.$$

B. Flow Control

This description of the *Router/server flow control problem* is largely inspired by Lagoudakis and Parr [17], itself inspired by Altman [9].

A router is trying to control the flow of jobs into a server buffer (of maximum buffer size $bMax$ and initial buffer length $bInit$) under unknown, and possibly changing, service conditions. This problem can be modeled as an MDP with the server being an uncertain part of the environment. However, to provide worst-case guarantees the router can view the server as an opponent that plays against him. This viewpoint enables to router to adopt control policies that perform well under worst-case/changing service conditions.

The system state is described by the current length of the buffer. Available actions are:

- for the router: low (L) and high (H), corresponding to a low (PA_L) and a high (PA_H) probability of a job arrival to the buffer at the current time step, with $0 < PA_L < PA_H \leq 1$;
- for the server: low (L) and high (H), corresponding to a low (PD_L) and a high (PD_H) probability of a job departure from the buffer at the current time step, with $0 \leq PD_L < PD_H < 1$.

Once the agents have picked their actions, the size of the buffer is adjusted to a new state according to the chosen probabilities, and the game continues.

The immediate cost $R(s, a, o)$ for each transition depends on the current state s and the actions a and o of the agents: $R(s, a, o) = c(s) + \alpha \times PA_a + \beta \times PD_o$, where $c(s)$ is a real non-decreasing convex function, $\alpha \leq 0$, and $\beta \geq 0$. $c(s)$ is related to the holding cost per time step in the buffer, α is related to the reward for each incoming job, and β is related to the cost for the quality of service. The router attempts to minimize the expected discounted cost, whereas the server strives to maximize it. The discount factor is set to $\gamma = 0.95$. Under these conditions, the optimal policies can be shown to have an interesting threshold structure, with mostly deterministic choices and randomization in at most one state [9]. However, the exact thresholds and probabilities cannot be easily determined analytically from the parameters of the problems which might actually be unknown. These facts make the problem suitable for learning with function approximation. We tested our method on a buffer of length 100 with: $PA_L = 0.2$, $PA_H = 0.9$, $PD_L = 0.1$, $PD_H = 0.8$, $c(s) = 10^{-4}s^2$, $\alpha = -0.1$, $\beta = +1.5$. With these settings neither a full nor an empty buffer is desirable for either player. Increasing the buffer size beyond 100 does not cause any change in the final policies, but will require more training data to cover the critical area (0–100).

In FlowControl, the number of states is:

$$|\mathcal{S}| = bMax + 1.$$

C. Alesia

This description of the Alesia game—illustrated by Fig. 1—is largely inspired by Meyer et al. [32].

In this game, also known as "Footsteps", "Citadel" or "Quo Vadis?" [37], two citadels, each belonging to one player, are placed at distance $D = 2R + 1$ from each other. A mark is initially placed in-between the citadels (in the middle). Each player starts with 50 units (input parameter $units$).

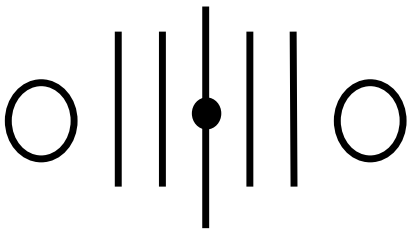

Fig. 1: Alesia's game field/board (copied from [32])

At each time step, both players simultaneously bid/engage (and definitely consume) some of their remaining units (at least 1 unit). The player with the highest bid pushes the marker one distance unit towards the opponent's citadel. In case of equal bids, the marker does not move.

A player wins if he manages to move the marker to the adversary's citadel. Otherwise it is a draw. A win (respectively a loss) leads to a reward of $+1$ (resp. -1).

In Alesia, the number of states is:

$$|\mathcal{S}| = (2 \cdot R + 1) \cdot (units + 1)^2.$$

Note: See also Oshi-Zumo (japanese for "pushing wrestler") [33], which is very similar, or the paper game Tennis.[7]

Alesia2: Soccer and Alesia share strong similarities as, in both cases, players are on the same field/playground and each would like to reach one side while confronting the other player. In both cases, a large part of the state space has to be covered for the value function to converge, even with a heuristic search algorithm, due to non-zero rewards being given only when a player reaches his target end of the field. In Alesia2, at each time step the instant reward corresponds to the x position of the "pack". As a consequence, one does not need to cover a large part of the state space to find optimal strategies for both players.

REFERENCES

[1] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*. Englewood Cliffs, NJ: prentice Hall, 2010.

[2] V. Bulitko and G. Lee, "Learning in real-time search: A unifying framework," *Journal of Artificial Intelligence Research*, vol. 25, pp. 119–157, 2006.

[3] B. Bonet and H. Geffner, "Labeled RTDP: Improving the convergence of real-time dynamic programming," in *Proc. of the 13th Int. Conf. on Automated Planning and Scheduling*, 2003, pp. 12–21.

[4] E. A. Hansen and S. Zilberstein, "LAO*: A heuristic search algorithm that finds solutions with loops," *Artificial Intelligence*, vol. 129, no. 1–2, pp. 35–62, 2001.

[5] T. Smith and R. Simmons, "Point-based POMDP algorithms: Improved analysis and implementation," in *Proc. of the 21st Conf. on Uncertainty in Artificial Intelligence*, 2005, pp. 542–549.

[7] https://en.wikipedia.org/wiki/Tennis_(paper_game)

[6] D. Szer, F. Charpillet, and S. Zilberstein, "MAA*: A heuristic search algorithm for solving decentralized POMDPs," in *Proc. of the 21st Conf. on Uncertainty in Artificial Intelligence*, 2005, pp. 576–583.

[7] J. Dibangoye, C. Amato, O. Buffet, and F. Charpillet, "Optimally solving Dec-POMDPs as continuous-state MDPs," *Journal of Artificial Intelligence Research*, vol. 55, pp. 443–497, 2016.

[8] D. M. Roijers, P. Vamplew, S. Whiteson, and R. Dazeley, "A survey of multi-objective sequential decision-making," *Journal of Artificial Intelligence Research*, vol. 48, pp. 67–113, 2013.

[9] E. Altman, "Flow control using the theory of zero-sum Markov games," *IEEE Trans. on Auto. Control*, vol. 39, no. 4, pp. 814–818, 1994.

[10] S. Ontañón, N. A. Barriga, C. R. Silva, R. O. Moraes, and L. H. Lelis, "The first MicroRTS artificial intelligence competition." *AI Magazine*, vol. 39, no. 1, 2018.

[11] K. Horák, B. Bošanský, and M. Pěchouček, "Heuristic search value iteration for one-sided partially observable stochastic games," in *Proc. of the 31st AAAI Conf. on Artificial Intelligence*, 2017, pp. 558–564.

[12] K. Horák and B. Bošanský, "Solving partially observable stochastic games with public observations," in *Proc. of the 33rd AAAI Conf. on Artificial Intelligence*, 2019, pp. 2029–2036.

[13] H. B. McMahan, M. Likhachev, and G. J. Gordon, "Bounded real-time dynamic programming: RTDP with monotone upper bounds and performance guarantees," in *Proc. of the 22nd Int. Conf. on Machine Learning*, 2005, pp. 569–576.

[14] T. E. S. Raghavan and J. A. Filar, "Algorithms for stochastic games – a survey," *Zeitschrift für Operations Research*, vol. 35, no. 6, pp. 437–472, Nov 1991.

[15] M. Littman and C. Szepesvári, "A generalized reinforcement learning model: Convergence and applications," in *Proc. of the 13th Int. Conf. on Machine Learning*, 1996, pp. 310–318.

[16] M. Bowling and M. Veloso, "Multiagent learning using a variable learning rate," *Artificial Intelligence*, vol. 136, no. 2, pp. 215–250, 2002.

[17] M. G. Lagoudakis and R. Parr, "Value function approximation inf zero-sum Markov games," in *Proc. of the 18th Conf. on Uncertainty in Artificial Intelligence*, 2002, pp. 283–292.

[18] J. Pérolat, B. Piot, M. Geist, B. Scherrer, and O. Pietquin, "Softened approximate policy iteration for Markov games," in *Proc. of the 33rd Int. Conf. on Machine Learning*, ser. Proc. of Machine Learning Research, vol. 48, 2016, pp. 1860–1868.

[19] A. Saffidine, H. Finnsson, and M. Buro, "Alpha-Beta pruning for games with simultaneous moves," in *Proc. of the 26th AAAI Conf. on Artificial Intelligence*. AAAI Press, Jul. 2012, pp. 556–562.

[20] B. Bošanský, V. Lisý, M. Lanctot, J. Čermák, and M. H. M. Winands, "Algorithms for computing strategies in two-player simultaneous move games," *Artificial Intelligence*, vol. 237, pp. 1–40, 2016.

[21] D. Churchill, A. Saffidine, and M. Buro, "Fast heuristic search for RTS game combat scenarios," in *Proc. of the 8th AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment*, 2012, pp. 112–117.

[22] R. O. Moraes, J. R. H. Mariño, and L. H. S. Lelis, "Nested-greedy search for adversarial real-time games," in *Proc. of the 14th AAAI Conf. on Artificial Intelligence and Interactive Digital Entertainment*, 2018, pp. 67–73.

[23] E. Solal, "Stochastic games," *Encyclopedia of Database Systems*, 2009.

[24] T. Smith, "Probabilistic planning for robotic exploration," Ph.D. dissertation, The Robotics Institute, Carnegie Mellon University, 2007.

[25] J. Nash, "Equilibrium points in n-person games," *Proc. of the National Academy of Science*, vol. 36, no. 1, pp. 48–49, 1950.

[26] J. von Neumann, "Zur Theorie der Gesellschaftsspiele," *Math. Annalen*, vol. 100, 1928.

[27] L. S. Shapley, "Stochastic games," *Proc. of the National Academy of Science*, vol. 39, no. 10, pp. 1095–1100, 1953.

[28] ——, "Some topics in two person games," *Annals of Mathematical Studies*, vol. 5, pp. 1–28, 1964.

[29] R. Bellman, "The theory of dynamic programming," *Bulletin of the American Mathematical Society*, vol. 60, no. 6, pp. 503–515, 1954.

[30] D. Bertsekas and J. Tsitsiklis, *Neurodynamic Programming*. Athena Scientific, 1996.

[31] M. Littman, "Markov games as a framework for multi-agent reinforcement learning," in *Proc. of the 11th Int. Conf. on Machine Learning*, 1994, pp. 157–163.

[32] C. Meyer, J.-G. Ganascia, and J.-D. Zucker, "Learning strategies in games by anticipation," in *Proc. of the 15th Int. Joint Conf. on Artificial Intelligence*, 1997, pp. 698–707.

[33] M. Buro, "Solving the Oshi-Zumo game," in *Advances in Computer Games Conf. 10*, 2003, pp. 361–366.

[34] B. Bonet and H. Geffner, "Faster heuristic search algorithms for planning with uncertainty and full feedback," in *Proc. of the 18th Int. Joint Conf. on Artificial Intelligence*, 2003, pp. 1233–1238.

[35] O. Buffet, J. Dibangoye, A. Delage, A. Saffidine, and V. Thomas, "On Bellman's optimality principle for zs-POSGs," *Computing Research Repository*, vol. abs/2006.16395, 2020.

[36] M. Fehr, O. Buffet, V. Thomas, and J. Dibangoye, "ρ-POMDPs have Lipschitz-continuous ϵ-optimal value functions," in *Advances in Neural Information Processing Systems 31*, 2018, pp. 6933–6943.

[37] R. Morris and T. Watson, "Evolving strategies for the game footsteps," in *Proc. of the 2008 UK Workshop on Computational Intelligence*, 2008.