Approximating Nash Equilibria in Finite-Horizon Multi-Adversarial Team Markov Games

Prasanna Maddila[1] **Régis Sabbadin**[1] **Meritxell Vinyals**[1]

[1]Univ Toulouse, INRAE, MIAT, Castanet-Tolosan, France

Abstract

Multi-Adversarial Team Games (MATGs) extend the canonical normal-form framework of von Stengel and Koller – where a team of players sharing a common objective but unable to coordinate faces a single adversary – to settings involving multiple independent adversaries. From an algorithmic perspective, MATGs are notable as one of the few game classes admitting polynomial-time algorithms for computing ε-Nash equilibria. However, no complexity or algorithmic guarantees are known for the finite-horizon Markovian generalisation of MATGs. This paper establishes positive and negative results for approximating *non-stationary* Nash equilibria in finite-horizon Multi-Adversarial Team Markov Games. Specifically, we provide a polynomial-time algorithm for the single-adversary case, prove PPAD-hardness for the multiple-adversary case and present a polynomial-time algorithm for the multiple-adversary setting with additive transitions. We also provide an empirical evaluation of the proposed algorithms.

1 INTRODUCTION

Multi-Adversarial Team Games (MATGs) [Maddila et al., 2026a] model strategic interactions in which a team of agents – sharing a common objective but unable to coordinate their strategies – faces multiple independent adversaries simultaneously. The single-adversary case was first introduced by Von Stengel and Koller [1997], and has been extensively studied in the subsequent literature [Anagnostides et al., 2023, Kalogiannis et al., 2023].

The lack of team coordination arises naturally in many real-world settings, including anti-poaching [Lam et al., 2023, Maddila et al., 2024], robotic planning [McMahan et al., 2003] and hider-seeker games [Halvorson et al., 2009]. In these examples, communication is either forbidden (e.g., by game rules) or simply impractical (e.g., due to security constraints or the costly overload induced in large systems). Given the inability to coordinate, solving MATGs amounts to computing Nash equilibria (NE) in a multi-player game, where the players are the adversaries and the team members.

On the other hand, approximating NE in general-sum normal-form games is PPAD-complete, even for two players [Daskalakis et al., 2009, Chen et al., 2009]. MATGs are among the few multi-player (> 2) normal-form games where NE can be approximated in polynomial time, alongside zero-sum polymatrix [Cai et al., 2016], potential games [Monderer and Shapley, 1996, Anagnostides et al., 2022], and Adversarial Team Games (ATGs) [Anagnostides et al., 2023]. Specifically, Maddila et al. [2026a] generalise Anagnostides et al. [2023]'s results on ATGs to provide the MATG-GDM algorithm which approximates NE in time polynomial in the game parameters and $1/\varepsilon$.

This paper generalises normal-form MATGs by focusing on MATGs played over a finite horizon with Markovian state transitions. Finite horizon Markov Games admit even fewer subclasses that admit polynomial-time algorithms for NE approximation. Indeed, intractability results carry over directly to the finite-horizon setting. In contrast, positive results do not necessarily extend: even when a normal-form game admits a polynomial-time algorithm, NE approximation may be computationally intractable in the finite-horizon Markovian setting. Surprisingly, while the infinite-horizon Markovian extension of (single-adversary) ATGs admits a polynomial-time algorithm for NE approximation [Kalogiannis et al., 2023, 2024], the finite-horizon Markovian case has so far remained unexplored in the literature.

In this context, this paper addresses the following question: *Can we design polynomial-time algorithms for approximating (non-stationary) NE in finite-horizon Adversarial Markov Team Games, in both single-adversary and multi-adversary settings?* It is worth noting that by addressing this question, we also address the question asked by Kalogiannis

Proceedings of the 42nd Conference on Uncertainty in Artificial Intelligence (UAI 2026), PMLR 337:4232–4251.

and Panageas [2023]: *Are there any other settings of Markov games that encompass both competition and coordination while maintaining the tractability of NE computation?*

Original contributions. *First*, we show that ATMGs (single-adversary) admit a polynomial-time algorithm for approximating non-stationary NE. *Second*, we show that even with 2 adversaries, computing approximate NE in MATMGs is PPAD-Hard. This motivates the search for tractable subclasses. In particular, we focus on MATMGs with *additive transitions*, a structure that generalises many common transition dynamics, including single controller [Parthasarathy and Raghavan, 1981] and switching controller [Vrieze et al., 1983] games. We show that this large class of games also admits a fully polynomial time approximation scheme to calculate approximate NE. *Lastly*, we empirically evaluate our algorithm on finite-horizon MATMGs with additive transitions, demonstrating strong scalability with respect to the number of adversaries, the number of states, and the horizon, at various precision levels.

2 RELATED WORK

Finite-horizon Markov games are typically solved by backward induction, also referred to as Nash Value Iteration (Nash-VI) [Filar and Vrieze, 2012, Kearns et al., 2000]. This method works backward in time by approximating the NE of a normal-form stage game at each time-step and state; these are particular games whose payoffs combine immediate rewards with the continuation values from future stages. Consequently, using Nash-VI for finite-horizon Markov Games requires the stage games themselves to admit a polynomial-time algorithm for NE approximation. The choice of such tractable stage-game structures is thus limited by the number of known tractable normal-form games classes. Let us list these tractable classes.

In the Markov setting, Kalogiannis et al. [2023, 2024] have proposed polynomial-time algorithms that approximate *stationary* NE in infinite-horizon, single-adversary Adversarial Team Markov Games (ATMGs). However, results in the infinite-horizon setting do not carry over immediately to the finite-horizon, rendering this line of work orthogonal to ours.

Kalogiannis and Panageas [2024] consider a generalisation of single-adversary ATMGs in the finite-horizon setting, termed Adversarial Reward-Potential Markov games, where the team members rewards in each state admit a potential function and are not necessarily identical. They show that when the transitions are *additive*, Nash-VI finds approximate NE in polynomial time for this class of games. In contrast, we show that for finite-horizon ATMGs with *shared* (i.e. identical) team rewards, there exists a polynomial-time algorithm for approximating non-stationary NE, without any additional structural assumptions. Park et al. [2023] study Nash-VI for finite-horizon (multi-player) zero-sum Markov Games with Networked Separable Interactions i.e. Markov games with additive transition dynamics whose stage games are zero-sum and polymatrix. Our work builds along the same lines, but focuses on finite-horizon MATMGs, a class of games not previously explored in the literature.

Additive transitions generalise turn-based [Daskalakis et al., 2023], single-controller [Parthasarathy and Raghavan, 1981] and switching controller [Vrieze et al., 1983] transitions. They have attracted sustained attention over the past four decades as one of the few nontrivial structural assumptions that can circumvent the intractability of approximating NE in Markov games [Raghavan et al., 1985, Thuijsman and Raghavan, 1997, Küenle, 1999, Kalogiannis and Panageas, 2023, Jaśkiewicz and Nowak, 2024]. For example, computing a non-stationary ε-NE in zero-sum polymatrix Markov games is PPAD-hard in general; however, under additive transitions, polynomial-time algorithms are available [Kalogiannis and Panageas, 2023, Park et al., 2023]. Our work extends this list of positive cases by proving that, in finite-horizon MATMGs with additive transitions, a non-stationary ε-NE can be computed in polynomial time.

In practical applications, MATMGs provide a natural modeling framework for settings in which a team of players interact with multiple independent adversaries. Such scenarios arise frequently in applications, particularly in law enforcement and security domains (see, e.g., [Tambe, 2011]). Examples satisfying the more general Additive Transition assumption are comparatively less common in the literature. Nevertheless, additive transitions strictly generalize the more widely studied single-controller and switching-controller models, which are known to hold in a variety of domains [Zhang et al., 2018, Eldosouky et al., 2016].

3 FINITE-HORIZON MULTI-ADVERSARIAL TEAM MARKOV GAMES

We first define the Finite-Horizon Multi-Adversarial Team Markov Game (MATMG) framework.

Definition 1 (Finite-horizon MATMG)**.** *A finite horizon Multi-Adversarial Team Markov Game (FH-MATMG) is defined by a tuple* $\Gamma = \langle \mathcal{S}, \mathcal{N}, \mathcal{M}, (\mathcal{A}_i)_{i\in\mathcal{N}}, (\mathcal{B}_j)_{j\in\mathcal{M}}, (r_j)_{j\in\mathcal{M}}, \mathbb{P}, H, \rho \rangle$. Γ *consists of a finite set* $\mathcal{S}$ *of states with cardinality* $S = |\mathcal{S}|$*, a finite set* $\mathcal{N} = \{1, \ldots, n\}$ *of team agents and a finite set* $\mathcal{M} = \{1, \ldots, m\}$ *of adversarial agents. Each team agent* $i \in \mathcal{N}$ *(resp. adversary* $j \in \mathcal{M}$*) has a finite strategy set* $\mathcal{A}_i$ *(resp.* $\mathcal{B}_j$*). The joint team strategy space is denoted as* $\mathcal{A} = \prod_{i\in\mathcal{N}} \mathcal{A}_i$*, and the joint adversary strategy space is denoted as* $\mathcal{B} = \prod_{j\in\mathcal{M}} \mathcal{B}_j$*.* $\mathbf{a} = (a_1, \ldots, a_n) \in \mathcal{A}$ *is a strategy profile of the team, and* $\mathbf{b} = (b_1, \ldots, b_m) \in \mathcal{B}$*, a strategy profile of adversaries.*

Each adversary $j \in \mathcal{M}$ has a reward function[1] $r_j : \mathcal{S} \times \mathcal{A} \times \mathcal{B}_j \to [0, 1]$. *Adversary reward functions are independent of other adversaries' strategies. Team members have identical reward functions $r_i : \mathcal{S} \times \mathcal{A} \times \mathcal{B} \to \mathbb{R}$ where $r_i(s, \mathbf{a}, \mathbf{b}) = r_{team}(s, \mathbf{a}, \mathbf{b})$, $\forall i \in \mathcal{N}$. The game is zero-sum in the sense that $\sum_{j \in \mathcal{M}} r_j(s, \mathbf{a}, b_j) + r_{team}(s, \mathbf{a}, \mathbf{b}) = 0$. $\mathbb{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{B} \to \Delta(\mathcal{S})$ is the transition function; $\mathbb{P}(s'|s, \mathbf{a}, \mathbf{b})$ is the probability of transitioning to state s' given that the team strategy profile $\mathbf{a}$ and the adversaries strategy profile $\mathbf{b}$ are selected in state s. $H \in \mathbb{N}$ is the horizon of the problem. $\Delta(\mathcal{S})$ is the probability simplex over $\mathcal{S}$. $\rho \in \Delta(\mathcal{S})$ is the initial state distribution and is not necessarily full support on $\mathcal{S}$.*

$-i$ denotes the set of team members excluding agent i (analogously for $-j$, set of adversaries excluding adversary j).

Remark 1. *To simplify notation, we assume that transitions $\mathbb{P}(\cdot|\cdot, \cdot, \cdot)$ and rewards $r_j(\cdot, \cdot, \cdot)$ are stationary, i.e. independent of the decision step $h \in [H] = \{1, \dots, H\}$. This assumption is without loss of generality, since a finite-horizon non-stationary MATMG can be represented in $O(H)$ times more space as a stationary MATMG, by including the time step h as an additional state variable and adapting transitions and rewards accordingly.*

Policies For any $i \in \mathcal{N}$ we define a *non-stationary* policy $\mathbf{x}_i := \{\mathbf{x}_i^{s,h} \in \Delta(\mathcal{A}_i), \forall (s, h) \in \mathcal{S} \times [H]\}$ (resp. $\mathbf{y}_j := \{\mathbf{y}_j^{s,h} \in \Delta(\mathcal{B}_j), \forall (s, h) \in \mathcal{S} \times [H]\}$ for $j \in \mathcal{M}$).

We denote the joint policies of the team as $\mathbf{x} = (\mathbf{x}_i)_{i \in \mathcal{N}}$ and that of the adversaries as $\mathbf{y} = (\mathbf{y}_j)_{j \in \mathcal{M}}$.

We overload notation so that r_j is not only the reward function of adversary j but also the mixed extension of such reward: $r_j(s, \mathbf{x}^{s,h}, \mathbf{y}_j^{s,h}) = \mathbb{E}_{(\mathbf{a}, b_j) \sim (\mathbf{x}^{s,h}, \mathbf{y}_j^{s,h})}[r_j(s, \mathbf{a}, b_j)]$.

Value functions. Given a joint policy $(\mathbf{x}, \mathbf{y})$, we define the value function for some team member $i \in \mathcal{N}$ (resp. for some adversary $j \in \mathcal{M}$) as a function from $[H] \times \mathcal{S}$ to $\mathbb{R}$: $\forall h \in [H], \forall s \in \mathcal{S}$,

$$V_{i,h}^{\mathbf{x},\mathbf{y}}(s) = \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\sum_{\tau=h}^{H} r_{team}(s_\tau, \mathbf{a}_\tau, \mathbf{b}_\tau) \middle| s_h = s, \mathbf{x}, \mathbf{y}\right]$$

$$V_{j,h}^{\mathbf{x},\mathbf{y}}(s) = \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\sum_{\tau=h}^{H} r_j(s_\tau, \mathbf{a}_\tau, \mathbf{b}_{j,\tau}) \middle| s_h = s, \mathbf{x}, \mathbf{y}, \right]$$

If the state at time step $h \in [H]$ is instead sampled from distribution $\rho \in \Delta(\mathcal{S})$, the *values* of policy $(\mathbf{x}, \mathbf{y})$ at step h are defined as $V_{j,h}^{\mathbf{x},\mathbf{y}}(\rho) = \mathbb{E}_{s \sim \rho}\left[V_{j,h}^{\mathbf{x},\mathbf{y}}(s)\right]$, $V_{i,h}^{\mathbf{x},\mathbf{y}}(\rho) = \mathbb{E}_{s \sim \rho}\left[V_{i,h}^{\mathbf{x},\mathbf{y}}(s)\right]$.

[1]Without loss of generality, we assume adversary rewards belong to $[0, 1]$.

NE-Gap. For a mixed joint policy $(\mathbf{x}, \mathbf{y})$, we define the Nash equilibrium gap (NE-GAP) as:

$$\text{NE-Gap}_k(\mathbf{x}, \mathbf{y}) = \max_{\mathbf{x}'_k} V_{k,1}^{\mathbf{x}_{-k}, \mathbf{x}'_k, \mathbf{y}}(\rho) - V_{k,1}^{\mathbf{x},\mathbf{y}}(\rho) \text{ if } k \in \mathcal{N}$$

$$\text{NE-Gap}_k(\mathbf{x}, \mathbf{y}) = \max_{\mathbf{y}'_k} V_{k,1}^{\mathbf{x}, \mathbf{y}_{-k}, \mathbf{y}'_k}(\rho) - V_{k,1}^{\mathbf{x},\mathbf{y}}(\rho) \text{ if } k \in \mathcal{M}$$

$$\text{NE-Gap}(\mathbf{x}, \mathbf{y}) = \max_{k \in \mathcal{N} \cup \mathcal{M}} \text{NE-Gap}_k(\mathbf{x}, \mathbf{y})$$

ε-NE. For any $\varepsilon \geq 0$, a joint mixed policy is an ε-NE of a MATMG Γ if and only if NE-Gap$(\mathbf{x}, \mathbf{y}) \leq \varepsilon$. Note that, by definition, $(\mathbf{x}, \mathbf{y})$ is a 0-NE if and only if it is a NE.

We can easily prove, by backward induction, that in finite-horizon MATMGs, value functions satisfy both a *team equal-value property (Property 1)* and an *adversarial zero-sum property (Property 2)*. The proofs are deferred to Appendix A, since they present no difficulty.

Proposition 1 (**Property 1: Team equal value).**
Given a finite horizon MATMG Γ and a joint policy $(\mathbf{x}, \mathbf{y})$, we have that for any time step $h \in [H]$:

$$V_{i,h}^{\mathbf{x},\mathbf{y}}(s) = V_{i',h}^{\mathbf{x},\mathbf{y}}(s) = V_{team,h}^{\mathbf{x},\mathbf{y}}(s), \forall i, i' \in \mathcal{N}, \forall s \in \mathcal{S}.$$

Proposition 2 (**Property 2: Adversarial zero-sum value).**
Given a finite horizon MATMG Γ and a joint policy $(\mathbf{x}, \mathbf{y})$, we have that for any time-step $h \in [H]$,

$$V_{team,h}^{\mathbf{x},\mathbf{y}}(s) = -\sum_{j \in \mathcal{M}} V_{j,h}^{\mathbf{x},\mathbf{y}}(s), \forall s \in \mathcal{S}.$$

The *Polynomial Expectation Properties* (PEP), formalised below in Assumptions 1 and 2 are useful when it comes to studying the complexity of various ε-NE computation problems in FH-MATMG. In particular, PEP avoid trivializing complexity results by avoiding to store rewards and transitions in space exponential in the number of players [Papadimitriou and Roughgarden, 2008].

Assumption 1 (PEP of Adversary Rewards)**.** *For any state $s \in \mathcal{S}$ and any (mixed) joint strategy profile $(\mathbf{x}, \mathbf{y}_j)$, $j \in \mathcal{M}$, there exists a polynomial-time oracle which outputs the expectation $r_j(s, \mathbf{x}, \mathbf{y}_j) = \mathbb{E}_{(\mathbf{a}, b_j) \sim (\mathbf{x}, \mathbf{y_j})} r_j(s, \mathbf{a}, b_j)$ in time* $\text{poly}(n, \sum_{i \in \mathcal{N}} |\mathcal{A}_i|, |\mathcal{B}_j|, |\mathbf{x}|, |\mathbf{y}_j|)$.

Assumption 2 (PEP of transitions)**.** *For any states $s, s' \in \mathcal{S}$ and any (mixed) joint strategy profile $(\mathbf{x}, \mathbf{y})$, there exists a polynomial-time oracle which outputs the expectation $\mathbb{P}(s'|s, \mathbf{x}, \mathbf{y}) = \mathbb{E}_{(\mathbf{a}, \mathbf{b}) \sim (\mathbf{x}, \mathbf{y})} \mathbb{P}(s'|s, \mathbf{a}, \mathbf{b})$ in time* $\text{poly}(n, m, \sum_{i \in \mathcal{N}} |\mathcal{A}_i|, \sum_{j \in \mathcal{M}} |\mathcal{B}_j|, |\mathbf{x}|, |\mathbf{y}|)$.

Under Assumptions 1 and 2, the FH-MATMG definition does not require storing reward or transition tables.

4 APPROXIMATE NE COMPUTATION IN FH-MATMGS

In this section, we study the complexity of computing an ε-NE in finite horizon MATMGs. We establish a positive result for the single-adversary case and then show that, when extending to $m \geq 2$ adversaries, the problem becomes PPAD-hard.

4.1 THE EASY CASE: FINITE-HORIZON ATMG

Theorem 1. *Let Γ be a Finite-Horizon (single adversary) ATMG and let $\varepsilon > 0$. Under Assumptions 1 and 2, there exists an algorithm that computes a (non-stationary) ε-NE in time polynomial in the natural parameters of Γ and in $\frac{1}{\varepsilon}$.*

The proof, deferred to Appendix B, is based on showing that the Nash-VI algorithm, when using the GradientDescentMax (GDM) procedure from Anagnostides et al. [2023] as a NE-Oracle, provides an efficient method for computing ε-NE in FH-ATMGs.

4.2 THE HARD CASE: PPAD-HARDNESS OF APPROXIMATE NE COMPUTATION IN FH-MATMG

We show the PPAD-hardness of computing an ε-NE of a FH-MATMG when there are at least two adversaries. This result is proved through a reduction from the PPAD-Complete problem BIMATRIX [Chen et al., 2009], the problem of finding an ϵ-NE in a bimatrix game.

Theorem 2. *Computing an ε-NE in a finite-horizon MATMG is PPAD-Hard, for $m \geq 2$ adversaries.*

Proof. We show that BIMATRIX reduces to FH-MATMG, thus implying that FH-MATMG is PPAD-Hard. Consider a bimatrix normal-form game $\Gamma^B = \langle (1,2), (\mathcal{A}_1^B, \mathcal{A}_2^B), (B_1, B_2) \rangle$ with pure strategy sets $\mathcal{A}_1^B$ and $\mathcal{A}_2^B$ for players 1 and 2 and rational payoff[2] matrices (B_1, B_2). Hence, $B_1, B_2 \in [0,1]^{|\mathcal{A}_1^B| \times |\mathcal{A}_2^B|}$.

The reduction constructs a 3-player FH-MATMG $\Gamma = \langle \mathcal{S}, \mathcal{N}, \mathcal{M}, \mathcal{A}, \mathcal{B}_1 \times \mathcal{B}_2, (r_j)_{j \in \mathcal{M}}, \mathbb{P}, H = 2, \rho \rangle$ as follows:

- $\mathcal{S} = \{s_1\} \bigcup_{(b_1,b_2) \in \mathcal{A}_1^B \times \mathcal{A}_2^B} \{s_{b_1,b_2}\}$. The state space is composed of an initial state s_1 plus one state for every pure joint strategy of Γ^B.
- The set of team members is composed of a single "dummy" team player, $\mathcal{N} = \{0\}$, with a single strategy, $\mathcal{A} = \{\perp\}$.
- The set of adversaries is $\mathcal{M} = \{1, 2\}$. Adversary $j \in \mathcal{M}$ has strategy set $\mathcal{B}_j = \mathcal{A}_j^B$, identical to that of the corresponding player in Γ^B.
- The rewards are defined as follows $\forall s \in \mathcal{S}, \forall b_j \in \mathcal{B}_j, \forall j \in \mathcal{M}$

$$r_j(s, \perp, b_j) = \begin{cases} 0 & s = s_1 \\ B_j(b_1, b_2) & s = s_{b_1,b_2} \end{cases} \quad (2)$$

- The game transitions deterministically from s_1 at time $h = 1$ to s_{b_1,b_2} at time $h = H = 2$ when joint strategy $(\perp, b_1, b_2)$ is chosen. The state remains unchanged if different from s_1 at $h = 1$. Thus, $\forall (b_1, b_2), (b_1', b_2') \in \mathcal{B}_1 \times \mathcal{B}_2$

$$\mathbb{P}(s' = s_{b_1,b_2} | s = s_1, \perp, b_1, b_2) = 1,$$
$$\mathbb{P}(s' = s_{b_1,b_2} | s = s_{b_1,b_2}, \perp, b_1', b_2') = 1 \quad (3)$$

- The game always starts in state s_1 ($\rho(s_1) = 1$).

The reduction is polynomial-time since it takes $O(|\mathcal{A}_1^B| \times |\mathcal{A}_2^B|)$ time to generate Γ (assuming constant time to read/write elements of B_1 and B_2)[3]. Now, consider any mixed strategy $\mathbf{y}^B = (\mathbf{y}_1^B, \mathbf{y}_2^B)$ of Γ^B and any policy $(\mathbf{x}, \mathbf{y}_1, \mathbf{y}_2)$ of Γ, such that $\mathbf{y}_j^{s_1,h=1}(b_j) = \mathbf{y}_j^B(b_j), \forall j \in \mathcal{M}, b_j \in \mathcal{B}_j$ (the other components of $\mathbf{x}$ and $\mathbf{y}$ are arbitrarily chosen). Then, one can check that in Γ, $\forall j \in \mathcal{M}$,

$$\begin{aligned} & V_{j,h=1}^{\mathbf{x},\mathbf{y}}(\rho) = V_{j,h=1}^{\mathbf{x},\mathbf{y}}(s_1) \\ &= \mathbb{E}_{\mathbf{x},\mathbf{y}} \left[\sum_{h=1}^{H} r_j(s^h, a^h, b^h) \mid s^1 = s_1 \right] \\ &= \mathbb{E}_{\mathbf{x},\mathbf{y}} \left[r_j(s_{b_1,b_2}, \perp, b_j) \mid s^1 = s_1 \right] \\ &= \sum_{(b_1,b_2) \in \mathcal{B}_1 \times \mathcal{B}_2} \mathbf{y}_1^{s_1,h=1}(b_1) \mathbf{y}_2^{s_1,h=1}(b_2) r_j(s_{b_1,b_2}, \perp, b_j) \\ &= \sum_{(b_1,b_2) \in \mathcal{B}_1 \times \mathcal{B}_2} \mathbf{y}_1^B(b_1) \mathbf{y}_2^B(b_2) B_j(b_1, b_2) = U_j^B(\mathbf{y}^B). \end{aligned}$$

Converting a mixed joint policy $(\mathbf{x}, \mathbf{y})$ of Γ back into a mixed strategy $\mathbf{y}^B$ of Γ^B is obviously polynomial: $\forall (b_1, b_2) \in \mathcal{B}_1 \times \mathcal{B}_2$, $(\mathbf{y}_1^B(b_1), \mathbf{y}_2^B(b_2)) = (\mathbf{y}_1^{s_1,h=1}(b_1), \mathbf{y}_2^{s_1,h=1}(b_2))$. Consequently, ε-NE of Γ^B can be obtained in polynomial-time from ε-NE of Γ. So, BIMATRIX reduces to FH-MATMG. Since BIMATRIX is PPAD-complete, it results that FH-MATMG is PPAD-hard. □

5 ADDITIVE TRANSITIONS FH-MATMG

Having established that approximating NE in finite-horizon MATMGs is PPAD-hard in the general case, we now identify

[2] Without loss of generality we consider rational payoffs bounded in $[0, 1]$.

[3] With a fixed number of players, PEPs are trivially satisfied by storing reward and transition tables in polynomial time.

a subclass of FH-MATMGs for which the problem becomes tractable, namely finite-horizon MATMGs with *additive transitions*. As discussed in the Related Work section, additive transitions—where the transition dynamics decompose additively according to the actions of the different controllers—have long been studied (dating back to [Raghavan et al., 1985, Thuijsman and Raghavan, 1997, Küenle, 1999]) as a non-trivial structural assumption that circumvents the computational intractability of approximating NE in Markov games. We now formalize the *Additive Transition* assumption in the context of finite-horizon MATMGs.

Definition 2 (Finite-horizon MATMG with additive transitions (AT-FH-MATMG)). *A FH-MATMG* Γ *defined by the tuple* $\Gamma = \langle \mathcal{S}, \mathcal{N}, \mathcal{M}, (\mathcal{A}_i)_{i\in\mathcal{N}}, (\mathcal{B}_j)_{j\in\mathcal{M}}, (r_j)_{j\in\mathcal{M}}, \mathbb{P}, H, \rho\rangle$ *as in Definition 1 has* additive transitions *if and only if the transition function* $\mathbb{P}$ *satisfies,* $\forall s, s' \in \mathcal{S}, (\mathbf{a}, \mathbf{b}) \in \mathcal{A} \times \mathcal{B}$,

$$\mathbb{P}(s'|s,a,b) = \sum_{i\in\mathcal{N}} \omega_{i,s}\mathbb{P}_i(s'|s,a_i) + \sum_{j\in\mathcal{M}} \omega_{j,s}\mathbb{P}_j(s'|s,b_j),$$

$$\text{with } \omega_{k,s} \geq 0, \quad \forall s \in \mathcal{S}, \forall k \in \mathcal{N}\cup\mathcal{M}$$
$$\sum_{i\in\mathcal{N}} \omega_{i,s} + \sum_{j\in\mathcal{M}} \omega_{j,s} = 1, \quad \forall s \in \mathcal{S}$$

For any subsets $\mathcal{N}' \subseteq \mathcal{N}$ and $\mathcal{M}' \subseteq \mathcal{M}$, we denote by $\mathbb{P}^{AT}_{\mathcal{N}'\cup\mathcal{M}'}$ the sum defining $\mathbb{P}$, restricted to the components in $\mathcal{N}' \cup \mathcal{M}'$. That is, for all $s, s' \in \mathcal{S}$, $\mathbf{a}_{\mathcal{N}'} \in \prod_{i\in\mathcal{N}'} \mathcal{A}_i$, and $\mathbf{b}_{\mathcal{M}'} \in \prod_{j\in\mathcal{M}'} \mathcal{B}_j$: $\mathbb{P}^{AT}_{\mathcal{N}'\cup\mathcal{M}'}(s' \mid s, \mathbf{a}_{\mathcal{N}'}, \mathbf{b}_{\mathcal{M}'}) = \sum_{i\in\mathcal{N}'} \omega_{i,s}\,\mathbb{P}_i(s' \mid s, a'_i) + \sum_{j\in\mathcal{M}'} \omega_{j,s}\,\mathbb{P}_j(s' \mid s, b'_j)$. Note that $\mathbb{P}^{AT}(\cdot|s,\mathbf{a},\mathbf{b})$ does not sum to one.

Observation 1 (Additive transitions satisfy the PEP)**.** *Under additive transitions, the expected transition induced by a mixed action profile decomposes additively into independent expectations, one per player, each taken over the corresponding individual strategy space, and hence it satisfies the PEP.*

5.1 NASH-VI FOR AT-FH-MATMG

This section presents Algorithm 1, a variant of Nash-VI that computes an ε-NE in FH-MATMGs with additive transitions. We also state our main theoretical result: Algorithm 1 computes an ε-NE with computational complexity polynomial in the natural parameters of the game and in $1/\varepsilon$. This result is formalised in Theorem 3 and proved in Section 5.2.

Algorithm 1 takes as input an AT-FH-MATMG and an approximation error $\varepsilon > 0$. The algorithm first initialises each adversary's value function to zero in every state at time step $H+1$ (line 1). Then, as in finite-horizon Nash-VI, for each state-time pair (s,h) it proceeds (backward) by solving (Line 5), a one-shot stage game whose payoffs include continuation values (constructed in line 4) with a precision $\varepsilon_h = \varepsilon/H$. Each iteration then computes the value function of each adversary $j \in \mathcal{M}$ with policies $(\mathbf{x}^{s,h}, (\mathbf{y}_j^{s,h}))_{j\in\mathcal{M}}$ for time step h and the value functions $(V_{j,h+1})_{j\in\mathcal{M}}$ for time step $h+1$ (Line 10). Note that the transition probabilities $p_{s\cdot}$ from a given state s under the current (mixed) policy profile used in Line 7 are *additive*, as is $\mathbb{P}$. Because the stage games for all states rely solely on value functions computed in subsequent time steps, the inner loop over states is parallelisable.

Also, note that in line 4 of Algorithm 1 the utility tables of $\tilde{\mathcal{G}}_{s,h}$ are not explicitly constructed. Indeed, the MATG-GDM algorithm defined in Maddila et al. [2026a] invoked in line 5 only requires polynomial time access (in particular, in $\sum_{i\in\mathcal{N}} |\mathcal{A}_i|$) to the values $\tilde{U}_j^{s,h}(\mathbf{x}, \mathbf{y}_j)$. Constructing the full tables $\tilde{U}_j^{s,h}(\cdot,\cdot)$ of $\tilde{\mathcal{G}}_{s,h}$ would take $\mathcal{O}(\prod_{i\in\mathcal{N}} |\mathcal{A}_i|)$ time and space, instead. Therefore, whenever an access to $\tilde{U}_j^{s,h}(\mathbf{x}, \mathbf{y}_j)$ is required in the MATG-GDM algorithm, it is queried by oracle EPO, described in Algorithm 2. In Section 5.2, we prove that, under the additive transition assumption and provided that the reward functions r_j satisfy the PEP (Assumption 1), each call to the EPO oracle can be executed in polynomial time.

Furthermore, in Section 5.2, we justify this particular construction of the stage games, showing that they define a valid (single-step) MATG at each (state, step) and establishing the correctness of the resulting algorithm.

Algorithm 1: Variant of Nash-VI for AT-FH-MATMG

Input: AT-FH-MATMG Γ, precision parameter ε
Output: $(\mathbf{x}^{s,h}, (\mathbf{y}_j^{s,h})_{j\in\mathcal{M}})_{s\in\mathcal{S}, h\in[H]}$

1 **Initialize:** $V_{j,H+1}(s) \leftarrow 0$ for all $j \in \mathcal{M}$ and $s \in \mathcal{S}$;
2 **for** $h = H \to 1$ **do**
3 **for** $s \in \mathcal{S}$ **do**
4 $\tilde{\mathcal{G}}_{s,h} \leftarrow$ $\langle \mathcal{N}, \mathcal{M}, (\mathcal{A}_i)_{i\in\mathcal{N}}, (\mathcal{B}_j)_{j\in\mathcal{M}}, \text{EPO}(s,h,V_{\cdot,h+1})\rangle$;
  `// Solve MATG` $\tilde{\mathcal{G}}_{s,h}$ `to precision` $\frac{\varepsilon}{H}$
5 $(\mathbf{x}^{s,h}, \mathbf{y}^{s,h}) \leftarrow \text{MATG-GDM}(\tilde{\mathcal{G}}_{s,h}, \varepsilon/H)$;
  `// Compute transitions from` s
6 **for** $s' \in \mathcal{S}$ **do**
7 $p_{ss'} \leftarrow \sum_{i\in\mathcal{N}} \sum_{a_i\in\mathcal{A}_i} \mathbf{x}_i^{s,h}(a_i)\omega_{i,s}\mathbb{P}_i(s'|s,a_i) + \sum_{j\in\mathcal{M}} \sum_{b_j\in\mathcal{B}_j} \mathbf{y}_j^{s,h}(b_j)\omega_{j,s}\mathbb{P}_j(s'|s,b_j)$;
8 **end**
  `// Update value functions`
9 **for** $j \in \mathcal{M}$ **do**
10 $V_{j,h}(s) \leftarrow r_j(s, \mathbf{x}^{s,h}, \mathbf{y}_j^{s,h}) + \sum_{s'\in\mathcal{S}} p_{ss'} V_{j,h+1}(s')$;
11 **end**
12 **end**
13 **end**

We are now ready to formulate our main theoretical result:

Algorithm 2: ExpectedPayoffOracle (EPO)

Input: State s, time h, continuation values $\mathbf{v}$,
Output: Functions $(\tilde{U}_j^{s,h})_{j\in\mathcal{M}}$

for $j \in \mathcal{M}$ **do**
  Define $\tilde{U}_j^{s,h} : \tilde{\mathcal{X}} \times \tilde{\mathcal{Y}}_j \to \mathbb{R}$:
  $\tilde{\mathbf{x}}, \tilde{\mathbf{y}}_j \to$
  $r_j(s, \tilde{\mathbf{x}}, \tilde{\mathbf{y}}_j) + \sum_{s'\in\mathcal{S}} \omega_{j,s} \sum_{b_j\in\mathcal{B}_j} \mathbb{P}_j(s'|s, \tilde{\mathbf{y}}_j(b_j))\mathbf{v}(s') +$
  $\sum_{s'\in\mathcal{S}} \sum_{i\in\mathcal{N}} \omega_{i,s} \sum_{a_i\in\mathcal{A}_i} \mathbb{P}_i(s'|s, \tilde{\mathbf{x}}_i(a_i))\mathbf{v}(s')$
end

Theorem 3. *Let Γ be a finite-horizon MATMG with additive transitions and let $\varepsilon > 0$. Under Assumption 1 (PEP of Adversary Rewards), Algorithm 1 computes a non stationary ε-NE for Γ in time polynomial in $\frac{1}{\varepsilon}$ and in the natural parameters of Γ.*

5.2 PROOF OF THE MAIN THEOREM

We first prove that Algorithm 1 is correct, that is, computes an ε-NE of AT-FH-MATMG Γ. Then, we prove that Algorithm 1 runs in time polynomial in the natural parameters of Γ and in $\frac{1}{\varepsilon}$.

Correctness of Algorithm 1

In Nash-VI, let $\mathcal{G}_{s,h}(V_{\cdot,h+1})$ be the stage game constructed from any finite-horizon Markov game Γ at state–time pair (s,h), given the value vector $V_{\cdot,h+1}$ computed in an earlier step, according to Definition 3.

Definition 3 (Stage game $\mathcal{G}_{s,h}(\mathbf{v})$)**.** *Given a AT-FH-MATMG Γ, the stage game for the (s,h) state-time pair, $\mathcal{G}_{s,h}(\mathbf{v})$, is defined for arbitrary fixed continuation value functions $\mathbf{v} = ((\mathbf{v}_{i,h+1})_{i\in\mathcal{N}}, (\mathbf{v}_{j,h+1})_{j\in\mathcal{M}})$, where the $\mathbf{v}_{i,h+1}$ and $\mathbf{v}_{j,h+1}$ are functions from $\mathcal{S}$ to $\mathbb{R}$, as the normal form game:*

$$\mathcal{G}_{s,h}(\mathbf{v}) = \langle \mathcal{N}\cup\mathcal{M}, ((\mathcal{A}_i)_{i\in\mathcal{N}}, (\mathcal{B}_j)_{j\in\mathcal{M}}), (U_k^{s,h})_{k\in\mathcal{N}\cup\mathcal{M}}\rangle$$

with the payoff function of each adversary $j \in \mathcal{M}$ defined as:

$$U_j^{s,h}(\mathbf{a},\mathbf{b}) = r_j(s,\mathbf{a},b_j) + \sum_{s'\in\mathcal{S}} \mathbb{P}^{AT}_{\mathcal{N}\cup\mathcal{M}}(s'|s,\mathbf{a},\mathbf{b})\mathbf{v}_{j,h+1}(s')$$

and the payoff function of each team member $i \in \mathcal{N}$ defined as:

$$U_i^{s,h}(\mathbf{a},\mathbf{b}) = r_{team}(s,\mathbf{a},\mathbf{b}) + \sum_{s'\in\mathcal{S}} \mathbb{P}^{AT}_{\mathcal{N}\cup\mathcal{M}}(s'|s,\mathbf{a},\mathbf{b})\mathbf{v}_{i,h+1}(s')$$

It is a standard result that, for any general transition function $\mathbb{P}$, not necessarily additive, if $\mathcal{G}_{s,h}(V_{\cdot,h+1})$ can be solved (for every step and state) up to an ε_h-NE, then the resulting backward induction procedure produces a non-stationary Markov policy profile that constitutes a $(\sum_{h\in[H]}\varepsilon_h)$-NE of the finite-horizon Markov game Γ.

Remark 2. *Notice that there is an important difference between Algorithm 1 and Nash-VI. In particular, at each state–time pair, Algorithm 1 solves a transformed normal-form game $\tilde{\mathcal{G}}_{s,h}(V_{\cdot,h+1})$ rather than $\mathcal{G}_{s,h}(V_{\cdot,h+1})$. This modification is necessary because, for any state–time pair, the corresponding stage game $\mathcal{G}_{s,h}(V_{\cdot,h+1})$ does not preserve the multi-adversarial independent structure (see Observation 2) and thus cannot be efficiently solved using the NE oracle for MATGs. In other words, without this transformation, Nash-VI requires to solve a general normal-form game at each state–time pair.*

Observation 2. *The stage game $\mathcal{G}_{s,h}(V_{\cdot,h+1})$ is not an MATG. The payoff function of each adversary $j \in \mathcal{M}$, $U_j^{s,h}(\mathbf{a},\mathbf{b})$, depends on the joint strategy profile of adversaries, $\mathbf{b}$, due to the transition function.*

Given this fact, we establish the correctness of Algorithm 1 by proving that at each state–time iteration:

1. The transformed game $\tilde{\mathcal{G}}_{s,h}(V_{\cdot,h+1})$ is a MATG (Observation 3).
2. Any ε_h-NE of MATG $\tilde{\mathcal{G}}_{s,h}(V_{\cdot,h+1})$ is also an ε_h-NE of the original stage game $\mathcal{G}_{s,h}(V_{\cdot,h+1})$ (Prop. 3);

Since the stage games $\mathcal{G}_{s,h}(V_{\cdot,h+1})$ are not MATGs, and inspired by a result of Tewolde and Conitzer [2024] (see Lemma 4.3 therein), we next introduce a class of transformations of normal-form games that preserve players' best responses and, consequently, the NE of the games:

Lemma 1 (ε-NE preserving transformation in NFG)**.** *Given a NFG $\mathcal{G} = \langle \mathcal{N}, (\mathcal{A}_i)_{i\in\mathcal{N}}, (U_i)_{i\in\mathcal{N}}\rangle$ and a set of real-valued functions $\{\psi_i : \mathcal{A}_{-i} \to \mathbb{R}\}_{i\in\mathcal{N}}$, we define the transformed NFG $\tilde{\mathcal{G}} = \langle \mathcal{N}, (\mathcal{A}_i)_{i\in\mathcal{N}}, (\tilde{U}_i)_{i\in\mathcal{N}}\rangle$, where*

$$\tilde{U}_i(\mathbf{a}) = U_i(\mathbf{a}) + \psi_i(\mathbf{a_{-i}}), \forall i \in \mathcal{N}, \forall \mathbf{a} \in \mathcal{A}.$$

Then, for any $\varepsilon \geq 0$, $\mathbf{x}$ is a ε-NE of $\mathcal{G}$ if and only if it is a ε-NE of $\tilde{\mathcal{G}}$.

Proof. The following equality of differences between joint strategy utilities holds $\forall \mathbf{a} \in \mathcal{A}, a_i' \in \mathcal{A}_i$:

$$\begin{aligned}&\tilde{U}_i(\mathbf{a}) - \tilde{U}_i(a_i', \mathbf{a_{-i}})\\ &= (U_i(\mathbf{a}) + \psi_i(\mathbf{a_{-i}})) - (U_i(a_i', \mathbf{a_{-i}}) + \psi_i(\mathbf{a_{-i}}))\\ &= U_i(\mathbf{a}) - U_i(a_i', \mathbf{a_{-i}})\end{aligned}$$

As a consequence, for any joint mixed strategy $\mathbf{x}$, for any player i and pure strategy a_i', by linearity of expectations, we have:

$$\left(\tilde{U}_i(\mathbf{x}) - \tilde{U}_i(a_i', \mathbf{x}_{-i})\right) = \left(U_i(\mathbf{x}) - U_i(a_i', \mathbf{x}_{-i})\right). \quad (5)$$

Thus, $\mathbf{x}$ is an ε-NE of $\mathcal{G}$ if and only if it is a ε-NE of $\tilde{\mathcal{G}}$. □

We now formally define the stage games used in Algorithm 1:

Definition 4 (Transformed stage games)**.** *For any stage game $\mathcal{G}_{s,h}(\mathbf{v})$, let us define the transformed stage game*

$$\tilde{\mathcal{G}}_{s,h}(\mathbf{v}) = \langle \mathcal{N} \cup \mathcal{M}, ((\mathcal{A}_i)_{i\in\mathcal{N}}, (\mathcal{B}_j)_{j\in\mathcal{M}}), (\tilde{U}_l^{s,h})_{l\in\mathcal{N}\cup\mathcal{M}} \rangle$$

with same players and strategies sets as $\mathcal{G}_{s,h}(\mathbf{v})$, but modified utility functions, defined as follows: $\forall j \in \mathcal{M}, \forall i \in \mathcal{N}$,

$$\tilde{U}_j^{s,h}(\mathbf{a}, b_j) = r_j(s, \mathbf{a}, b_j) + \sum_{s'\in\mathcal{S}} \mathbb{P}^{AT}_{\mathcal{N}\cup\{j\}}(s'|s, \mathbf{a}, b_j)\mathbf{v}_{j,h+1}(s')$$
$$\tilde{U}_i^{s,h}(\mathbf{a}, \mathbf{b}) = \sum_{j\in\mathcal{M}} -\tilde{U}_j^{s,h}(\mathbf{a}, b_j)$$

Observation 3. *$\tilde{\mathcal{G}}_{s,h}(\mathbf{v})$ is an MATG since (1) each adversary's payoff is independent of the other adversaries' strategies, (2) team members share identical utility tables and (3) the game is zero-sum.*

Now that we have defined a transformation from stage games to MATGs, it remains to prove that ε-NE are preserved in the transformation.

Proposition 3 (ε-NE preservation)**.** *Let Γ be an AT-FH-MATMG and $\varepsilon \geq 0$. For any continuation values $\mathbf{v}$ satisfying Properties 1 and 2, and any state–time pair (s, h), games $\mathcal{G}_{s,h}(\mathbf{v})$ and $\tilde{\mathcal{G}}_{s,h}(\mathbf{v})$ have the same set of ε-NE.*

Proof. Consider:

$$\psi_j(\mathbf{b}_{-j}) =_{def} -\sum_{s'\in\mathcal{S}} \mathbb{P}^{AT}_{\mathcal{M}-\{j\}}(s'|s, \mathbf{b}_{-j})\mathbf{v}_{j,h+1}(s'),$$
$$\psi_i(\mathbf{b}) = -\sum_{j\in\mathcal{M}} \psi_j(\mathbf{b}_{-j}).$$

Let us observe that for any (s, h)-pair:

$$\begin{aligned} &U_j^{s,h}(\mathbf{a}, \mathbf{b}) + \psi_j(\mathbf{b}_{-j}) \\ &= r_j(s, \mathbf{a}, b_j) + \sum_{s'\in\mathcal{S}} \mathbb{P}^{AT}_{\mathcal{N}\cup\{j\}}(s'|s, \mathbf{a}, b_j)\mathbf{v}_{j,h+1}(s') \\ &= \tilde{U}_j(\mathbf{a}, b_j) \end{aligned}$$

and

$$\begin{aligned} &U_i^{s,h}(\mathbf{a}, \mathbf{b}) + \psi_i(\mathbf{b}) \\ &= r_{team}(s, \mathbf{a}, \mathbf{b}) + \sum_{s'\in\mathcal{S}} \mathbb{P}^{AT}_{\mathcal{N}\cup\mathcal{M}}(s'|s, \mathbf{a}, \mathbf{b})\mathbf{v}_{i,h+1}(s') \\ &\quad + \sum_{j\in\mathcal{M}} \sum_{s'\in\mathcal{S}} \mathbb{P}^{AT}_{\mathcal{M}-\{j\}}(s'|s, \mathbf{b}_{-j})\mathbf{v}_{j,h+1}(s') \\ &\underset{\text{By Prop. 1, 2}}{=} r_{team}(s, \mathbf{a}, \mathbf{b}) - \sum_{s'\in\mathcal{S}} \mathbb{P}^{AT}_{\mathcal{N}}(s'|s, \mathbf{a}) \sum_{j\in\mathcal{M}} \mathbf{v}_{j,h+1}(s') \\ &\quad - \sum_{s'\in\mathcal{S}} \mathbb{P}^{AT}_{\mathcal{M}}(s'|s, \mathbf{b}) \sum_{j\in\mathcal{M}} \mathbf{v}_{j,h+1}(s') \\ &\quad + \sum_{j\in\mathcal{M}} \sum_{s'\in\mathcal{S}} \mathbb{P}^{AT}_{\mathcal{M}-\{j\}}(s'|s, \mathbf{b}_{-j})\mathbf{v}_{j,h+1}(s') \\ &= r_{team}(s, \mathbf{a}, \mathbf{b}) - \sum_{s'\in\mathcal{S}} \mathbb{P}^{AT}_{\mathcal{N}}(s'|s, \mathbf{a}) \sum_{j\in\mathcal{M}} \mathbf{v}_{j,h+1}(s') \\ &\quad - \sum_{s'\in\mathcal{S}} \sum_{j\in\mathcal{M}} \mathbb{P}^{AT}_{j}(s'|s, b_j)\mathbf{v}_{j,h+1}(s') \\ &= -\sum_{j\in\mathcal{M}} \left(r_j(s, \mathbf{a}, b_j) + \sum_{s'\in\mathcal{S}} \mathbb{P}^{AT}_{\mathcal{N}\cup\{j\}}(s'|s, \mathbf{a}, b_j)\mathbf{v}_{j,h+1}(s') \right) \\ &= -\sum_{j\in\mathcal{M}} \tilde{U}_j^{s,h}(\mathbf{a}, b_j) = \tilde{U}_i^{s,h}(\mathbf{a}, \mathbf{b}). \end{aligned}$$

So, we can apply Lemma 1 to $\mathcal{G}_{s,h}(\mathbf{v})$ to show that it has the same ε-NE as $\tilde{\mathcal{G}}_{s,h}(\mathbf{v})$ and we are done. □

Correctness of Theorem 3 At each state–time iteration (s, h) of Algorithm 1, the policy $\{\mathbf{x}^{s,h}, \mathbf{y}^{s,h}\}$ is guaranteed to be a ε/H-NE of MATG $\tilde{\mathcal{G}}_{s,h}(V_{\cdot,h+1})$ since the MATG-GDM algorithm is guaranteed for any $\varepsilon_h > 0$ to return a ε_h-NE of a MATG.

Now, from Proposition 3, $\{\mathbf{x}^{s,h}, \mathbf{y}^{s,h}\}$ is also a ε_h-NE of $\mathcal{G}_{s,h}(V_{\cdot,h+1})$. So, from the Nash-VI property, the full joint mixed policy $(\mathbf{x}, \mathbf{y})$ is a $\sum_{h\in[H]} \varepsilon_h$-NE of Γ, that is a ε-NE (since $\sum_{h\in[H]} \varepsilon_h = \sum_{h\in[H]} \varepsilon/H = \varepsilon$).

Algorithm 1 takes polynomial time to run

Note that Algorithm 1 executes $H \times S$ inner iterations, after the obviously polytime initialization. At each state–time pair (s, h), it invokes the MATG-GDM algorithm [Maddila et al., 2026a] as a NE oracle (Line 5) to compute an ε/H-approximate NE of the stage game $\tilde{\mathcal{G}}_{s,h}(V_{\cdot,h+1})$. The construction of $\tilde{\mathcal{G}}_{s,h}(V_{\cdot,h+1})$ (Line 4) builds the expected payoff functions of adversaries, $(\tilde{U}_j^{s,h})_{j\in\mathcal{M}}$, by calling the ExpectedPayoffOracles (EPO) procedure (Algorithm 2). We observe in Algorithm 2 that for any mixed strategy profile $(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}_j) \in \tilde{\mathcal{X}} \times \tilde{\mathcal{Y}}_j$, the (on demand) computation of the corresponding expected payoff $\tilde{U}_j^{s,h}(\tilde{\mathbf{x}}, \tilde{\mathbf{y}}_j) = \mathbb{E}_{(\mathbf{a},\mathbf{b}_j)\sim(\tilde{\mathbf{x}},\tilde{\mathbf{y}}_j)}\tilde{U}_j(\mathbf{a}, \mathbf{b}_j)$ satisfy the PEP. This is because: (i) the computation of $r_j(s, \tilde{\mathbf{x}}, \tilde{\mathbf{y}}_j)$ takes polynomial time (Assumption 1) and (ii) under Additive Transitions, the computation of $\mathbb{P}^{AT}_{\mathcal{N}\cup\{j\}}(s'|s, \mathbf{a}, b_j)$ decomposes across agents and

can therefore be computed in $\mathcal{O}(|\mathcal{S}| \cdot (|\mathcal{B}_j| + \sum_{i \in \mathcal{N}} |\mathcal{A}_i|))$. Now, since $\tilde{\mathcal{G}}_{s,h}(V_{\cdot,h+1})$ is a MATG (Observation 3) and the payoff functions of adversaries satisfy the PEP, MATG-GDM is guaranteed to return an ε_h-NE of any MATG, with $\varepsilon_h = \varepsilon/H$, in time polynomial in the natural parameters of $\tilde{\mathcal{G}}_{s,h}(V_{\cdot,h+1})$ and in $1/\varepsilon_h = H/\varepsilon$. The running time of each iteration is dominated by this call to MATG-GDM (Line 5). Indeed, the remaining two subloops are polynomial-time: the computation of the transition probabilities $p_{ss'}$ (Line 7) is polynomial thanks to the additive transition structure, while the value function update (Line 10) is polynomial because the expected reward functions satisfy the PEP (Assumption 1). Therefore, it follows that Algorithm 1 itself runs in time polynomial in the natural[4] parameters of Γ and in $1/\varepsilon$.

Remark 3. *Requiring the adversaries' reward functions to satisfy the PEP (Assumption 1) is crucial for proving the polynomial-time complexity of the algorithm. Such an assumption is standard in algorithmic game theory [Papadimitriou and Roughgarden, 2008] and holds for most succinctly representable game classes.*

6 EXPERIMENTAL EVALUATION

We benchmark our implementation of Algorithm 1 on randomly generated instances of AT-FH-MATMGs with uniformly sampled rewards, transition kernels, and transition weights[5]. Additional experiments are deferred to Appendix C. We provide a model of the Fishery Game, modelled as a structured AT-FH-MATMG, in Appendix D and illustrate the performance of Algorithm 1 on these games. We also illustrate the performance of Nash-VI (Algorithm 1) using a generic Nash Equilibrium oracle instead of MATG-GDM in Appendix E.

Each configuration corresponds to an AT-FH-MATMG with n team agents, m adversaries, a actions per agent, S states, and horizon H, solved up to a target precision ε ($\varepsilon = 0.1$ unless otherwise specified).

The implementation[6] is in Python 3.11 using JAX [Bradbury et al., 2018], enabling efficient GPU acceleration. We also use the implementation of MATG-GDM provided by Maddila et al. [2026a] as the NE-Oracle to solve stage games. All experiments were run on an NVIDIA 2080Ti GPU.

For each configuration, performance is evaluated by the NE-Gap of the returned solution (i.e., the effective precision achieved) and the Runtime. Each instance was solved independently on the GPU to accurately measure runtime. Reported results correspond to the mean and standard deviation computed over 10 instances of the same configuration.

We assess the scalability of the algorithm by varying the number of adversaries, $m \in \{1, 3, 6, 9\}$, as well as the horizon, the number of states, and the target precision.

Scalability w.r.t Horizon (H). Figure 1 reports the results obtained by varying the horizon $H \in \{2, 4, \ldots, 20\}$ for instances with $n = 4$ team agents, $a = 6$ actions per agent, and $S = 10$ states. Figure 1a shows that the NE-Gap of the calculated solutions is consistently smaller than the theoretical guarantee ($\varepsilon = 0.1$), even as the number of adversaries changes. Figure 1b corroborates the theoretical polynomial dependency on the horizon of algorithm runtime. For example, for $m = 1$, instances with $H = 2$ are solved in $\approx$ 1s, with $H = 10$ in $\approx$ 5min, and with $H = 20$ in $\approx$ 20min. Interestingly, instances with more adversaries are solved faster – for $m = 9$, instances with $H = 10$ take $\approx$ 1min and $\approx$ 10min when $H = 20$. This mirrors Maddila et al. [2026a]'s empirical results where increasing the number of adversaries in MATGs decreased MATG-GDM's runtime. The authors hypothesised that this was due to larger gradient updates applied to the team policy $\mathbf{x}^{s,h}$ as the number of adversaries increases.

Scalability w.r.t. Number of States (S). In Figure 2, we varied the number of states $S \in \{10, 20, \ldots, 50\}$ for instances with $n = 4$ team agents, $a = 6$ actions per agent, and horizon $H = 10$. In Figure 2a, as for Figure 1a, the empirical NE-Gap of the returned solutions is always smaller than the theoretical guarantee. We also observe that the NE-Gap is largely unaffected by the increase in the number of states. Figure 2b shows that the number of states has negligible impact on the runtime, compared with Figure 1b. For example, for $m = 1$, instances with $S = 10$ take $\approx$ 5min to solve, compared to $\approx$ 8min when $S = 50$. This is because in Algorithm 1's implementation, each (transformed) stage game $\tilde{\mathcal{G}}_{s,h}$ is solved in parallel by leveraging the GPU. Finally, as also observed when varying the horizon, instances with a larger number of adversaries (m) are solved faster.

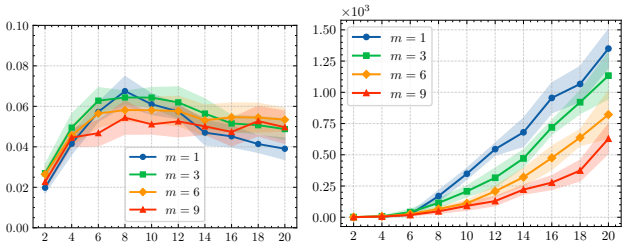
Scalability w.r.t Precision (ε). Figure 3 reports the results obtained by varying the required precision $\varepsilon \in \{0.1, 0.05, 0.01, 0.005\}$ for instances[7] with $n = 4$ team agents, $a = 6$ actions per agent, and horizon and number of states $H = S = 10$. Figure 3a shows that the empirical NE-Gap of the returned solutions is always smaller than the required precision ε. Figure 3b shows that Algorithm 1's runtime scales favourably with the precision ε. For instance, when $m = 1$, solving games with precision $\varepsilon = 0.1$ requires $\sim$ 100s, whereas targeting a precision 20 times smaller ($\varepsilon = 0.005$) only increases runtime by 6x ($\sim$ 600s).

[4] H belongs to the natural parameters of Γ.

[5] Transition weights are normalized to ensure that Def. 2 holds.

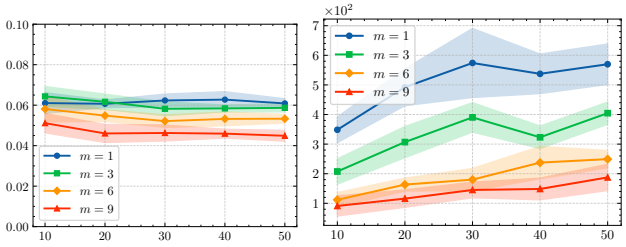
[6] Code is provided as Supplementary Material at `https://forge.inrae.fr/chip-gt/fhat_matmg`. Pinned version provided at (Maddila et al. [2026b]).

[7] We solve the same 10 instances varying only ε.



(a) Horizon vs. NE-Gap (b) Horizon vs. Runtime (in s)

Figure 1: Results for horizon $H \in \{2, 4, \ldots, 20\}$ in games with $n = 4, a = 6, S = 10$ and precision $\varepsilon = 0.1$.



(a) S vs. NE-Gap (b) S vs. Runtime (in s)

Figure 2: Results for number of states $S \in \{10, 20, \ldots, 50\}$ in games with $n = 4, a = 6, H = 10$ and precision $\varepsilon = 0.1$.

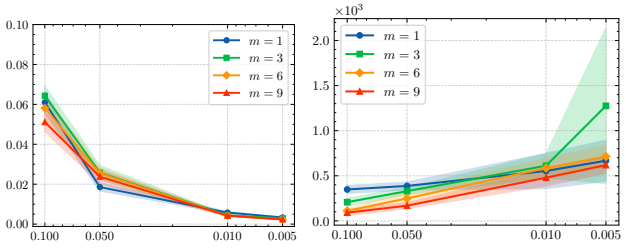
7 CONCLUDING REMARKS

In this article, we explored the boundary between tractable and intractable problems in stochastic games approximate-NE computation. More precisely, in the context of FH-MATMG, we proved that the problem was intractable in the general case, but became tractable when there was a single adversary, or when transitions were *additive* (in AT-FH-MATMGs).

We highlight that an open question for future research is whether additive transitions are a necessary condition for tractability, or whether weaker structural assumptions could be enough to guarantee tractability in FH-MATMGs.

Another natural direction for future research is to investigate whether our results on computing non-stationary approximate NE in finite-horizon MATMGs with additive transitions can be extended to the computation of stationary NE in their infinite-horizon counterparts. We note that existing results for infinite-horizon Adversarial Team Markov games are restricted to the single-adversary setting [Kalogiannis et al., 2023, 2024], whereas the independent multi-adversary setting remains largely unexplored.

Finally, another natural extension of this work is to consider settings in which the game dynamics are accessible only through simulation. This learning setting has already been investigated in Kalogiannis et al. [2024] who proposed a learning algorithm for computing approximate stationary NE in single-adversary infinite-horizon ATMGs. Alternatively, one could investigate extensions of the popular Policy Space Response Oracles (PSRO) framework [Lanctot et al., 2017], which has recently been successfully applied to approximate TMECor equilibria in large two-team adversarial games [McAleer et al., 2023]. Adapting these approaches to compute approximate NE in the independent multi-adversarial finite-horizon setting considered in this paper constitutes an interesting avenue for future research.



(a) (ε) vs. Nash-Gap (b) (ε) vs. Runtime (in s)

Figure 3: Results for precision $\varepsilon \in \{0.1, 0.05, 0.01, 0.005\}$ in games with $n = 4, a = 6, H = S = 10$.

Acknowledgements

This work was funded by the French National Research Agency (ANR) under grant ANR-22-CE92-0011-01.

References

Ioannis Anagnostides, Ioannis Panageas, Gabriele Farina, and Tuomas Sandholm. On last-iterate convergence beyond zero-sum games. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato, editors, *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pages 536–581. PMLR, 2022. URL https://proceedings.mlr.press/v162/anagnostides22a.html.

Ioannis Anagnostides, Fivos Kalogiannis, Ioannis Panageas, Emmanouil-Vasileios Vlatakis-Gkaragkounis, and Stephen Mcaleer. Algorithms and complexity for computing nash equilibria in adversarial team games. In *Proceedings of the 24th ACM Conference on Economics and Computation*, EC '23, page 89, New York, NY, USA, 2023. Association for Computing Machinery. ISBN 9798400701047. doi: 10.1145/3580507.3597776. URL https://doi.org/10.1145/3580507.3597776.

James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye

Wanderman-Milne, and Qiao Zhang. JAX: composable transformations of Python+NumPy programs, 2018. URL http://github.com/jax-ml/jax.

Yang Cai, Ozan Candogan, Constantinos Daskalakis, and Christos Papadimitriou. Zero-sum polymatrix games: A generalization of minmax. *Mathematics of Operations Research*, 41(2):648–655, 2016.

Xi Chen, Xiaotie Deng, and Shang-Hua Teng. Settling the complexity of computing two-player nash equilibria. *Journal of the ACM (JACM)*, 56(3):1–57, 2009.

Constantinos Daskalakis, Paul W. Goldberg, and Christos H. Papadimitriou. The complexity of computing a nash equilibrium. *SIAM Journal on Computing*, 39(1):195–259, 2009. doi: 10.1137/070699652. URL https://doi.org/10.1137/070699652.

Constantinos Daskalakis, Noah Golowich, and Kaiqing Zhang. The complexity of markov equilibrium in stochastic games. In Gergely Neu and Lorenzo Rosasco, editors, *Proceedings of Thirty Sixth Conference on Learning Theory*, volume 195 of *Proceedings of Machine Learning Research*, pages 4180–4234. PMLR, 12–15 Jul 2023. URL https://proceedings.mlr.press/v195/daskalakis23a.html.

AbdelRahman Eldosouky, Walid Saad, and Dusit Niyato. Single controller stochastic games for optimized moving target defense. In *2016 IEEE International Conference on Communications (ICC)*, pages 1–6. IEEE, 2016.

Jerzy Filar and Koos Vrieze. *Competitive Markov decision processes*. Springer Science & Business Media, 2012.

Srihari Govindan and Robert Wilson. Computing nash equilibria by iterated polymatrix approximation. *Journal of Economic Dynamics and Control*, 28(7):1229–1241, 2004.

Erik Halvorson, Vincent Conitzer, and Ronald Parr. Multi-step multi-sensor hider-seeker games. In Craig Boutilier, editor, *IJCAI 2009, Proceedings of the 21st International Joint Conference on Artificial Intelligence, Pasadena, California, USA, July 11-17, 2009*, pages 159–166, 2009. URL http://ijcai.org/Proceedings/09/Papers/037.pdf.

Anna Jaśkiewicz and Andrzej S. Nowak. On markov perfect equilibria in discounted stochastic arat games. *SIAM Journal on Control and Optimization*, 62(4):2148–2175, 2024. doi: 10.1137/23M1592365. URL https://doi.org/10.1137/23M1592365.

Fivos Kalogiannis and Ioannis Panageas. Zero-sum polymatrix markov games: Equilibrium collapse and efficient computation of nash equilibria. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/bcdcd565f83a8a6681a8269d325a5304-Abstract-Conference.html.

Fivos Kalogiannis and Ioannis Panageas. Nash equilibria in reward-potential markov games: Algorithms, complexity, and applications, 2024. URL https://openreview.net/forum?id=r0kY4SS7ts.

Fivos Kalogiannis, Ioannis Anagnostides, Ioannis Panageas, Emmanouil V. Vlatakis-Gkaragkounis, Vaggos Chatziafratis, and Stelios Andrew Stavroulakis. Efficiently computing nash equilibria in adversarial team markov games. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL https://openreview.net/forum?id=mjzm6btqgV.

Fivos Kalogiannis, Jingming Yan, and Ioannis Panageas. Learning equilibria in adversarial team markov games: A nonconvex-hidden-concave min-max optimization problem. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang, editors, *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024. URL http://papers.nips.cc/paper_files/paper/2024/hash/a8bc668b1559d705221b0e7510c45e48-Abstract-Conference.html.

Michael J. Kearns, Yishay Mansour, and Satinder Singh. Fast planning in stochastic games. In Craig Boutilier and Moisés Goldszmidt, editors, *UAI '00: Proceedings of the 16th Conference in Uncertainty in Artificial Intelligence, Stanford University, Stanford, California, USA, June 30 - July 3, 2000*, pages 309–316. Morgan Kaufmann, 2000. URL https://dslpitt.org/uai/displayArticleDetails.jsp?mmnu=1&smnu=2&article_id=37&proceeding_id=16.

Heinz-Uwe Küenle. Equilibrium strategies in stochastic games with additive cost and transition structure. *International Game Theory Review (IGTR)*, 1(02):131–147, None 1999. doi: 10.1142/S0219198999000098. URL https://ideas.repec.org/a/wsi/igtrxx/v01y1999i02ns0219198999000098.html.

Wai Yee Lam, Chee-Chean Phung, Zainal Abidin Mat, Hamidi Jamaluddin, Charina Pria Sivayogam, Fauzul Azim Zainal Abidin, Azlan Sulaiman, Melynda Ka Yi Cheok, Noor Alif Wira Osama, Salman Sabaan, Abdul Kadir Abu Hashim, Mark Daniel Booton, Abishek

Harihar, Gopalasamy Reuben Clements, and Rob Stuart Alexander Pickles. Using a crime prevention framework to evaluate tiger counter-poaching in a southeast asian rainforest. *Frontiers in Conservation Science*, 4, 2023. ISSN 2673-611X. doi: 10.3389/fcosc.2023.1213552. URL https://www.frontiersin.org/articles/10.3389/fcosc.2023.1213552.

Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Perolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3323fe11e9595c09af38fe67567a9394-Paper.pdf.

Prasanna Maddila, Casellas Eric, Patrick Chabrier, Régis Sabbadin, and Meritxell Vinyals. APE: An anti-poaching multi-agent reinforcement learning benchmark. In *Seventeenth European Workshop on Reinforcement Learning*, 2024. URL https://openreview.net/forum?id=JSWRnHC93W.

Prasanna Maddila, Régis Sabbadin, and Meritxell Vinyals. Efficiently computing approximate nash equilibria in multi-adversarial team games. In *Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems*, AAMAS '26, page 938–946, Richland, SC, 2026a. International Foundation for Autonomous Agents and Multiagent Systems. ISBN 9798400723179. doi: 10.65109/TBAX6220. URL https://doi.org/10.65109/TBAX6220.

Prasanna Maddila, Régis Sabbadin, and Meritxell Vinyals. *Finite-Horizon Multi-Adversarial Team Markov Games*, July 2026b. URL https://doi.org/10.5281/zenodo.21158188.

Stephen McAleer, Gabriele Farina, Gaoyue Zhou, Mingzhi Wang, Yaodong Yang, and Tuomas Sandholm. Team-psro for learning approximate tmecor in large team games via cooperative reinforcement learning. *Advances in Neural Information Processing Systems*, 36:45402–45418, 2023.

H Brendan McMahan, Geoffrey J Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning (ICML-03)*, pages 536–543, 2003.

Dov Monderer and Lloyd S. Shapley. Potential games. *Games and Economic Behavior*, 14(1):124–143, 1996.

Christos H Papadimitriou and Tim Roughgarden. Computing correlated equilibria in multi-player games. *Journal of the ACM (JACM)*, 55(3):1–29, 2008.

Chanwoo Park, Kaiqing Zhang, and Asuman E. Ozdaglar. Multi-player zero-sum markov games with networked separable interactions. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, 2023. URL http://papers.nips.cc/paper_files/paper/2023/hash/75c411b0a06fa9e78f2a516b57b2ce62-Abstract-Conference.html.

T. Parthasarathy and T. E. Raghavan. An orderfield property for stochastic games when one player controls transition probabilities. *J. Optim. Theory Appl.*, 33(3):375–392, March 1981. ISSN 0022-3239. doi: 10.1007/BF00935250. URL https://doi.org/10.1007/BF00935250.

T. Raghavan, S. Tijs, and O. Vrieze. On stochastic games with additive reward and transition structure. *Journal of Optimization Theory and Applications*, 47:451–464, 12 1985. doi: 10.1007/BF00942191.

Rahul Savani and Theodore L Turocy. Gambit: The package for computation in game theory, version 16.3.1, 2025. URL https://gambitproject.readthedocs.io/en/latest/index.html.

Milind Tambe. *Security and game theory: algorithms, deployed systems, lessons learned*. Cambridge university press, 2011.

Emanuel Tewolde and Vincent Conitzer. Game transformations that preserve nash equilibria or best-response sets. In Kate Larson, editor, *Proceedings of the Thirty-Third International Joint Conference on Artificial Intelligence, IJCAI-24*, pages 2984–2993. International Joint Conferences on Artificial Intelligence Organization, 8 2024. doi: 10.24963/ijcai.2024/331. URL https://doi.org/10.24963/ijcai.2024/331. Main Track.

Frank Thuijsman and T. Raghavan. Perfect information stochastic games and related classes. *International Journal of Game Theory*, 26, 03 1997. doi: 10.1007/s001820050042.

Bernhard Von Stengel and Daphne Koller. Team-maxmin equilibria. *Games and Economic Behavior*, 21(1-2):309–321, 1997.

OJ Vrieze, SH Tijs, Tirukkannamangai ES Raghavan, and JA Filar. A finite algorithm for the switching control stochastic game. *Operations-Research-Spektrum*, 5(1):15–24, 1983.

Hui Zhang, Christian Wernz, and Danny R. Hughes. A stochastic game analysis of incentives and behavioral barriers in chronic disease management. *Service Science*, 10(3):302–319, 2018. doi: 10.1287/serv.2018.0211. URL https://doi.org/10.1287/serv.2018.0211.

Approximating Nash Equilibria in Finite-Horizon Multi-Adversarial Team Markov Games

Prasanna Maddila[1] **Régis Sabbadin**[1] **Meritxell Vinyals**[1]

[1]Univ Toulouse, INRAE, MIAT, Castanet-Tolosan, France

A PRELIMINARIES: EQUAL-VALUE AND ZERO-SUM PROPERTIES IN FH-MATMG

Proof of Proposition 1

Proof. We prove it by (backward) induction on h on the Bellman Expectation equation. First, note that :

$$V_{i,H}^{\mathbf{x},\mathbf{y}}(s) = r_{team}(s, \mathbf{x}^H, \mathbf{y}^H), \forall i \in \mathcal{N}, \forall s \in \mathcal{S}$$

Thus, the induction hypothesis holds at step $h = H$. Now, for some time-step $1 \leq h < H$, assume that the induction hypothesis holds at $h + 1$, that is:

$$V_{i,h+1}^{\mathbf{x},\mathbf{y}}(s) = V_{team,h+1}^{\mathbf{x},\mathbf{y}}(s), \forall i \in \mathcal{N}, \forall s \in \mathcal{S}$$

Then, for time-step h we get the following Bellman Expectation equation $\forall s \in \mathcal{S}$:

$$\begin{aligned} V_{i,h}^{\mathbf{x},\mathbf{y}}(s) &= r_{team}(s, \mathbf{x}^h, \mathbf{y}^h) + \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s, \mathbf{x}^h, \mathbf{y}^h) V_{team,h+1}^{\mathbf{x},\mathbf{y}}(s') \\ &= V_{team,h}^{\mathbf{x},\mathbf{y}}(s) \end{aligned}$$

and we are done. □

Proof of Proposition 2

Proof. We prove it by (backward) induction on h in the Bellman Expectation equation. First, note that:

$$\begin{aligned} V_{team,H}^{\mathbf{x},\mathbf{y}}(s) &= r_{team}(s, \mathbf{x}^H, \mathbf{y}^H) \\ &= -\sum_{j \in \mathcal{M}} r_j(s, \mathbf{x}^H, \mathbf{y}^H) = -\sum_{j \in \mathcal{M}} V_{j,H}^{\mathbf{x},\mathbf{y}}(s) \end{aligned}$$

Thus, the induction hypothesis holds at step $h = H$. Now, for some time-step $1 \leq h < H$, assume that the induction hypothesis holds at $h + 1$, that is:

$$V_{team,h+1}^{\mathbf{x},\mathbf{y}}(s) = -\sum_{j \in \mathcal{M}} V_{j,h+1}^{\mathbf{x},\mathbf{y}}(s), \forall s \in \mathcal{S}$$

Then, for time-step h we get the following Bellman Expectation equations: $\forall s \in \mathcal{S}$,

$$\begin{aligned} V^{\mathbf{x},\mathbf{y}}_{team,h}(s) &= r_{team}(s, \mathbf{x}^h, \mathbf{y}^h) + \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s, \mathbf{x}^h, \mathbf{y}^h) V^{\mathbf{x},\mathbf{y}}_{team,h+1}(s') \\ &= -\sum_{j \in \mathcal{M}} r_j(s, \mathbf{x}^h, \mathbf{y}^h) - \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s, \mathbf{x}^h, \mathbf{y}^h) \cdot \left(\sum_{j \in \mathcal{M}} V^{\mathbf{x},\mathbf{y}}_{j,h+1}(s') \right) \\ &= -\sum_{j \in \mathcal{M}} \left(r_j(s, \mathbf{x}^h, \mathbf{y}^h) + \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s, \mathbf{x}^h, \mathbf{y}^h) \cdot V^{\mathbf{x},\mathbf{y}}_{j,h+1}(s') \right) \\ &= -\sum_{j \in \mathcal{M}} V^{\mathbf{x},\mathbf{y}}_{j,h}(s) \end{aligned}$$

and we are done. □

B NASH-VI FOR FINITE-HORIZON ATMGS

Algorithm 3: Nash VI for FH-MG

Input: FH-MG Γ, precision parameter ϵ
Output: $\{\mathbf{x}^{s,h}\}_{s \in \mathcal{S}, h \in [H]}$

Initialize: $V_{i,H+1}(s) \leftarrow 0$ for all $i \in \mathcal{N}$ and $s \in \mathcal{S}$;
for $h = H \rightarrow 1$ **do**
  for $s \in \mathcal{S}$ **do**
    // Construct stage game $\mathcal{G}_{s,h}(V_{\cdot,h+1})$
    $\mathcal{G}_{s,h}(V_{\cdot,h+1}) = \langle \mathcal{N}, (\mathcal{A}_i)_{i \in \mathcal{N}}, \{U_i\}_{i \in \mathcal{N}} \rangle$
    // Solve $\mathcal{G}_{s,h}$ to precision $\frac{\epsilon}{H}$
    $\mathbf{x}^{s,h} \leftarrow$ NE-Oracle$(\mathcal{G}_{s,h}(V_{\cdot,h+1}), \epsilon/H)$
    // Update value functions
    for $i \in \mathcal{N}$ **do**
      $V_{i,h}(s) \leftarrow r_i(s, \mathbf{x}^{s,h}) + \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s, \mathbf{x}^{s,h}) \cdot V_{i,h+1}(s')$
    end
  end
end

This section establishes that Nash-VI (Algorithm 3), when applied to a finite-horizon ATMG and using the algorithm of Anagnostides et al. [2023] as a NE-Oracle, computes an ε-NE in time polynomial in $\frac{1}{\varepsilon}$ and in the natural parameters of the game (as formalised in Theorem 4).

Definition 5 (Finite-Horizon ATMG). *A finite-horizon ATMG (FH-ATMG) is defined by the tuple $\Gamma = \langle \mathcal{S}, \mathcal{N}, (\mathcal{A}_i)_{i \in \mathcal{N}}, \mathcal{B}_{adv}, r_{adv}, \mathbb{P}, H, \rho \rangle$ where $n = |\mathcal{N}|$ team agents play against a single adversary over a time horizon H. Formally:*

- *$\mathcal{S}$, with cardinality $S = |\mathcal{S}|$, is the (finite) state space.*
- *$\mathcal{A}_i$ is the strategy space of team agent $i \in \mathcal{N}$, with $\mathcal{A} := \prod_{i \in \mathcal{N}} \mathcal{A}_i$ denoting the team joint strategy space and $\mathbf{a} = (a_1, ..., a_n) \in \mathcal{A}$ denoting an element of this set;*
- *$\mathcal{B}_{adv} = \mathcal{B}$ is the strategy space of the adversary, with $b \in \mathcal{B}$ denoting an element of this set;*
- *$\mathbb{P} : \mathcal{S} \times \mathcal{A} \times \mathcal{B} \rightarrow \Delta(S)$[1] is the transition probability function so that $\mathbb{P}(s'|s, a, b)$ is the probability of transitioning to state s' given the current state s, the joint strategy of the team $a \in \mathcal{A}$ and the adversary's strategy $b \in \mathcal{B}$.*

[1] In the general finite-horizon setting, it is possible to specify different transition and reward functions for each time step $h \in H$. Without loss of generality, we focus on the case where the transition and reward functions are stationary.

- $r_{adv} : \mathcal{S} \times \mathcal{A} \times \mathcal{B} \to [0,1]$ *is the reward function of the adversary. We have for all the team agents* $i \in \mathcal{N}$, $r_i(s,a,b) = r_{team}(s,a,b)$ *and the game is zero-sum in the sense that*

$$r_{adv}(s,a,b) + r_{team}(s,a,b) = 0 \tag{7}$$

  for all $s \in \mathcal{S}, (\mathbf{a}, b) \in \mathcal{A} \times \mathcal{B}$.

- *Finally,* $\rho \in \Delta(\mathcal{S})$ *is the initial state distribution, and is not necessarily full support.*

Note that a FH-ATMG is the Markov game extension of normal-form ATGs Von Stengel and Koller [1997] and the single-adversary case FH-MATMG (Definition 1).

We restrict our attention to the class of FH-ATMGs that satisfy the Polynomial Expectation Property (PEP) for both the (adversary) reward functions and the transition dynamics, i.e., Assumptions 1 and 2.

We first note that as a consequence of Propositions 1 and 2, we have that the value functions of a FH-ATMG satisfy Properties 1 and 2. This is formalised below.

Corollary 1. *For an ATMG* Γ *and some joint policy* $(\mathbf{x}, \mathbf{y})$*, we have that*

1. $V_{i,h}^{\mathbf{x},\mathbf{y}}(s) = V_{i',h}^{\mathbf{x},\mathbf{y}}(s) = V_{team,h}^{\mathbf{x},\mathbf{y}}(s),\ \forall i, i' \in \mathcal{N}, \forall s \in \mathcal{S}$.
2. $V_{team,h}^{\mathbf{x},\mathbf{y}}(s) = -V_{adv,h}^{\mathbf{x},\mathbf{y}}(s),\ \forall s \in \mathcal{S}$.

Proof. Parts 1 and 2 follow directly from setting $\mathcal{M} = \{adv\}$ in Propositions 1 and 2, respectively. □

We now define the induced stage games that are defined and solved within the Nash-VI algorithm when applied to a FH-ATMG.

Definition 6. ***(ATMG stage games)*** *Given an ATMG* Γ*, for each time-step* $h \in [H]$*, state* $s \in \mathcal{S}$*, and arbitrary fixed continuation value functions* $(\mathbf{v}_{i,h+1} \in \mathbb{R}^S : i \in \mathcal{N}, \mathbf{v}_{adv,h+1} \in \mathbb{R}^S)$*, define the stage game as the following normal-form game* $\mathcal{G}_{s,h}(\mathbf{v}) := \langle \mathcal{N} \cup \{adv\}, ((\mathcal{A}_i)_{i\in\mathcal{N}}, \mathcal{B}_{adv}), ((U_i)_{i\in\mathcal{N}}, U_{adv}) \rangle$*, where the payoff functions are defined as*

$$U_{adv}(s,a,b) = r_{adv}(s,a,b) + \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s,a,b)\mathbf{v}_{adv,h+1}(s') \tag{8a}$$

$$\begin{aligned} U_i(s,a,b) &= r_i(s,a,b) + \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s,a,b)\mathbf{v}_{i,h+1}(s') \\ &= r_{team}(s,a,b) + \sum_{s' \in \mathcal{S}} \mathbb{P}(s'|s,a,b)\mathbf{v}_{i,h+1}(s') \end{aligned} \tag{8b}$$

Observation 4 (ATMG stage games are ATGs)**.** *As a consequence of Corollary 1, we have that for continuation value functions* $\mathbf{v}$ *that satisfy Properties 1 and 2,* $\mathcal{G}_{s,h}(\mathbf{v})$ *are ATGs. This is in sharp contrast with the general FH-MATMG case, whose stage games are not MATGs (See Observation 2).*

Finally, we state the following observation concerning adversary utility functions in stage games.

Observation 5 (PEP of adversary utility functions in stage games)**.** *For any continuation value vector* $\mathbf{v}$*, if the adversary reward functions and transition kernels used to define the adversary's utility functions in a stage game* $\mathcal{G}(\mathbf{v})$ *satisfy the PEP (Assumptions 1 and 2), then the mixed extensions of such utility functions (i.e., the extensions taking mixed policies as inputs) also satisfy the PEP (i.e. it exists a polynomial-time oracle that, given mixed policies, computes the expected utility).*

Theorem 4. ***(Formal version of Theorem 1).*** *Let* Γ *be a Finite-Horizon ATMG and let* $\varepsilon > 0$*. Then, under Assumptions 1 and 2, Nash-VI (Algorithm 3) using GDM algorithm from Anagnostides et al. [2023] as NE-Oracle, computes a non-stationary* ε*-NE for* Γ *in time polynomial in* $\frac{1}{\varepsilon}$ *and in the natural parameters of* Γ.

Proof. We prove that each of the $H \cdot |\mathcal{S}|$ iterations of Algorithm 3 runs in polynomial time:

- Line 5 of Algorithm 3 executes GDM algorithm from Anagnostides et al. [2023] with a precision ε/H on stage game $\mathcal{G}_{s,h}(V_{h+1}) = \langle \mathcal{N}(\mathcal{A}_i)_{i\in\mathcal{N}}, \mathcal{B}_{adv}, U^{s,h}_{adv} \rangle$ where $U^{s,h}_{adv}$ is not a utility vector but rather its mixed extension function that (from Observation 5) satisfies the PEP. Furthermore, as for FH-MATMGs, its values are obtained through a polynomial-time oracle, rather than stored in (an exponential size in $|\mathcal{N}|$) table. From Observation 4, we have that for each iteration (s,h) of Algorithm 3 the stage game $\mathcal{G}_{s,h}(V_{h+1})$ is an ATG. GDM runs (see Corollary 3.2 in Anagnostides et al. [2023]) in time $\text{poly}(n+1, \sum_{i\in\mathcal{N}} |\mathcal{A}_i|, |\mathcal{B}_{adv}|, \frac{H}{\varepsilon})$.
- Line 7 of Algorithm 3 updates the value function of the adversary under the (s,h)-iteration (mixed) policies $(\mathbf{x}^{s,h}, \mathbf{y}^{s,h})$. Under the PEP (Assumptions 1 and 2) both the expected adversary reward in any state s and the transition probabilities from any state s, can be computed in polynomial time and, in turn, computing the value function of each adversary takes polynomial-time.

□

C ADDITIONAL EXPERIMENTS

Here, we discuss MATG-GDM, its parameters and additional graphs related to Section 6. All code is provided at `https://forge.inrae.fr/chip-gt/fhat_matmg`, with the pinned version used for this paper provided at (Maddila et al. [2026b]).

MATG-GDM MATG-GDM, proposed by Maddila et al. [2026a], is a Fully-Polynomial Time Approximation Scheme used to compute approximate NE for (normal-form) MATGs. The algorithm generalises the GradientDescentMax (GDM) algorithm for ATGs proposed by Anagnostides et al. [2023]. At each iteration $t \leq T$, it performs a projected gradient descent step on the team strategy $\mathbf{x}^{s,h}_t$ and then solves a Linear Program to obtain the adversary strategy $\mathbf{y}^{s,h}_t$. The algorithm is guaranteed to find an ε-NE after at most $\mathcal{O}(\frac{1}{\varepsilon^4})$ iterations, when using a learning rate $\eta = \mathcal{O}(\varepsilon^2)$ for the gradient step.

Hyperparameters (MATG-GDM) For all experiments in Section 6, we specified MATG-GDM's hyperparameters as follows. We set the learning rate $\eta = 0.01$, maximum number of iterations to run $T_{max} = 5 \times 10^4$, and the desired gap to which we solve the stage game as $\varepsilon_{stage} = \varepsilon/H$. The latter choice implies that all stage games in Algorithm 1 were solved to obtain at least ε_{stage}-NE, so that the final error is bounded by $\sum_{h=1}^{H} \varepsilon_{stage} = \sum_{h=1}^{H} \varepsilon/H = \varepsilon$.

Number of Iterations In addition to the metrics plotted in Section 6, we also tracked the total number of iterations taken by MATG-GDM across all stage games, plotted here in Figure 4. This is a hardware-agnostic metric that serves as a good proxy for actual Runtime, and indicates how the algorithm scales with each tested parameter.

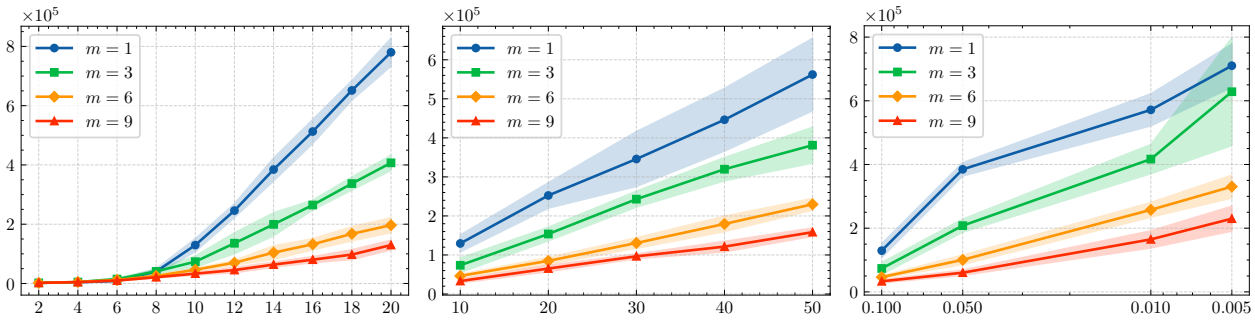
Figure 4 shows the number of iterations recorded alongside the experimental benchmark in Section 6. Figure 4a tracks the number of iterations to verify the scaling with horizon $H \in \{2, 4, \ldots, 20\}$ for games with $n = 4, a = 6, S = 10$. Figure 4b tracks the variation with the number of states $S \in \{10, 20, \ldots, 50\}$ for games with $n = 4, a = 6, H = 10$ and the default $\varepsilon = 0.1$. Lastly, Figure 4c tracks the effect of decreasing $\varepsilon \in \{0.1, 0.05, 0.01, 0.005\}$.

In all cases, we notice that the number of iterations is monotonically increasing, albeit for different reasons. Figure 4a shows this trend because we are solving more stage-games across a longer horizon. On the other hand, Figure 4b does so because there are more games to solve per horizon-step i.e. S itself is increasing. This is in contrast with Figure 2b, where increasing the size of S does not significantly affect the runtime. As noted in the main paper, this is because all stage games are solved in parallel for each horizon step $h \leq H$ by leveraging the capacities of the GPU.

Lastly, Figure 4c shows that demanding more precise equilibria impacts the number of iterations needed. For example, for $m = 1$, we observe a jump from 1.75×10^5 iterations to nearly 7×10^5 (a 4x increase). The increase is comparable with games having more adversaries, with $m = 9$ rising from 5×10^4 to 2×10^5 (another 4x increase). The $m = 3$ curve corroborates that some instances had unusually long runtimes (~$2000s$), since the number of iterations jumps in proportion with the increase in runtime.

D EXAMPLE OF ADDITIVE-TRANSITION FH-MATMG

We now turn to an application of Finite-Horizon MATMGs with Additive Transitions to marine protection, inspired by Fishery Games Filar and Vrieze [2012]. Consider the protection of a marine population by n guards (or teams of guards),



(a) Horizon (H) vs. Iterations (b) S vs. Iterations (c) ε vs. Iterations

Figure 4: Comparing the Number of Iterations taken by Nash-VI (Algorithm 1) across games with configurations $n = 4$, $a = 6$, $m \in \{1, 3, 6, 9\}$, $S = 10$ and $H = 10$ and with default desired Nash-Gap as $\varepsilon = 0.1$, unless otherwise specified.

against m fishermen. We model this as a Finite-Horizon MATMG with Additive Transitions in Section D.1.

We then provide additional experiments for these games in Section D.2. These instances provide highly structured games, helping us compare the effectiveness of Nash-VI against randomly generated games.

D.1 DEFINITION OF FISHERY GAMES

Let us define an example of AT-FH-MATMG related to illegal fishing. We assume a stock of fish is threatened to be depleted in some marine area due to illegal fishing. Illegal fisher boats perform illegal fishing campaigns, during a chosen period of the year (let us assume 4 periods of fishing are available). The Team members, on their side, patrol the area during some of these periods (one team member chooses only one period to patrol, given the length of the campaign and other duties. Every year, each illegal fisher (i.e. each adversary) chooses her fishing/patrolling period (i.e. action). We assume the reward of a fisher depends on the stock of fish (which, in return, is influenced by the activity of fishers/team members) and the number of team members patrolling the area in the same period.

The state of the system is modelled as the fish stock level, which is an integer in a fixed range. This fish level changes stochastically from year to year, and is influenced additively by the actions of the fishermen and the patrolmen. The fishers' actions tend to decrease the stock level (differently for different periods), while the team members' actions tend to increase stock.

We formalise the above description as the following AT-FH-MATMG:

$$\Gamma = \langle \mathcal{S}, \mathcal{N}, \mathcal{M}, (\mathcal{A}_i)_{i\in\mathcal{N}}, (\mathcal{B}_j)_{j\in\mathcal{M}}, (r_j)_{j\in\mathcal{M}}, \mathbb{P}, H, \rho \rangle.$$

where

1. $\mathcal{S} = \{1, \ldots, |\mathcal{S}|\}$ is the game state space represented as a finite set of integer fish stock levels. $s_h \in \mathcal{S}$ is the fish stock level at decision step $h \in \{0, \ldots, H\}$.
2. $\mathcal{N} = \{1, \ldots, n\}$ is the set of team agents/patrol Boats, patrolling against illegal fishing.
3. $\mathcal{M} = \{1, \ldots, m\}$ is the set of adversaries, representing illegal fisher boats.
4. $\mathcal{A}_i = \mathcal{B}_j = A = \{\alpha_1, \ldots, \alpha_z\}$ is a set of time periods within the calendar year, during which an illegal fishing campaign and/or a patrolling campaign may be undertaken by any agent.
5. $r_j(s, \mathbf{a}, b_j)$is the reward to fisher $j \in \mathcal{M}$, obtained when j chooses time period b_j for her illegal fishing campaign, when the profile of patrolling campaigns by team agents is $\mathbf{a}$ and when the stock level of the year is s.
   In details, we assume r_j is formed as follows:
   (a) The fish stock of the year is $R_{max}(s)$.
   (b) The fishing period $b_j \in A$ determines the potential fishing level $\text{level}(b_j) \in [0, 1]$.
   (c) The potential fishing level is decreased by the presence of team members. We let

$$\delta(\mathbf{a}, b_j) = |\{i \in \mathcal{N}, a_i = b_j\}|$$

be the number of team members patrolling during the fishing campaign of adversary j. We assume the fishing potential i reduced by a factor $\exp(-\delta(\mathbf{a}, b_j))$.

Thus, all in all, we get:

$$r_j(s, \mathbf{a}, b_j) = R_{max}(s) \times \text{level}(b_j) \times \exp(-\delta(\mathbf{a}, b_j)). \tag{9}$$

6. Now, we define the additive transition table

$$\mathbb{P}^{AT}(s'|s, \mathbf{a}, \mathbf{b}) = \sum_{i \in \mathcal{N}} \omega_{i,s} \mathbb{P}_i(s'|s, a_i) + \sum_{j \in \mathcal{M}} \omega_{j,s} \mathbb{P}_j(s'|s, b_j).$$

For simplicity we assume that the weights do not depend on the identity of agents or state. However, we assume that the total weight of team members equals the total weight of illegal fishers. As a consequence:

$$\omega_{i,s} = \frac{1}{2|\mathcal{N}|} \text{ and } \omega_{j,s} = \frac{1}{2|\mathcal{M}|}.$$

Now, we assume that the impact of the presence of team members on the fishing level in the corresponding period is positive , leading to "better" states as follows.

$$\begin{aligned} \mathbb{P}_i(s'|s, a_i) &= 0, \text{ if } s' < s, \\ \mathbb{P}_i(s' = s + \sigma|s, a_i) &\propto (1+\sigma)^{\text{level}(a_i)}, \forall \sigma \in \{0, \ldots, |\mathcal{S}| - s\}. \end{aligned} \tag{10}$$

On the other hand, the influence of the presence of illegal fishers is negative, leading to "worse" states, contrary to the presence of patrol boats. It is defined as follows:

$$\begin{aligned} \mathbb{P}_j(s'|s, b_j) &= 0, \text{ if } s' > s, \\ \mathbb{P}_j(s' = s - \sigma|s, b_j) &\propto (1+\sigma)^{\text{level}(b_j)}, \forall \sigma \in \{0, \ldots, s\}. \end{aligned} \tag{11}$$

Once the above values are assigned, the transition kernels $\mathbb{P}_i$ and $\mathbb{P}_j$ are normalised (by dividing by the sum) such that $\sum_{s' \in \mathcal{S}} \mathbb{P}_i(s'|s, a_i) = \sum_{s' \in \mathcal{S}} \mathbb{P}_j(s'|s, b_j) = 1$ for all states $s, s' \in \mathcal{S}$ and actions a_i, b_j. This ensures that these transition kernels are valid.

D.2 EXPERIMENTAL EVALUATION FOR FISHERY GAMES

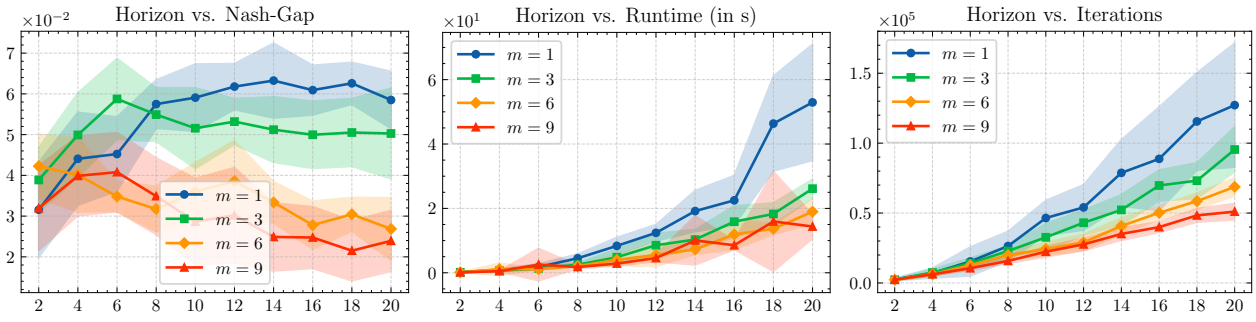
In this section, we describe the experimental results of using Nash-VI (Algorithm 1) on the Fishery Game described in Section D.1.

Implementation of Fishery Games To facilitate comparison, we use the same sizes of games as the experiments for Randomly Generated games (See Section 6). In particular, we tested Fishery Games with $S = 10$ discrete fish stock levels, $n = 4$ patrol boats, $m \in \{1, 3, 6, 9\}$ fisher boats, with the calendar year divided into $a = 6$ periods during which fishing may occur. To specify the rewards, $\{R_{max}(s) : s \in \mathcal{S}\}$ were assigned equally spaced values in the interval $[0, 1]$, and the potential fishing level, denoted level, was sampled according to a Normal distribution.

The implementation of the Fishery Game as an AT-FH-MATMG is provided in the supplied source code (`https://forge.inrae.fr/chip-gt/fhat_matmg`) as examples/ fishing .py, while the benchmark script is provided at benchmarks/benchmark_fishing.py.

Variation with Horizon Figure 5 presents our results for the Fishery Games with the selected configurations. We notice that the results for the Fishery Game for these configurations similar trends to the Randomly generated games from Figure 1. Firstly, we notice in Figure 5a that all instances have terminated with smaller gap than the desired Nash-Gap of $\varepsilon = 10^{-2}$, confirming correctness of the computed equilibria. As the number of adversaries increases, the Nash-Gap of the computed equilibrium solutions decreases. This trend is specific to the structured instances like the Fishery Game (Compare with Figure 1a), suggesting that structured games might be easier to solve as the number of adversaries increases.

Figure 5b shows the scaling of Runtime (in seconds) with respect to the Horizon H. We see that games with $m = 1$ adversaries still take less time to solve than the $m = 9$ adversary case, and notice overall the same trends as for the Randomly generated games (Compare with Figure 1b). However, the runtime is 25x faster than for the Randomly generated games.



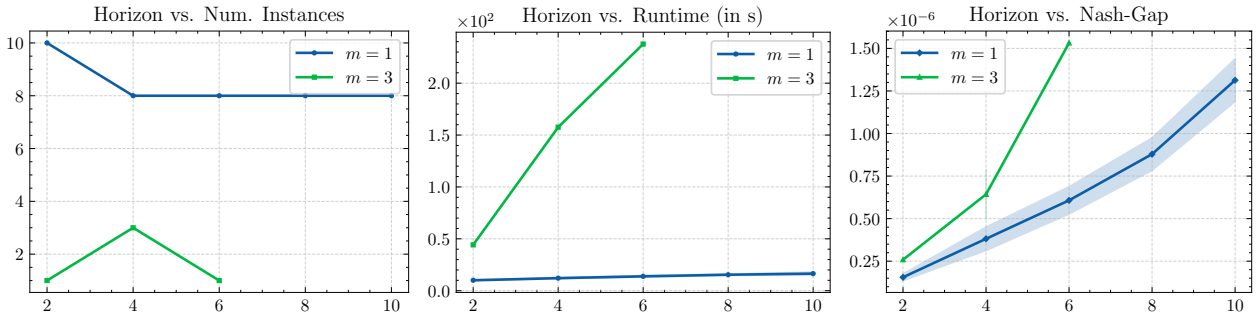
(a) Horizon (H) vs. NE-Gap (b) Horizon (H) vs. Runtime (in sec) (c) Horizon (H) vs. Iterations

Figure 5: Results for randomly generated Fishery Games with Horizon $H \in \{2, 4, \ldots, 20\}$ with $n = 4$ Patrolmen, $a = 6$ periods, $S = 10$ levels of stock and precision $\varepsilon = 0.1$.

Specifically, Nash-VI took $60s$ in the worst case of 4v1 games with $H = 20$ over $1500s$ for similarly configured Random instances.

This trend is supported by Figure 5c which shows the number of Iterations taken by the MATG-GDM stage game solver, where Random instances (See Figure 4a) took up to 6x less iterations (8×10^5 iterations for the random case over 1.2×10^5 for the Fishery Game).

E COMPARING MATG-GDM AND GAMBIT FOR STAGE-GAME SOLVING



(a) (b) (c)

Figure 6: Results for $H = \{2, 4, \cdots, 20\}$ for Algorithm 1 using Gambit IPA as Stage Game Solver with $n = 2$, $a = 6$,$S = 10$ and $m = \{1, 3, 6\}$ with timeout of 10 minutes. 10 instances were solved per configuration. No instances of $m = 6$ terminated within timeout.

Iterated Polymatrix Approximation Iterated Polymatrix Approximation (IPA) is an approximate solver for Normal-Form Games (NFGs) Govindan and Wilson [2004]. It is an approximation algorithm that works by approximating a Normal-Form Game by a sequence of polymatrix games. While it is not guaranteed to find a solution, it is quite fast in practice.

In this work, we use the IPA solver provided by Gambit (Savani and Turocy [2025]) in place of the MATG-GDM solver (Maddila et al. [2026a]) to solve the stage-games in the Nash-VI algorithm (See Algorithm 1). The key challenge is that representing these stage-games, which are MATGs, in normal form requires space exponential in the total number of players. Like all solvers relying on the normal-form representation, IPA is severely limited by this exponential growth in representation size. We defer to Maddila et al. [2026a] for a more detailed comparison of comparing IPA and MATG-GDM on many configurations of MATGs.

Game Configurations We illustrate the execution of Nash-VI (Algorithm 1) using the IPA solver as the Nash Equilibrium oracle for the Stage games. In line with the experiments in the main paper (See Section 6), we use randomly generated AT-FH-MATMGs. Following Maddila et al. [2026a], we tested configurations with the smallest team size ($n = 2$), the

number of adversaries varied as $m \in \{1, 3, 6\}$, $a = 6$ actions per agent and $S = 10$ states in total. We ran the experiments for 10 instances per configuration, and gave each instance a timeout of 10 minutes.

Experimental Results Figure 6 illustrates the results of using IPA as a stage-game solver for Nash-VI (Algorithm 1). Figure 6a reports the number of instances solved before timing out. While at least 8 out of 10 instances were solved for games with only $m = 1$ adversary, only 1-3 games with $m = 3$ adversaries were solved within the timeout. On the other hand, no instances with $m = 6$ adversaries were solved, highlighting the difficulty of solving these games with IPA.

Figure 6b reports the runtime of the instances that terminated. The Runtime for games with $m = 1$ is fairly stable (10-20 seconds), indicating that IPA can reliably solve games of this size. On the other hand, the runtime increases rapidly for instances with $m = 3$ adversaries, until all instances time out for horizons $H \geq 6$.

Figure 6c illustrates the NE-Gap of the computed equilibrium solutions. For all terminated solutions, IPA provides consistently high quality solutions with NE-Gap $\approx 10^{-6}$, even as the horizon increases. This shows that the tradeoff with using generic solvers for Normal-Form Games is between solution quality and the game size. While Nash-VI using MATG-GDM consistently provides solutions with NE-Gap $\approx 10^{-2}$, it can scale to very large instances easily (games with $n = 4$ team agents and $m = 9$ adversaries).