As Safe As It Gets: Near-Optimal Learning in Multi-Stage Games with Imperfect Monitoring

Danny Kuminov[1] and **Moshe Tennenholtz**[2]

Keywords: *DMA::Game Theoretic Foundations; PS::Planning with Incomplete Information*

Abstract. We introduce the first near-optimal polynomial algorithm for obtaining the mixed safety level value of an initially unknown multi-stage game, played in a hostile environment, under imperfect monitoring. In an imperfect monitoring setting all that an agent can observe is the current state and its own actions and payoffs, but it can not observe other agents' actions. Our result holds for any multi-stage generic game with a "reset" action.

1 Introduction

Decision making in adversarial settings is a central topic in both AI and game theory. Assuming a purely adversarial setting, playing the (mixed) safety-level strategy is the best one can hope for. Such a strategy maximizes the expected worst case payoff of the player, and it can also be computed efficiently. This leaves us with two complementary central problems. One problem is the need to deal with settings which are not purely adversarial. Another challenging issue is the need to deal with incomplete information about the environment. In particular, the game played might be unknown, and therefore guaranteeing the safety level value may be problematic. This paper deals with the latter issue.

Consider a multi-stage game. A multi-stage game consists of finitely many states, each of which is associated with a strategic form game. The actions selected by the agents in a given state determine the payoffs of the agents according to the payoff matrix of the corresponding game. Moreover, as a function of the current state and the selected actions we reach a new state. We will consider the situation where we have two agents, and we care about the payoffs that can be guaranteed by player 1 (which we refer to as the *agent*), when playing against player 2 (which we refer to as the *opponent*). If the multi-stage game starts from a given initial state, and is played along T stages then the agent can guarantee itself a particular, optimal safety-level value. This is common and highly natural solution for this general class of games. However, when the multi-stage game is unknown it is no longer clear what will be the best possibility for the agent. A clever algorithm should attempt to learn the structure of the game, in order to attempt to obtain a value which is close to the safety level value. A central issue in this regard is the type of information available to the agent. In particular, the literature distinguishes between perfect monitoring and imperfect monitoring. In the perfect monitoring setting the agent can recognize the state, and observe both its payoff and the opponent action after the state-game is played. In the imperfect monitoring setting the agent can only recognize the state and observe its own payoff; it can not observe the opponent actions. In both settings the idea is to come up with an algorithm that will guarantee that the average payoff will be close to the safety level value of the underlying game. Moreover, an important objective is that convergence to this value will be obtained in a polynomial number of iterations.

The above challenge fits into the so-called agent-centric approach to learning in games (see e.g. [10, 6]). We consider multi-stage games with incomplete information, where there is strict initial uncertainty about the game being played [8, 1]. In the context of repeated games with incomplete information [3], where the multi-stage game consists of only a single state, Banos [5] and Megiddo [9] proved the existence of an algorithm that converges to the safety-level value in any repeated game, even under imperfect monitoring. The algorithm they present, however, is highly inefficient; an efficient algorithm addressing this problem in repeated games can be found in [2]. These results, however, do not apply to general multi-stage games.[3] On the other hand, if we allow perfect monitoring, then the R-max algorithm, introduced in [7], provides a near-optimal polynomial algorithm. However, the problem of obtaining the (mixed) safety-level value in multi-stage games with imperfect monitoring was left open.

In this paper we address the above challenge, by presenting the first near-optimal polynomial algorithm for obtaining the safety-level value in generic multi-stage games, with strict initial uncertainty about the game. In a generic multi-stage game, for any given state s, action a of the agent, and actions b_1, b_2 of the opponent, the agent's payoff for (a, b_1) in s is different from its payoff for (a, b_2) in s. Although somewhat limiting, this assumption captures many interesting situations, and is quite common in the literature. Namely, given an initially unknown generic multi-stage game, we show an efficient algorithm that after polynomially many iterations (in the game size and accuracy parameters) guarantees (almost) the safety level value with overwhelming probability. A major challenge that this algorithm addresses is that given imperfect monitoring the agent can not know whether two payoffs he obtains in a given state, when playing actions a_1 and a_2 respectively, are associated with the same action by the opponent.

[1] Technion – Israel Institute of Technology, Haifa, Israel 32000. Email: dannykv@tx.technion.ac.il

[2] Technion – Israel Institute of Technology, Haifa, Israel 32000. Email: moshet@ie.technion.ac.il

[3] We have considered several ways to reduce a multi-stage game to a representation acceptable by the algorithm in [2], but all of them result in learning time that is exponential in the number of states.

2 The setting

In a multi-stage game (MSG) the players play a (possibly infinite) sequence of games from some given set of finite games (in strategic form). After playing each game, the players receive the appropriate payoff, as dictated by that game's matrix, and move to a new game. In the model we consider here, the identity of this new game is uniquely determined by the previous game and by the players' actions in it.[4] Formally:

Definition 1. *A fixed-sum, two player, multi-stage game (MSG) M on finite set of states S and finite sets of actions X_1, X_2 consists of:*

- *Stage Games: each state $s \in S$ is associated with a two-player, fixed-sum game in strategic form, where the action set of each player is X_1, X_2 accordingly, and the utility function of player 1 is $U_s : X_1 \times X_2 \to \Re$. For brevity, we denote $X = X_1 \times X_2$.*
- *Transition Function $f_{tr} : S \times X_1 \times X_2 \to S$: $f_{tr}(s, x_1, x_2)$ is the state to which the game transfers from state s given that the first player (the agent) plays x_1 and the second player (the opponent) plays x_2.*
- *Designated initial state. W.l.o.g let us denote it by $start$.*

In this work, we assume that player 1 (the row player of the stage games) does not know *a priori* what the payoff matrices are, neither he is informed after each stage about the action taken by player 2 (but he observes what his payoff at that stage was, and he knows what the current state is before playing the respective game). Player 2 (the column player), however, is fully informed about both the payoff matrices and the history of the game. We use the following definitions:

- The set of histories of length t of M is $H^t = \prod_{i=1}^{t} (S \times X)$.
- The set of all finite histories is $H = \bigcup_{k=0}^{\infty} H^k$ (here we use the simplifying notation that $H^0 = \{e\}$, where e denotes the empty history).
- $H^\infty = \prod_{i=1}^{\infty} (S \times X)$ is the set of all *infinite* histories.
- Given a history h, we will slightly abuse notation and denote by $U_i(h^t)$ the payoff to player i in round t of the history h.
- A behavioral policy of the informed player (player 2) in the multi-stage game is a function $p_2 : H \times S \to \Delta(X_2)$.
- The set of histories of the game which is available to the uninformed player (player 1) at stage t is $H'^t = \prod_{l=1}^{t} (S \times X_1 \times \Re)$.
- We denote $H' = \bigcup_{k=0}^{\infty} H'^k$.
- A behavioral policy of the uninformed player in the multi-stage game is a function $p_1 : H' \times S \to \Delta(X_1)$. The S in the function parameters represents the current state of the game.
- We denote the set of possible behavioral policies of player i by P_i.

A utility function in this setting is a function $\tilde{U_i} : H^\infty \to \Re$. There are several possible definitions of this function based on the utilities in the one-shot game; we will use the function $\tilde{U_i}(h) = \liminf_{t\to\infty} \frac{1}{t} \sum_{k=1}^{t} U_i(h^k)$. The true game M together with the players' behavioral policies generate a probability measure over H^∞, which can be described uniquely by its values for finite cylinder sets. Given this measure, the expected utility of player i given policies p_1, p_2 is defined as

$$\tilde{U_i}(p_1, p_2) = \liminf_{t\to\infty} \sum_{h\in H^t} Pr\langle h|(p_1, p_2)\rangle \frac{1}{t} \sum_{k=1}^{t} U_i(h^k)$$

where $Pr\langle h|(p_1, p_2)\rangle$ denotes the probability that a finite history $h \in H^t$ occurs in the first t stages of the game.

[4] Note that this model is a special case of the well known stochastic game model.

In this work, we will assume that all payoffs in the stage games are positive and bounded from above by U_{max}, that the stage games are generic and that there is a designated action (w.l.o.g let us denote it by $reset$) that from any state and given any opponent action, transfers the game to the initial state and gives payoff 0 to the agent.[5] In a generic game, for any given action a of the agent, and actions b_1, b_2 of the opponent, the agent's payoff for (a, b_1) is different from its payoff for (a, b_2). We also assume that the above is the only prior information available to the agent. He does not know *a priori* the exact payoff matrices of the stage games, neither does he know *a priori* what the transition function is for actions that are not $reset$.

2.1 Unknown time horizon

Let $V_1 = \max_{p_1\in P_1} \min_{p_2\in P_2} \tilde{U_1}(p_1, p_2)$ be the safety level value of player 1 in the multi-stage *complete information* game, then an intuitive goal would be to develop a policy p_1 for player 1 which will, given confidence δ and accuracy ϵ, achieve after $\hat{l} = poly(m, n, \log\left(\frac{1}{\delta}\right), \frac{1}{\epsilon})$ rounds an actual average payoff of at least $(1-\epsilon)V_1$ with probability at least $1-\delta$. [6] Formally, for any game and any policy p_2 of the opponent in the multi-stage game:

$$Pr_{p_1,p_2}\left[\forall l \geq \hat{l} : \frac{1}{l}\sum_{t=1}^{l} U_1(h^t) \geq (1-\epsilon)V_1\right] \geq 1-\delta$$

Unfortunately, this goal cannot be achieved, as demonstrated by the following example (the matrix entries have the form (*row player payoff*, *next state*):

I	L	R
G	2, I	3, I
R	0, I	0, I
U	1,II	0, I
D	0, I	1,II

II	L	R
R	0, I	0, I
U	1,III	0, I
D	0, I	1,III

$\ldots$

$\ldots$

$\mathbf{S-1}$	L	R
R	0,I	0,I
U	1,S	0,I
D	0,I	1,S

S	C
R	0,I
C	U,S

I is the initial state, R denotes the $reset$ action required by the model, and G is an action that guarantees 2 right from the start. Except for those two actions, this example consists of $|S|-1$ consecutive "matching penny" games, in which victory for the row player advances him one state further and loss returns him to the initial state. The last state is a "heaven" state - there the row player always gets U and he can remain there forever. Now there are two cases:

$U > 2$: The safety level of the row player is U, and it takes him $\Omega(2^{|S|})$ steps to get to the last state and obtain the safety level payoff (he needs to win $|S|-1$ consecutive "matching penny" games).

$U \leq 2$: The safety level of the row player is 2, and he can obtain it immediately by playing G in the starting state.

[5] The $reset$ action ensures that there are no irreversible actions – the agent can always return to the starting state. It can be easily verified that learning is impossible otherwise, since, by trying an unknown action, the agent might trap himself in an inferior subgame, without any possibility for going back.

[6] Note that the average is taken over all stages of the game, including the initial learning period $\hat{l}$.

The point is that the agent does not know *a priori* whether $U > 2$, and when it is not so, he has to play $\Omega(2^S)$ stage games just to find out that $U \leq 2$, and that he could obtain his safety level payoff in just one step. Therefore, the learning time can be exponential relative to the number of states, even if the optimal policy can achieve its reward in polynomial time.

2.2 Known time horizon

The lesson from the previous example is that in order to learn efficiently, one needs to know the number of steps that is required by the optimal policy to reach its target expected payoff. A possible solution is, given a time limit T, to limit ourselves only to policies that play for T steps and always do $reset$ on step $T+1$ (and then repeat themselves *ad infinitum*), both in our learning algorithm and in the optimal policy to which we compare. Let $V_{p_1,p_2}(T) = E_{p_1,p_2}\left[\frac{1}{T+1}\sum_{t=1}^{T} U_1(h^t)\right]$ denote the expected average payoff guaranteed by such policy p_1 against opponent policy p_2, and $V(T) = \max_{p_1}\min_{p_2} V_{p_1,p_2}(T)$ denote the maximal expected average payoff that can be guaranteed by such policy. Given this definition, our goal is to develop a policy p_1 for player 1 which, given confidence δ, accuracy ϵ and finite time horizon T, guarantees after $\hat{l} = poly(|X_1|, |X_2|, |S|, \frac{1}{\delta}, \frac{1}{\epsilon}, T)$ rounds an expected average payoff of at least $(1-\epsilon)V(T)$ with probability at least $1-\delta$. [7] Formally, for any game for which the above assumptions hold and for any policy p_2 of the opponent:

$$Pr_{p_1,p_2}\left[\forall l \geq \hat{l} : \frac{1}{l}\sum_{t=1}^{l} U_1(h^t) \geq (1-\epsilon)V(T)\right] \geq 1-\delta$$

Note that the optimal policy under this criterion can be described as a mapping $S \times \{1 \dots T\} \rightarrow \Delta(X_1)$. Informally, the policy only has to take into account the current state and the number of steps remaining in the current T-steps sequence, when determining the next action – the specific previous history does not matter. This means that the T-step min-max policy can be described concisely (i.e., the size of its representation is polynomial in T and the problem parameters) and that it can be computed efficiently, by combining backward induction with the usual techniques for computing mixed min-max strategy in strategic form games. In fact, this observation holds for any stochastic game, which is a more general model.

3 The algorithm

The basic idea of the algorithm can be summarized as follows:

- In each iteration, the algorithm constructs an approximate (optimistic) model of the multi-stage game, computes the T-step optimal strategy for it and executes it.
- The agent represents its knowledge about the game matrix of each stage game by a partition of the set of opponent's actions. For each element of the partition and each action of the agent, it keeps the set of payoffs associated with that subset of the opponent's actions.
- With some small probability the algorithm will explore in the current iteration - that it, it will draw a number $i \in [1, 2, \dots, T]$ (distributed uniformly and independently) and in round i of the iteration, it will play a random action (distributed uniformly and independently) and count the number of times each distinct payoff was encountered for each action in each state during the sampling. After sampling, it will play $reset$ and start the next iteration.
- The algorithm uses the counter values accumulated by sampling as an estimates of the number of times the opponent used his corresponding (hidden) action since the last model update. Therefore, for two payoff values in different columns that correspond to the same (hidden) action, those values must be similar.
- When updating its model, for each stage game, the algorithm tries to find a refinement of the partition of the opponent's actions so that payoffs with sufficiently different counts[8] are in different groups, and payoffs with similar counts are in the same group (and the new partition is the same for all rows).
- We prove that, with high probability, if there are two groups of actions that the opponent used sufficiently different number of times, we will be able to separate the respective payoffs correctly *in all rows* - we will learn something about the game matrix.
- Otherwise, the difference between the number of times that the opponent used actions that are in a given element of the partition is small. Note that when constructing the tentative model, the algorithm treats each element as a single *meta-action*, takes the payoff for the agent when the opponent plays this meta-action to be the average of the distinct payoffs associated with it, and takes the transition function for this meta-action to be uniformly distributed over the successive states for the payoff values that are associated with it.[9] Given that the above difference is small, the algorithm obtains a sufficiently high payoff when using this model.

We assume that the agent knows *a priori* the following parameters of the problem:

- $|S|$ - the number of the states in the multi-stage game.
- $|X_1|, |X_2|$ - the sizes of the strategy sets (of the agent and the opponent respectively).
- U_{max} - the largest possible payoff for the agent in the game.
- ϵ, δ - accuracy parameters.

We will also use the following notation:

- β, γ - two parameters that control the behavior of the algorithm (to be determined later).
- $S' = (S \times \{1, \dots, T\}) \bigcup \{0\}$ is the extended set of states. Note that we add a fictitious state 0 to the model, and we treat being in the same state at different times of the T-step sequence as being in different states.[10]
- Let C_s be a variable that holds the counters that the algorithm maintains, for a stage game $s \in S'$. Specifically, $C_s : X_1 \times [0, U_{max}] \rightarrow \mathbb{N}$ is a function that maps the distinct payoff values for each row to the number of times they were encountered while sampling that row. We denote by C the set of all such variables.
- Let Ω_s and ϕ_s, for $s \in S'$, be two variables that represent the partial knowledge that the algorithm has regarding the game matrix (of stage game s). Specifically, Ω_s is a partition over the opponent's action set and $\phi_s : X_1 \times \Omega_s \rightarrow 2^{[0,U_{max}]}$ is a function that maps (for each row) elements of the partition to associated groups of payoff values. Note that the initial state of "no

[7] Note that the average is taken over all stages of the game, including the initial learning period $\hat{l}$.

[8] We use the word "count" to denote the number of times the algorithm encountered a specific payoff value while sampling, as opposed to the actual number of times the respective action was used by the opponent.

[9] Note that although the real transition function is deterministic, the algorithm uses a stochastic game as a tentative model.

[10] This distinction is required, since the optimal policy must be able to treat these states differently.

knowledge" is represented by $\forall s \in S' : \Omega_s = \{X_2\}$ and $\forall s \in S', i \in X_1 : \phi_s(i, X_2) = \emptyset$, and a state of complete and accurate knowledge is represented by $\Omega_s = \{\{j\} | j \in X_2\}$ and $\phi_s(i, \{j\}) = \{U_{ij}\}$.

- Let $f_{tr} : S' \times X_1 \times [0, U_{max}] \to S'$ be the transition function that the algorithm maintains (since the game is generic, utility values can be used in place of opponent actions).
- Let $l_s(\omega)$, for $\omega \in \Omega_s$, be a variable that holds the number of times a stage game $s \in S'$ has been played *and* the opponent used an action in ω since the last time the partition Ω_s was refined. We denote by l the set of all such variables.

The algorithm uses the following sub-routines:

Procedure RecordNewPayoff($s, s', x, u, \Omega_s, \phi_s, f_{tr}, C_s$)
  Let $\omega' \in \arg\min_{\omega \in \Omega_s} \min_{u \in \phi_s(x,\omega)} C_s(x, u)$
  Let $\phi_s(x, \omega') := \phi_s(x, \omega') \cup \{u\}$
  Let $f_{tr}(s, x, u) = s'$
End procedure
Procedure ResetCounters(C, l)
  For all $s \in S'$
    $\forall x \in X_1, u \in [0, U_{max}] : C_s(x, u) := 0$
    $\forall \omega \in \Omega_s : l_s(\omega) := 0$
  End for
End procedure

Now we define the algorithm:

// Initialization
For all $s \in S'$
  For all $x_1 \in X_1 \setminus \{reset\}$
    Let $f_{tr}(s, x_1, U_{max}) = 0$
For all $s \in S'$
  Let $f_{tr}(s, reset, U_{max}) = (start, 1)$
For all $s \in S'$
  $\Omega_s := \{X_2\}; l_s(X_2) = 0$
  For all $x \in X_1$
    $\phi(x, X_2) := \emptyset$
  For all $x \in X_1, u \in [0, U_{max}]$
    $C_s(x, u) := 0$
// Endless loop
While true
  // Model update
  For all $s \in S'$
    For each $\omega \in \Omega_s$:
      For each row $i \in X_1$
        Let $(u_{i1}, \ldots, u_{i|\omega|})$ be the elements of $\phi_s(i, \omega)$, ordered in non-decreasing order of $C_s(i, u)$.
        If $|\phi_s(i, \omega)| < |\omega|$ then
          // the number of observed payoffs for ω
          // is less than the number of actions in ω
          Add $(|\omega| - |\phi_s(i, \omega)|)$ entries of $U_{max} + 1$ to $\phi_s(i, \omega)$ (with count 0). [11]
        End if
      End for
      If there exists $1 < k \leq |\omega|$ such that $\forall i \in X_1$:
          $C_s(i, u_{ik}) - C_s(i, u_{i(k-1)}) > 2\gamma l_s(\omega)^{\frac{3}{4}}$,
        // Here we split (refine) the partition
        split $\omega = \{y_1, \ldots, y_{|\omega|}\}$ into
          $\omega_1 = \{y_1, \ldots, y_{k-1}\}$
          and $\omega_2 = \{y_k, \ldots, y_{|\omega|}\}$.
          Replace ω with ω_1, ω_2 in Ω_s.
          Modify ϕ_s so that $\forall i \in X_1$:
            $\phi_s(i, \omega_1) = \{u_{i1}, \ldots, u_{i(k-1)}\} \setminus \{U_{max} + 1\}$
            and $\phi_s(i, \omega_2) = \{u_{ik}, \ldots, u_{i|\omega|}\} \setminus \{U_{max} + 1\}$
        End if
      End for
      Repeat the previous loop until no more splits are made.
      If any split was made
        $\forall x \in X_1, u \in [0, U_{max}] : C_s(x, u) := 0$
        $\forall \omega \in \Omega_s : l_s(\omega) := 0$
End for
Build a stochastic game in which:
  S' is the set of states
  The game matrix $U_s \in \Re^{|X_1| \times |\Omega_s|}$ for each state $s \in S'$ is:
  For each $i \in X_1$
    For each $\omega \in \Omega_s$
      Let $U_{ik} = \frac{1}{|\omega|} \left(\sum_{u_i \in \phi(i,\omega)} u_i + \right.$
        $\left. + (|\omega| - |\phi(i, \omega)|) U_{max} \right)$ // See [12]
    End for
  End for
  The game matrix $U_s \in \Re^{|X_1| \times |X_2|}$ for the state 0 is:
    For all $x_1 \in X_1, x_2 \in X_2 : U_{x_1, x_2} = U_{max}$
  For all states $s \in S', t \in S' \setminus \{0\}$, agent action $x \in X_1$ and opponent meta-action $\omega \in \Omega_s$, the transition probability is: $Pr(s, x, \omega, t) =$
  $= \frac{1}{|\omega|} |\{u \in \phi_s(x, \omega) : f_{tr}(s, x, u) = t\}|$
  For all states s, agent action $x \in X_1$ and opponent meta-action $\omega \in \Omega_s$, the probability of transition to state 0 is:
    $Pr(s, x, \omega, 0) = \frac{|\omega| - |\phi(x,\omega)|}{|\omega|}$ // See [13]
  For the state 0, the transition function is
    $\forall x_1 \in X_1, \omega \in \Omega_0 : Pr(0, x_1, \omega, 0) = 1$
    $\forall x_1 \in X_1, \omega \in \Omega_0, s \in S' : Pr(0, x_1, \omega, s) = 0$
Compute the T-step mixed safety level strategy for this stochastic game.
Let $explore$ be a random boolean value with $P(explore = true) = \beta$
Let i be an integer selected from $[1, T]$ with uniform probability
Repeat for t from 1 to T
  Let s denote the current state.
  If $explore = true$ and $t = i$:
    Let $x \in X_1$ be an action selected at random with uniform probability
    Execute action $x \to$
      let u be the observed payoff and s' - the new state.
    Let $C_s(x, u) := C_s(x, u) + 1$
    If $\nexists \omega \in \Omega_s : u \in \phi_s(x, \omega)$ then
      Call RecordNewPayoff($s, s', x, u, \Omega_s, \phi_s, f_{tr}, C_s$)
      Call ResetCounters(C, l)
    End if
    Break // T-step Repeat
  End if
  Let x be the action prescribed by the safety level strategy for the current state and step.
  Execute action $x \to$
    let u be the observed payoff and s' - the new state.
  If $\nexists \omega \in \Omega_s : u \in \phi_s(x, \omega)$ then
    Call RecordNewPayoff($s, s', x, u, \Omega_s, \phi_s, f_{tr}, C_s$)

Call ResetCounters(C, l)
Break // T-step Repeat
End if
End // T-step Repeat
Play $reset$
End while

4 The analysis

Let $\tilde{H}^\infty = (X \times \{true, false\} \times X_1)^{\mathbb{N}}$ be the set of all infinite histories of the game that includes information about the realization of the random decision variables used by the algorithm (whether an exploration has been done in a specific round and the row chosen for exploration). The true multi-stage game M together with both players' policies generate a probability measure over $\tilde{H}^\infty$, which can be described uniquely by its values for finite cylinder sets. All random variables that we use in this analysis are derived from this probability measure.

We show:

Theorem 1. *Given a multi-stage game that conforms to the requirements set in Section 2, $\epsilon > 0, \delta > 0$, there exists $\hat{l} = poly(|X_1|, |X_2|, T, |S|, U_{max}, \frac{1}{\delta}, \frac{1}{\epsilon})$ such that for any policy of the opponent, the above algorithm achieves in every round $l \geq \hat{l}$ an expected average (over all rounds since the start of the game) payoff of at least $(1-\epsilon)\,V(T)$ with probability at least $1-\delta$, where $V(T)$ is the maximal expected average payoff that can be guaranteed after playing T steps.*

To prove the theorem, we need the following notation:

1. We use the notation $[condition]$ to denote an indicator variable that equals 1 if the condition holds and 0 otherwise.
2. Let $(l_1, l_2 \ldots)$ be the indices of the rounds of the multi-stage game at which the algorithm updates the partition and/or records a new payoff value for one of the states (note that there can be at most $|X_2||S|T + |X_1||X_2||S|T$ such rounds), and let us divide the rounds of the game into epochs $((0, \ldots, l_1 - 1), (l_1, \ldots, l_2 - 1), (l_2, \ldots, l_3 - 1), \ldots)$.
3. For brevity, we will denote by $Q_1, Q_2, \ldots$ constant values that are polynomial with respect to the problem parameters and are constant throughout the execution of the algorithm. In particular, we will denote by $Q_1 = (|X_1| + 1)|X_2||S|T + 1$ the maximal number of epochs.
4. For a given stage game $s \in S'$ and epoch $e = (l_i, \ldots, l_{i+1} - 1)$, let $C^l_{s,e}$ be the counter function that the algorithm maintains at round l of the epoch (i.e., at round $l_i + l$ of the game) for the stage game s.
5. For an epoch $e = (l_i, \ldots, l_{i+1} - 1)$ and for each $j \in X_2$, let $F^l_{s,e}(j)$ be the number of times that the stage game s was played in the first l rounds of the epoch *and* the opponent played j. Note that, by definition, the value of $l_s(\omega)$ at round l of the epoch equals to $\sum_{j \in \omega} F^l_s(j)$.
6. When the epoch under consideration is clear from context, we will omit the subscript e.
7. Note that all of the above are random variables.
8. Note that the probability that sampling occurs in a given round of the multi-stage game is $\frac{\beta}{T}$ and the probability that a specific action is sampled is $\frac{\beta}{T|X_1|}$, independent of any other random variables or the actions of the opponent.
9. Therefore, for any stage game s and actions $i \in X_1, j \in X_2$, the expected value of the counter maintained by the algorithm ($C^l_s(x_i, U_{ij})$) given the value of $F^l_s(j)$ is $\frac{\beta F^l_s(j)}{T|X_1|}$.

The following Lemma shows that the counters maintained by our algorithm represents in an adequate manner the frequency in which actions are used by the opponent.

Lemma 1 (Counter accuracy)**.** *Let us examine a specific epoch $e = (l_k, \ldots, l_{k+1} - 1)$. There exists $\gamma = poly(|S|, T, \frac{1}{\delta}, |X_1|, |X_2|)$ so that for any policy of the opponent and any $0 < \beta < 1$:*

$$Pr\left[\begin{array}{l}\exists l \in \mathbb{N} \\ \exists i \in X_1 \\ \exists s \in S' \\ \exists j \in \omega \in \Omega_s\end{array} : \left| C^l_s(i, U_{ij}) - \frac{\beta F^l_s(j)}{T|X_1|} \right| \geq \gamma l_s(\omega)^{\frac{3}{4}}\right] \leq \frac{\delta}{Q_1}$$

The intuition here is that, since the sampling is independent of any action of the adversary and any other action of the algorithm, the counters collected by the algorithm result from a representative sample of the opponent's actions, and therefore result in a reliable estimate of the number of times the opponent used the respective action. Technically, it is proved using the Azuma bound ([4]).

Proof. Let e^t_i be an indicator random variable for the event that in round t, the row i was explored by the algorithm, and let h^t_2 be a random variable that denotes the action of the opponent in round t. Note that

$$C^l_s(i, U_{ij}) - \frac{\beta F^l_s(j)}{T|X_1|} = \sum_{t=1}^{l} \left(e^t_i - \frac{\beta}{T|X_1|} \right) [h^t_2 = j]$$

Denoting

$$M_k = \sum_{t=1}^{k} \left(e^t_i - \frac{\beta}{T|X_1|} \right) [h^t_2 = j]$$

and defining $M_0 = 0$, we can see that the series $\{M_k\}^l_{k=0}$ is a martingale and $|M_k - M_{k-1}| < 1$ and therefore by applying Azuma's inequality ([4]) we have (for specific state, round and actions):

$$Pr\left[|M_k - M_0| \geq \gamma l_s(\omega)^{\frac{3}{4}}\right] \leq 2e^{-\frac{\gamma^2 l_s(\omega)^{\frac{3}{2}}}{2l_s(\omega)}} = 2e^{-\frac{\gamma^2\sqrt{l_s(\omega)}}{2}}$$

Therefore (for a specific round):

$$Pr\left[\begin{array}{l}\exists i \in X_1, \exists s \in S' \\ \exists \omega \in \Omega_s, \exists j \in \omega\end{array} : \left| C^l_s(i, U_{ij}) - \frac{\beta F^l_s(j)}{T|X_1|} \right| \geq \gamma l_s(\omega)^{\frac{3}{4}}\right] \leq$$

$$\leq 2|X_1||X_2||S|Te^{-\frac{\gamma^2\sqrt{l_s(\omega)}}{2}}$$

Summing over all rounds of the infinite game:

$$Pr\left[\begin{array}{l}\exists l \in \mathbb{N}, \exists i \in X_1 \\ \exists s \in S', \exists \omega \in \Omega_s, \exists j \in \omega\end{array} : \left| C^l(i, U_{ij}) - \frac{\beta F^l_s(j)}{T|X_1|} F^l_j \right| \geq\right.$$

$$\left.\gamma l_s(\omega)^{\frac{3}{4}}\right] \leq 2|X_1||X_2||S|T\sum_{l \in \mathbb{N}} e^{-\frac{\gamma^2\sqrt{l}}{2}}$$

[11] Those values are just placeholders for unknown values - we could use any impossible value here.

[12] Here we again make an optimistic assumption that payoffs yet unobserved are equal to U_{max}.

[13] Here we make an optimistic assumption that transitions yet unobserved lead to the "heaven" state 0.

It can be easily seen that there is $\gamma = poly(\log \frac{1}{\delta}, |X_1|, |X_2|, |S|, T)$ so that $\forall l \in \mathbb{N} : 2\ln l - \ln \frac{6\delta}{2\pi^2|X_1||X_2||S|TQ_1} \leq \frac{1}{2}\gamma^2\sqrt{l}$ and therefore:

$$2|X_1||X_2||S|T\sum_{l\in\mathbb{N}} e^{-\frac{\gamma^2\sqrt{l}}{2}} \leq$$

$$2|X_1||X_2||S|T\sum_{l\in\mathbb{N}} \frac{6\delta e^{-2\ln l}}{2\pi^2|X_1||X_2||S|TQ_1} =$$

$$= \frac{6\delta}{\pi^2}\sum_{l\in\mathbb{N}}\frac{1}{l^2} = \frac{\delta}{Q_1}$$

□

Given the above, from now on, we will assume as given that for all epochs in which the game is not yet fully known:

$$\begin{array}{l}\forall l \in \mathbb{N}, \forall i \in X_1 \\ \forall s \in S, \forall j \in \omega \in \Omega_s\end{array} : \left| C_s^l(i, U_{ij}) - \frac{\beta F_s^l(j)}{T|X_1|} \right| < \gamma l_s(\omega)^{\frac{3}{4}} \quad (1)$$

(that is, the negation of the inequality in the above lemma holds for all states and rounds of play in the epoch). Using this assumption we show that the algorithm achieves the required expected average payoff against any policy of the opponent with probability 1.

The following pair of Lemmas, show that (under the above assumption) the algorithm refines the information structure appropriately.

Lemma 2 (Sufficient condition for split)**.** *Given (1), if at any round $l \in \mathbb{N}$ in a given epoch there exist stage game s and two actions $y, y' \in \omega \in \Omega_s$ of the opponent such that $\left|F_s^l(y) - F_s^l(y')\right| > 4\gamma l_s(\omega)^{\frac{3}{4}}\frac{|X_1||X_2|}{\beta}$, then the algorithm must split ω in this round.*

The intuition here is that since the sampling process is representative, the counters collected in different rows for payoffs that result from the same (hidden) action by the adversary must have similar values. In particular, if the frequency with which the opponent used two of his actions is sufficiently different, the counters for the respective payoff values will have significantly different values – in all rows. Therefore, the algorithm can safely conclude that those payoff values result from distinct actions by the opponent.

Proof. Let us order the actions in ω in non-decreasing order of $F(y)_s^l$ (so that $F(y_1)_s^l \leq F(y_2)_s^l \leq \ldots \leq F(y_{|\omega|})_s^l$), then there must be two consecutive actions y_j, y_{j-1} between y and y' in this ordering so that $F(y_j)_s^l - F(y_{j-1})_s^l > 4\gamma l_s(\omega)^{\frac{3}{4}}\frac{|X_1|}{\beta}$. Using (1), we have that $\forall i \in X_1 : C_s^l(i, U_{iy_j}^s) - C_s^l(i, U_{iy_{j-1}}^s) > 2\gamma l_s(\omega)^{\frac{3}{4}}$ and therefore the algorithm must split ω. □

Lemma 3 (Split correctness)**.** *Given Eq. (1), the algorithm never makes a mistake in assigning the payoffs in the "split" phase. Formally, a mistake would mean that in partitioning ω into ω_1 and ω_2 at round l, there are two payoff values $u_1 \in \phi_s^l(i_1, \omega)$ and $u_2 \in \phi_s^l(i_2, \omega)$ that belong to the same column in the true game matrix (i.e. $\exists j \in X_2 : u_1 = U_{i_1 j}^s, u_2 = U_{i_2 j}^s$) and the algorithm assigns u_1 to $\phi_s^l(i_1, \omega_1)$ and u_2 to $\phi_s^l(i_2, \omega_2)$.*

The intuition here is that given the error margin asserted by Eq. 1, the algorithm cannot, while refining the partition, mistakenly assign a payoff value to a partition element that does not contain the respective opponent action. This is so, since the algorithm relies on the counter values when assigning the payoffs, and the counter values are representative of the actual opponent actions so far.

Proof. Assume by contradiction that a mistake was made and $u_1 = U_{i_1 j}^s$ and $u_2 = U_{i_2 j}^s$ are the wrongly assigned payoffs, then there must be two other payoffs $v_1 = U_{i_2 j'}^s$ and $v_2 = U_{i_1 j'}^s$ from another column j' that are assigned wrongly "in the other direction" - the algorithm assigns v_1 to $\phi_s^l(i_2, \omega_1)$ and v_2 to $\phi_s^l(i_1, \omega_2)$. This is so because if no such v_1, v_2 exist (i.e. $\forall j' \neq j : U_{i_1 j'}^s \in \phi_s^l(i_1, \omega_1) \vee U_{i_2 j'}^s \in \phi_s^l(i_2, \omega_2)$), then

$$|\omega| = |\phi_s^l(i_1, \omega_1)| + |\phi_s^l(i_2, \omega_2)| =$$

$$= 2 + \sum_{j'\neq j}[U_{i_1 j'}^s \in \phi_s^l(i_1, \omega_1)] + \sum_{j'\neq j}[U_{i_2 j'}^s \in \phi_s^l(i_2, \omega_2)] >$$

$$> \sum_{j'\neq j}[U_{i_1 j'}^s \in \phi_s^l(i_1, \omega_2)] + \sum_{j'\neq j}[U_{i_2 j'}^s \in \phi_s^l(i_2, \omega_1)] =$$

$$= |\phi_s^l(i_1, \omega_2)| + |\phi_s^l(i_2, \omega_1)| = |\omega|$$

, which is a contradiction. Given u_1, u_2, v_1, v_2 as above, the fact that there was a split implies that $C_s^l(i_1, u_2) > 2\gamma l^{\frac{3}{4}} + C_s^l(i_1, v_1)$ and $C_s^l(i_2, v_2) > 2\gamma l^{\frac{3}{4}} + C_s^l(i_2, u_1)$ By summing the inequalities, we have that

$$\left(C_s^l(i_1, u_2) - C_s^l(i_2, u_1)\right) + \left(C_s^l(i_2, v_2) - C_s^l(i_1, v_1)\right) > 4\gamma l^{\frac{3}{4}}$$

which implies that either $C_s^l(i_1, u_2) - C_s^l(i_2, u_1) > 2\gamma l^{\frac{3}{4}}$ or $C_s^l(i_2, v_2) - C_s^l(i_1, v_1) > 2\gamma l^{\frac{3}{4}}$ (or both). But $C_s^l(i_1, u_2) - C_s^l(i_2, u_1) > 2\gamma l^{\frac{3}{4}}$ implies $\left(C_s^l(i_1, u_2) - F_s^l(j)\right) + \left(F_s^l(j) - C_s^l(i_2, u_1)\right) > 2\gamma l^{\frac{3}{4}}$, which contradicts (1), and therefore we have a contradiction (the proof for the other inequality is symmetric). □

Lemma 4. *Suppose that in a given epoch of length l, in all stage games, for any two opponent strategies $j_1, j_2 \in \omega \in \Omega_s$ (which are in the same part of the partition Ω_s) in a given stage game $s \in S'$ it holds that $|F_s^l(j_1) - F_s^l(j_2)| \leq \frac{\epsilon}{4T|X_2|} l_s(\omega)$. Then the expected average payoff of the algorithm in this epoch is at least $\left(1 - \frac{\epsilon}{4}\right) V(T)$.*

The intuition here is that as long as the opponent uses some of his actions the same (roughly) amount of times, the fact that the algorithm cannot distinguish which payoff belongs to which action (in this set of actions) does not decrease its payoff – the assumption that the payoff for each of those actions is the numerical average of the set of payoffs works well enough.

Proof. Since throughout the epoch the algorithm uses the same T-step policy, we can assign an expected payoff value $V_s(j)$ (to the agent) to each action of the opponent $j \in X_2$ in each state $s \in S'$. We can also define "meta-actions" of the opponent, which correspond to the partition elements $\omega \in \Omega_s$ that are maintained by the algorithm. If the opponent used a perfectly balanced strategy in all stage games $s \in S'$ (that is, for any strategies $j_1, j_2 \in X_2$ that are in the same part of the partition Ω_s, $|F(j_1)_s^l - F(j_2)_s^l| = 0$), the expected average payoff obtained by the algorithm against each meta-action $V_s(\omega) = \frac{\sum_{j\in\omega} V_s(j) F_s^l(j)}{\sum_{j\in\omega} F_s^l(j)} = \frac{1}{|\omega|}\sum_{j\in\omega} V_s(j)$ is at least $V(T)$. This is so, because the algorithm acts under the assumption that the payoff of each meta-action is indeed the arithmetic average of the payoffs of the actions in it, and because this assumption implies a limitation on the strategies available to the opponent, while the optimal safety level strategy assumes nothing regarding the strategy of the opponent.

However, all we know from lemma's assumptions is that $\forall j \in X_2 : F_s^l(j) - F(1)_s^l \leq \frac{\epsilon}{4T|X_2|U_{max}} l_s(\omega)$. If we assume w.l.o.g that the opponent's actions are numbered in decreasing order of use (in particular, $1 \in \arg\max_{j\in\omega} F_s^l(j)$), then:

$$\sum_{j\in\omega} F_s^l(j) - |\omega| F(1)_s^l \leq \frac{\epsilon}{4T} l_s(\omega)$$

$$|\omega| F(1)_s^l \geq \left(1 - \frac{\epsilon}{4T}\right) \sum_{j\in\omega} F_s^l(j)$$

This implies that $F(1)_s^l > 0$, and:

$$\sum_{j\in\omega} F_s^l(j) \leq \frac{|\omega| F(1)_s^l}{1 - \frac{\epsilon}{4T}}$$

For a state that is reached on the last round of the T-step policy:

$$V_s(\omega) = \frac{\sum_{j\in\omega} V_s(j) F_s^l(j)}{\sum_{j\in\omega} F_s^l(j)} \geq \frac{F(1)_s^l \sum_{j\in\omega} V_s(j)}{\sum_{j\in\omega} F_s^l(j)}$$

Substituting the formula for $\sum_{j\in\omega} F_s^l(j)$ we have:

$$V_s(\omega) \geq \left(1 - \frac{\epsilon}{4T}\right) \frac{\sum_{j\in\omega} V_s(j)}{|\omega|}$$

,which means that the expected average payoff of the algorithm against each meta-action in this state is at least $(1 - \frac{\epsilon}{4T})$ times the expected average payoff of the optimal T-step strategy (in this state). For a state which is reached on the next-to-last round of the T-step policy, since the effective payoff of an action in it is the sum of the immediate payoff and the expected payoff in the state to which the action leads, the expected average payoff of the algorithm in this state is at least $(1 - \frac{\epsilon}{4T})^2$ times the expected average payoff of the optimal T-step strategy (in this state). By induction, we get that the total expected average payoff is at least:

$$\left(1 - \frac{\epsilon}{4T}\right)^T V(T) = \left(\left(1 - \frac{\epsilon}{4T}\right)^{\frac{4T}{\epsilon}}\right)^{\epsilon/4} V(T) =$$

$$= \left(\frac{1}{e}\right)^{\epsilon/4} V(T) = \left(\left(1 - \frac{\epsilon}{4}\right)^{4/\epsilon}\right)^{\epsilon/4} V(T) =$$

$$= \left(1 - \frac{\epsilon}{4}\right) V(T)$$

□

The following Lemmas deal with the situation where nothing is learned for a "long time", and show that in this case the agent will get high payoff. The proofs of these Lemmas are omitted due to lack of space, and appear in the full paper.
Let us denote $Q_2 = \left(4\gamma \frac{4T|X_1||X_2|^2}{\beta\epsilon}\right)^4$.

Lemma 5. *If, during some epoch, there is a stage game $s \in S'$ and two strategies $j_1, j_2 \in \omega \in \Omega_s$ (which are in the same part of the partition Ω_s) such that $l_s(\omega) > Q_2$ and $|F_s^l(j_1) - F_s^l(j_2)| > \frac{\epsilon}{4T|X_2|} l_s(\omega)$, then a split will occur (and the epoch will end).*

Proof. Immediate from lemma 2. □

Lemma 6. *Let l denote the length (in rounds) of an epoch. Suppose that $l \geq \frac{1}{1-\beta} \frac{4U_{max}}{\epsilon} Q_2 |S| T^2 |X_2|$ rounds, then the expected average payoff in this epoch is at least $(1-\beta)\left(1 - \frac{\epsilon}{2}\right) V(T)$.*

The intuition here is that, if the algorithm did not refine any of the partitions for a long time, then, for each partition element, the opponent must have used the different actions in this partition element a similar amount of times. The key observation is that the bound on the difference in frequency of use of the actions that is implied by Lemma (2) is $O(l_s(\omega)^{\frac{3}{4}}) < O(l_s(\omega))$, and therefore, if an epoch is longer than some polynomial, the relative difference in frequency of use (relative to the overall length of epoch) will become small enough for Lemma (4) to hold.

Proof. Let us refer to a single execution of a T-step policy during this epoch (which takes T rounds, unless it is interrupted by sampling) as a "loop". In expectation, the algorithm samples in β-fraction of the loops, therefore the expected number of uninterrupted loops in an epoch is at least $\frac{1-\beta}{T} l$ and we need to show that the expected average payoff in those loops is at least $\left(1 - \frac{\epsilon}{2}\right) V(T)$. By the pigeonhole principle, the maximal number of loops in which a state $s \in S'$ is visited and the opponent uses an action j from the partition element $\omega \in \Omega_s$ such that $l_s(\omega) \leq Q_2$ is $Q_2|S|T|X_2|$. Since we assume that a split did not occur, it follows from the previous lemma that in $n - Q_2|S|T|X_2|$ of the loops, for any state s and two strategies j_1, j_2 that are in the same partition element $\omega \in \Omega_s$ it holds that $|F(j_1)_s^l - F(j_2)_s^l| \leq \frac{\epsilon}{4T|X_2|} l_s(\omega)$. This implies, together with lemma 4, that the expected average payoff in those loops is at least $(1 - \epsilon/4) V(T)$. Therefore, the expected average payoff in the uninterrupted loops is

$$\frac{n - Q_2|S|T|X_2|}{n} \left(V(T) - \frac{\epsilon}{4}\right) \geq$$

$$\geq V(T) - \frac{\epsilon}{4} - U_{max} \frac{Q_2|S|T|X_2|\epsilon}{n} \geq V(T) - \epsilon/2$$

□

Let us denote $Q_3 = \frac{1}{1-\beta} \frac{4U_{max}}{\epsilon} Q_2 |S| T^2 |X_2|$.
Combining the above, we can now prove our main theorem:

Proof. Let us select $\beta = \epsilon/4$ – it follows that from the previous lemma that the expected average payoff of any epoch that is longer than Q_3 is at least $\left(1 - \frac{3\epsilon}{4}\right) V(T)$. Recall that there are at most Q_1 epochs and therefore the maximal total length of epochs that contain less than Q_3 rounds is $Q_1 Q_3$. This means that if the algorithm runs for at least $\hat{l} = \frac{4}{\epsilon} Q_1 Q_3$ rounds the expected average payoff is at least

$$\frac{\hat{l} - Q_1 Q_3}{\hat{l}} \left(1 - \frac{3\epsilon}{4}\right) V(T) =$$

$$= \left(1 - \frac{\epsilon}{4}\right) \left(1 - \frac{3\epsilon}{4}\right) V(T) \geq (1 - \epsilon) V(T)$$

□

5 Conclusions

This paper introduced the first polynomial near-optimal algorithm for obtaining the safety level value in any, initially unknown, multi-stage generic game, under imperfect monitoring. Our major restriction was that the game is genetic. The question of whether this restriction can be relaxed, is a subject for further study.

REFERENCES

[1] I. Ashlagi, D. Monderer, and M. Tennenholtz, 'Robust learning equilibrium', in *Proceedings of the 22nd Conference on Uncertainty in Artificial Intelligence (UAI 2006).*, (2006).

[2] Peter Auer, Nicolò Cesa-Bianchi, Yoav Freund, and Robert E. Schapire, 'The non-stochastic multi-armed bandit problem', *SIAM J. Comput.*, **32**, 48–77, (2002).

[3] R. Aumann and M. Maschler, *Repeated Games with Incomplete Information*, MIT Press, 1995.

[4] K. Azuma, 'Weighted sums of certain dependent random variables', *Tôhoku Math. Journal*, **19**, 357–367, (1967).

[5] A. Banos, 'On pseudo games', *The Annals of Mathematical Statistics*, **39**, 1932–1945, (1968).

[6] M. Bowling and M. Veloso, 'Rational and covergent learning in stochastic games', in *Proc. 17th IJCAI*, pp. 1021–1026, (2001).

[7] R. I. Brafman and M. Tennenholtz, 'R-max – a general polynomial time algorithm for near-optimal reinforcement learning', *Journal of Machine Learning Research*, **3**, 213–231, (2002).

[8] N. Hyafil and C. Boutilier, 'Regret minimizing equilibria and mechanisms for games with strict type uncertainty', in *Proceedings of the 20th Annual Conference on Uncertainty in Artificial Intelligence (UAI-04)*, pp. 268–277, Arlington, Virginia, (2004). AUAI Press.

[9] N. Megiddo, 'On repeated games with incomplete information played by non-bayesian players', *Int. J. of Game Theory*, **9**, 157–167, (1980).

[10] R. Powers and Y. Shoham, 'New Criteria and a New Algorithm for Learning in Multi-Agent Systems', in *NIPS 2004*, (2004).