

---

# Maximizing utility in multi-agent environments by anticipating the behavior of other learners

---

**Angelos Assos**  
MIT CSAIL  
Cambridge, MA  
assos@mit.edu

**Yuval Dagan**  
UC Berkeley  
Berkeley, CA  
yuvaldag@berkeley.edu

**Constantinos Daskalakis**  
MIT CSAIL  
Cambridge, MA  
costis@mit.edu

## Abstract

Learning algorithms are often used to make decisions in sequential decision-making environments. In multi-agent settings, the decisions of each agent can affect the utilities/losses of the other agents. Therefore, if an agent is good at anticipating the behavior of the other agents, in particular how they will make decisions in each round as a function of their experience thus far, it could try to judiciously make its own decisions over the rounds of the interaction so as to influence the other agents to behave in a way that ultimately benefits its own utility. In this paper, we study repeated two-player games involving two types of agents: a learner, which employs an online learning algorithm to choose its strategy in each round; and an optimizer, which knows the learner’s utility function and the learner’s online learning algorithm. The optimizer wants to plan ahead to maximize its own utility, while taking into account the learner’s behavior. We provide two results: a positive result for repeated zero-sum games and a negative result for repeated general-sum games. Our positive result is an algorithm for the optimizer, which exactly maximizes its utility against a learner that plays the Replicator Dynamics — the continuous-time analogue of Multiplicative Weights Update (MWU). Additionally, we use this result to provide an algorithm for the optimizer against MWU, i.e. for the discrete-time setting, which guarantees an average utility for the optimizer that is higher than the value of the one-shot game. Our negative result shows that, unless  $P=NP$ , there is no Fully Polynomial Time Approximation Scheme (FPTAS) for maximizing the utility of an optimizer against a learner that best-responds to the history in each round. Yet, this still leaves open the question of whether there exists a polynomial-time algorithm that optimizes the utility up to  $o(T)$ .

## 1 Introduction

With the increased use of learning algorithms as a means to optimize agents’ objectives in unknown or complex environments, it is inevitable that they will be used in multi-agent settings as well. This encompasses scenarios where multiple agents are repeatedly taking actions in the same environment, and their actions influence the payoffs of the other players. For example, participants in repeated online auctions use learning algorithms to bid, and the outcome, who gets to win and how much they pay, depends on all the bids (see e.g. [39, 40]). Other examples arise in contract design (see e.g. [30]),

Bayesian persuasion (see e.g. [19]), competing retailers that use learning algorithms to set their prices (see e.g. [25, 14]), and more.

It is thus natural to contemplate the following: in a setting where multiple agents take actions repeatedly, using their past observations of the other players' actions to decide on their future actions, what is the optimal strategy for an agent? This question is rather difficult, hence players often resort to simple online learning algorithms such as *mean-based* learners: those algorithms select actions that approximately maximize the performance on the past, and include methods such as MWU, Best Response Dynamics, FTRL, FTPL, etc; see e.g. [16]. Yet, when an agent knows that the other agents are using mean-based learners, it can it can adjust its strategy to take advantage of this knowledge. This was shown in specific games [9, 24]; however, in general settings it is not known how to best play against mean-based learners, even if the agent knows *everything* about the learners, and can predict with certainty what actions they would take as a function of the history of play. This raises the following question:

**Meta Question.** *In a multi-agent environment where agents repeatedly take actions, what is the best strategy for an agent, if it knows that the other agents are using mean-based learners to select their actions? Can it plan ahead if it can predict how the other agents will react? Can such an optimal strategy be computed efficiently?*

**Setting.** We study environments with two agents: a *learner* who uses a learning algorithm to decide on its actions, and an *optimizer*, that plans its actions, by trying to optimize its own reward, taking into account the learner's behavior. The setting is modeled as a repeated game: in each iteration  $t = 1, \dots, T$ , the optimizer selects a strategy  $x(t)$ , which is a distribution over a finite set of *pure actions*  $\mathcal{A} = \{a_1, \dots, a_n\}$ , i.e.  $x(t) \in \Delta(\mathcal{A})$ . At the same time, the learner selects a distribution  $y(t)$  over  $\mathcal{B} = \{b_1, \dots, b_m\}$ , i.e.  $y(t) \in \Delta(\mathcal{B})$ . The strategies  $x(t)$  and  $y(t)$  are viewed as elements of  $\mathbb{R}^n$  and  $\mathbb{R}^m$ , respectively, and the elements of  $\mathcal{A}$  and  $\mathcal{B}$  are identified with unit vectors. In each round, each player gains a utility, which is a function of the strategies played by both agents. The utilities for the optimizer and the learner equal  $x(t)^\top A y(t)$  and  $x(t)^\top B y(t)$ , respectively, where  $A, B \in \mathbb{R}^{n \times m}$ . The goal of each player is to maximize their own utility, which is summed over all rounds  $t = 1, \dots, T$ . We split our following discussion according to the study of zero-sum and general-sum games, as defined below.

**Zero-sum games.** Zero-sum games are those where  $A = -B$ . Namely, the learner and the optimizer have opposing goals. Such games have a *value*, which determines the best utility that the players can hope for, if they are both playing optimally. It is defined as:

$$\text{Val}(A) = \max_{x \in \Delta(\mathcal{A})} \min_{y \in \Delta(\mathcal{B})} x^\top A y = \min_{y \in \Delta(\mathcal{B})} \max_{x \in \Delta(\mathcal{A})} x^\top A y.$$

The two quantities in the definition above are equal, as follows from von Neumann's minmax theorem. The first of these quantities implies that the optimizer has a strategy that guarantees them a utility of  $\text{Val}(A)$  against any learner, and the second quantity implies that the learner has a strategy which guarantees that the optimizer's utility is at most  $\text{Val}(A)$ . Such strategies, namely, those that guarantee minmax value, are termed *minmax strategies*.

If the learner would have used a minmax strategy in each round  $t$ , then the optimizer could have only received a utility of  $T \text{Val}(A)$ . Yet, when the learner uses a learning algorithm, the optimizer can receive higher utility. A standard theoretical guarantee in online learning is *no-regret*. When a learner uses a *no-regret* algorithm to play a repeated game, it is guaranteed that the optimizer's reward after  $T$  rounds is at most  $T \text{Val}(A) + o(T)$ . Yet,  $T$  is finite, and the  $o(T)$ -term can be significant. Consequently, we pose the following question, which we study in Section 2.

**Question 1.** *In repeated zero-sum games between an optimizer and a learner, when can the optimizer capitalize on the sub-optimality of the learner's strategy, to obtain significantly higher utility than the value of the game? Can the optimizer's algorithm be computationally efficient?*

**General-sum games.** Games where  $B \neq -A$  are termed *general-sum*. These games are significantly more intricate: they do not posses a value or minmax strategies. In order to study such games, various notions of *game equilibria* have been proposed, such as the celebrated Nash Equilibrium. In the context related to the topic of our paper, if the optimizer could commit on a strategy, and the other player would best respond to it, rather than using a learning algorithm, then the optimizer could

get the *Stackelberg* value, by playing according to the *Stackelberg Equilibrium*. Further, there is a polynomial-time algorithm to compute the optimizer’s strategy in this case [21, 41, 20].

Against a learner that uses a no-regret learning algorithm, the optimizer could still guarantee the Stackelberg value of the game, by playing according to the Stackelberg Equilibrium. Yet, it was shown that in certain games, the optimizer can gain up to  $\Omega(T)$  more utility compared to the Stackelberg value when they are playing against mean-based learners [9, 24, 37, 30, 18]. However, this often requires playing a carefully-planned strategy which changes across time. While such a strategy could be computed in polynomial-time for specific games [9, 13, 23], no efficient algorithm was known for general games. Deng et al. [24] devised a control problem whose solution is approximately the optimal strategy for the optimizer, against mean-based learners. Yet, they do not provide a computationally-efficient algorithm for this control problem. Finding an optimal strategy for the learner was posed as an open problem [24, 30, 11], that can be summarized as follows:

**Question 2.** *In repeated general-sum games between an optimizer and a mean-based learner, is there a polynomial-time algorithm to find an optimal strategy for the optimizer?*

We study this question in Section 3.

### 1.1 Our results

Our results are two-fold; In zero sum games, we provide positive results, and show what the optimizer’s optimal rewards and strategies for a given game should be, in both discrete and continuous time games. Our results in that realm are the following:

- For continuous time games, we provide an exact, closed form solution of the rewards of the optimizer against a learner that uses the replicator dynamics, i.e. continuous MWU (Section 2, Theorem 1). In the same theorem we prove that the optimizer can achieve optimal rewards by playing a constant strategy throughout the game. i.e.  $x(t) = x^*, x \in \Delta(\mathcal{A}) \forall t \in [0, T]$ .
- We also prove a range where the optimal rewards might be and provide a lower bound for when  $\eta T \rightarrow \infty$  (Section 2, Proposition 1).
- For discrete time games, when the optimizer is up against a learner that uses MWU, we first prove that the optimal rewards of the optimizer in this setting, will be lower bounded by the rewards of the optimizer in the continuous time game, if they use the same strategy  $x^*$  (Section 2 Proposition 2).
- We then prove that the reward ‘gap’ between continuous and discrete time games cannot be greater than  $\eta T/2$  (Section B, Proposition 10), and there are games that achieve a gap of at least  $\Omega(\eta T)$  (Section 2, Proposition 3). In fact we give a class of games that can achieve that gap in rewards (Section 2, Proposition 4).

In general-sum games, we provide the first known computational lower bound for calculating the optimal strategy against a mean based learner. We formalize the problem of optimizing rewards as a decision problem, where the answer for an instance is ‘YES’ if the optimizer can achieve rewards more than  $T$  and ‘NO’ if the optimizer can receive rewards at most  $T - 1$ . We prove that there is no polynomial time algorithm, assuming  $P \neq NP$  that distinguishes between the two cases by using a reduction from Hamiltonian Cycle. The formal theorem and a sketch of the reduction construction can be found in Section 2 Theorem 4.

### 1.2 Related Work

The game-theoretic modeling of multi-agent environments has long history [16, 45, 27]. Such studies often revolve around proving that if multiple agents use learning algorithms to repeatedly play against each other, the average history of play converges to various notions of game equilibria [43, 16]. Recent works have shown that some learning algorithms yield fast convergence to game equilibria and fast-decaying regret, if all players are using the same algorithm [44, 22, 2, 4, 26, 42, 46].

**Optimizing against no regret learners.** Braverman et al. [9] initiated a recent thread of works, which studies how a player can take advantage of the learning algorithms run by other agents in order to gain high utility. They showed that in various repeated auctions where a seller sells an item to a single buyer in each iteration  $t = 1, \dots, T$ , the seller can gain nearly the maximal utility that they

could possibly hope for, if the learner runs an EXP3 learning algorithm (i.e., the seller's utility can be arbitrarily close to the total welfare). This can be  $\Omega(T)$  larger than what they can get if the buyer is strategic. Deng et al. [24] generalized the study to arbitrary normal-form games. They showed an example for a game where an optimizer that plays against any mean-based learner gets  $\Omega(T)$  more than the Stackelberg value, which is what they would get against strategic agents. The same paper further showed that against no-swap-regret learners, the optimizer cannot gain  $\Omega(T)$  more than the Stackelberg value and Mansour et al. [37] showed that no-swap-regret learners are the only algorithms with this property. Brown et al. [11] showed a polynomial time algorithm to compute the optimal strategy for the optimizer against the no-regret learner that is most favorable to the optimizer — yet, such no-regret learner may be unnatural. Additional work obtained deep insights into the optimizer/learner interactions in general games [11, 5], in Bayesian games [37] and in specific games such as auctions [23, 13, 35, 34], contract design [30] and Bayesian persuasion [18].

**Optimizing against MWU in  $2 \times 2$ -games.** Guo and Mu [29] obtain a computationally-efficient algorithm for an optimizer that is playing a zero-sum game against a learner that uses MWU, in games where each player holds 2 actions, and they analyze that optimal strategy.

**Regret lower bounds for online learners.** Regret lower bounds for learning algorithms are tightly related to upper bounds for an optimizer in zero-sum games. The optimizer can be viewed as an adversary that seeks to maximize the regret of the algorithm. The main difference is that these lower bounds construct worst-case instances, yet, we seek to maximize the utility of the optimizer in any game. Regret lower bounds for MWU and generalizations of it were obtained by [17, 32, 15, 28]. Minmax regret lower bounds against any online learner can be found, for example, in [16].

**Approximating the Stackelberg value.** Lastly, the problem of computing the Stackelberg equilibrium is well studied [21, 41, 20]. Recent works also study repeated games between an optimizer and a learner, where the optimizer does not know the utilities of the learner and its goal is to learn to be nearly as good as the Stackelberg strategy [6, 38, 36, 31].

## 2 Optimizing utility in zero-sum games

**Continuous-time games.** We begin our discussion with continuous-time dynamics, where  $x(t)$  and  $y(t)$  are functions of the continuous-time parameter  $t \in [0, T]$ . The total reward for the optimizer is  $\int_0^T x(t)^\top A y(t) dt$ , whereas the learner's utility equals the optimizer's utility multiplied by  $-1$ . We assume that the learner is playing the *Replicator Dynamics* with parameter  $\eta > 0$ , which is the continuous-time analogue of the Multiplicative Weights Update algorithm, which plays at every time  $t$ :

$$y_i(t) = \frac{\exp\left(\eta \int_0^t x(s)^\top B e_i ds\right)}{\sum_{j=1}^m \exp\left(\eta \int_0^t x(s)^\top B e_j ds\right)}, \quad i = 1, \dots, m. \quad (1)$$

This formula gives a higher weight to actions that would have obtained higher utility in the past. When  $\eta$  is larger, the learner is more likely to play the actions that maximize the past utility. The value of  $\eta$  usually becomes smaller as  $T$  increases, and typically  $\eta \rightarrow 0$  as  $T \rightarrow \infty$ .

The following definition will be helpful for the analysis:

**Definition 1** (Historical Rewards for the learner). *The historical rewards for the learner in continuous games at time  $t$ , denoted by  $h(t) \in \mathbb{R}^m$ , is an  $m$  dimensional vector where  $h_i(t), i = 1, \dots, m$  corresponds to the sum of rewards achieved by action  $a_i$  of the learner against the history of play in the game so far. We assume a general setting where the learner at  $t = 0$  might have non-zero historical rewards, i.e.  $h(0) \in \mathbb{R}^m$ . If the strategy of the optimizer is  $x : [0, t] \rightarrow \Delta(\mathcal{A})$  we get:*

$$h(t) = h(0) + \int_0^t B^\top x(t) dt$$

Suppose the game has been played for some time  $t$  and the learner has collected historical rewards  $h(t)$ . The total reward that the optimizer gains from the remainder of this game can be written as a

function of the time left for the game  $T - t$ , the historical rewards of the learner at time  $t$ ,  $h(t)$ , and the strategy  $x : [0, T - t] \rightarrow \Delta(\mathcal{A})$  of the optimizer for the remainder of the game.

$$R_{cont}(x, h(t), T - t, A, B) = \int_0^{T-t} \frac{\sum_{i=1}^m e^{\eta(h_i(u) + e_i^\top B^\top \int_0^u x(s) ds)} \cdot e_i^\top A^\top x(u)}{\sum_{i=1}^m e^{\eta(h_i(u) + e_i^\top B^\top \int_0^u x(s) ds)}} du \quad (2)$$

For simplicity, we can just transfer the time to 0 and assume that the historical rewards of the learner are just  $h(0)$ . That way we can rewrite the above definition as:

$$R_{cont}(x, h(0), \tau, A, B) = \int_0^\tau \frac{\sum_{i=1}^m e^{\eta(h_i(t) + e_i^\top B^\top \int_0^t x(s) ds)} \cdot e_i^\top A^\top x(t)}{\sum_{i=1}^m e^{\eta(h_i(t) + e_i^\top B^\top \int_0^t x(s) ds)}} dt \quad (3)$$

In general we are interested in finding the maximum value that the optimizer can achieve at any moment of the game:

$$R_{cont}^*(h(0), \tau, A, B) = \max_x R_{cont}(x, h(0), \tau, A, B) \quad (4)$$

For finding the optimal reward for the optimizer from the beginning of the game we would have to find  $R_{cont}^*(\mathbf{0}, T, A, B)$ , where  $\mathbf{0} = (0, 0, \dots, 0)^\top$ .

The next theorem characterizes the exact optimal strategy for the optimizer, against a learner that uses the Replicator Dynamics in zero sum games:

**Theorem 1.** *In a zero-sum continuous game, when the learner is using the Replicator Dynamics, the optimal rewards for the optimizer can be achieved with a constant strategy throughout the game, i.e.  $x(t) = x^* \in \Delta(\mathcal{A})$ . The optimal reward is obtained by the following formula:*

$$R_{cont}^*(h(0), T, A, -A) = \max_{x \in \Delta(\mathcal{A})} \left\{ \frac{\ln \left( \sum_{i=1}^m e^{\eta h_i(0)} \right) - \ln \left( \sum_{i=1}^m e^{\eta(h_i(0) - T e_i^\top A^\top x)} \right)}{\eta} \right\} \quad (5)$$

Further,  $x^*$  is the maximizer in the formula above.

*Proof.* For zero sum games in continuous time, we have that equation 3 becomes:

$$R_{cont}(x, h(0), T, A, -A) = \int_0^T \frac{\sum_{i=1}^m e^{\eta(h_i(0) - e_i^\top A^\top \int_0^t x(s) ds)} \cdot e_i^\top A^\top x(t)}{\sum_{i=1}^m e^{\eta(h_i(0) - e_i^\top A^\top \int_0^t x(s) ds)}} dt \quad (6)$$

Notice that

$$\frac{d}{dt} \left( e^{\eta(h_i(0) - e_i^\top A^\top \int_0^t x(s) ds)} \right) = e^{\eta(h_i(0) - e_i^\top A^\top \int_0^t x(s) ds)} \cdot \frac{d}{dt} \left( \eta h_i(0) - \eta \cdot e_i^\top A^\top \int_0^t x(s) ds \right)$$

From Leibniz rule we have that  $\frac{d}{dt} \int_0^t x(s) ds = x(t)$ , thus we get:

$$\frac{d}{dt} \left( e^{\eta(h_i(0) - e_i^\top A^\top \int_0^t x(s) ds)} \right) = -e^{\eta(h_i(0) - e_i^\top A^\top \int_0^t x(s) ds)} \cdot \eta \cdot e_i^\top A^\top x(t)$$

Given that, note that we can find a closed form solution for  $R_{cont}(x, h(0), T, A, -A)$  as follows:

$$\begin{aligned} R_{cont}(x, h(0), T, A, -A) &= \left[ -\frac{1}{\eta} \ln \left( \sum_{i=1}^m e^{\eta(h_i(0) - e_i^\top A^\top \int_0^t x(s) ds)} \right) \right]_0^T \\ &= \frac{\ln \left( \sum_{i=1}^m e^{\eta h_i(0)} \right) - \ln \left( \sum_{i=1}^m e^{\eta(h_i(0) - e_i^\top A^\top \int_0^T x(s) ds)} \right)}{\eta} \end{aligned}$$

Suppose the optimal rewards  $R_{cont}^*(h(0), T, A, -A)$  are achieved with  $x^{opt}(t)$ . Note that the final reward only depends on  $\int_0^T x(s) ds$ , thus the same reward that is achieved by  $x^{opt}(t)$ , can be achieved by  $x^* = \frac{1}{T} \int_0^T x^{opt}(s) ds$ . Thus there exists a  $x^*$  such that:

$$R_{cont}^*(h(0), T, A, -A) = \frac{\ln \left( \sum_{i=1}^m e^{\eta h_i(0)} \right) - \ln \left( \sum_{i=1}^m e^{\eta(h_i(0) - e_i^\top A^\top x^* T)} \right)}{\eta} \quad (7)$$

which is what we wanted.  $\square$

The optimal strategy  $x^*$  for the optimizer in continuous zero-sum games can be therefore obtained by finding the minimum of the convex function  $f(x) = \ln \left( \sum_{i=1}^m e^{\eta(h_i(0) - e_i^\top A^\top x T)} \right)$ . We can compute the optimal strategy of the optimizer in an efficient way using techniques from convex optimization. More details can be found in the appendix at Proposition 5.

We further analyze how larger the optimal achievable reward is, compared to the naive bound of  $T \text{Val}(A)$ . We specifically show that always the optimal rewards of the optimizer are in the range of  $[T \text{Val}(A), T \text{Val}(A) + \log(m)/\eta]$ , where  $m$  is the number of actions of the learner, showing the optimizer can always get more utility than just the value of the game.

We also analyze what happens to the rewards of the optimizer as  $\eta T \rightarrow \infty$ . First, let us define for any optimizer's strategy  $x$ , the set of all best-responses,

$$\text{BR}(x) = \arg \max_{b \in \mathcal{B}} x^\top B b.$$

This defines a set, and if there are multiple maximizers,  $|\text{BR}(x)| > 1$ . We then have the following proposition:

**Proposition 1 (informal).** *The optimal reward for an optimizer playing against a learner that uses the replicator dynamics with parameter  $\eta$  in an  $n \times m$  game, is in the range  $[T \text{Val}(A), T \text{Val}(A) + \log(m)/\eta]$ . Further, as  $\eta T \rightarrow \infty$ , this optimal utility is at least*

$$T \text{Val}(A) + \frac{\log(m/k)}{\eta}, \quad \text{where } k = \min_{x \in \Delta(\mathcal{A}) \text{ minmax strategy}} |\text{BR}(x)|.$$

The proof of Proposition 1 can be found in Appendix B, Propositions 6 and 7. We note that the limiting utility is obtained by playing constantly any minmax strategy  $x$  that attains the minimum in the definition of  $k$  above.

**Connection to optimal control and the Hamilton-Jacobi-Bellman equation.** We will present another way of achieving Theorem 1, using literature from control theory. One can view the problem of maximizing the optimizer's utilities as an optimal control problem; what control (or strategy) should the optimizer use in order to maximize their utility given that the learner has some specific dynamics that depend on the control of the optimizer? The Hamilton-Jacobi-Bellman equation [8] gives us a partial differential equation (PDE) of  $R_{\text{cont}}^*(h, t, A, B)$  that if we solve, we can find a closed form solution of the optimal utility of the optimizer. The equation (for general sum games):

$$-\frac{dR_{\text{cont}}^*(h, t, A, B)}{dt} = \max_{x \in \Delta(\mathcal{A})} \left( \frac{\sum_{i=1}^m e^{\eta h_i} \cdot e_i^\top A^\top x}{\sum_{i=1}^m e^{\eta h_i}} + (\nabla_h R_{\text{cont}}^*(h, t, A, B))^\top \cdot B^\top x \right)$$

The intuition of the PDE is as follows; the current state of the learner can be defined given only the history  $h$  of the sum of what the optimizer played so far, and the time left in the game  $t$ . Given we are at a state  $h, t$ , the optimal rewards for the optimizer are going to be equal to the rewards of playing action  $x \in \Delta(\mathcal{A})$  for time  $\Delta t$  added together with the optimal reward in the new state, namely  $R_{\text{cont}}^*(h + B^\top x \Delta t, t + \Delta t, A, B)$ . Taking the limit as  $\Delta t \rightarrow 0$ , gives us the above partial differential equation.

Plugging in  $B = -A$ , for the case of zero-sum games, we get:

$$-\frac{dR_{\text{cont}}^*(h, t, A, -A)}{dt} = \max_{x \in \Delta(\mathcal{A})} \left( \frac{\sum_{i=1}^m e^{\eta h_i} \cdot e_i^\top A^\top x}{\sum_{i=1}^m e^{\eta h_i}} - (\nabla_h R_{\text{cont}}^*(h, t, A, -A))^\top \cdot A^\top x \right)$$

If one plugs in the formula we calculated in Theorem 1, they would find that indeed it is a solution to the above PDE.

**Discrete-time games.** We now move to the discrete-time setting. The learner is assumed to be playing the Multiplicative-Weights update algorithm, defined by:

$$y_i(t) = \frac{\exp \left( \eta \sum_{s=0}^{t-1} x(s)^\top B e_i \right)}{\sum_{j=1}^m \exp \left( \eta \sum_{s=1}^{t-1} x(s)^\top B e_j \right)}, \quad i = 1, 2, \dots, m. \quad (8)$$

We show that if the learner constantly plays the strategy  $x^*$  that is optimal for continuous-time, the obtained utility against MWU can only be higher, compared to playing against the Replicator Dynamics in continuous-time, with the same step size  $\eta$ :

**Proposition 2** (informal). *Let  $A$  be a zero-sum game matrix and  $\eta > 0$  and  $T \in \mathbb{N}$ . Let  $x^* \in \Delta(\mathcal{A})$  be the optimal strategy against the replicator dynamics, from Theorem 1. Then, the utility in the discrete time, achieved by an optimizer which plays  $x(t) = x^*$  for all  $t \in \{1, \dots, T\}$  against MWU with parameter  $\eta$ , is at least the utility achieved in the continuous-time by playing  $x^*$  against the replicator dynamics with the same parameters  $\eta, T$ .*

The proof can be found in Appendix B (Proposition 8).

Proposition 2 implies that Proposition 1 provides lower bounds on the achievable utility of this constant discrete-time strategy. In order to further analyze its performance, we would like to ask how much larger the discrete-time utility of the optimizer can be from the optimal continuous-time utility. We include the following statement, which follows from a standard approach:

**Proposition 3.** *Let  $T \in \mathbb{N}$  and  $\eta > 0$ . For any zero-sum game whose utilities are bounded in  $[-1, 1]$ , the best discrete-time optimizer obtains a utility of at most  $\eta T/2$  more than the best continuous-time optimizer. Further, there exists a game where the best discrete-time optimizer achieves a utility of  $\tanh(\eta)T/2 = (\eta - O(\eta^2))T/2$  more than the optimal continuous-time optimizer, and  $\tanh(\eta)T/2$  more than the discrete-time optimizer from Proposition 2.*

The proof can be found in Appendix B (Propositions 9 and 10).

Proposition 3 implies that in some cases, the best discrete-time optimizer can gain  $T \text{Val}(A) + \Omega(\eta T)$ . We would like to understand in which games this is possible for any choice of  $\eta \in (0, 1)$ . We provide a definition that guarantees that this would be possible:

**Condition 1.** *There exists a minmax strategy  $x$  for the optimizer such that there exist two best responses for the learner,  $b_{i_1}, b_{i_2} \in \text{BR}(x)$ , which do not coincide on  $\text{support}(x)$ . Namely, there exists an action  $a_k \in \text{support}(x)$  such that  $a_k^\top A b_{i_1} \neq a_k^\top A b_{i_2}$ .*

To motivate Condition 1, notice that in order to achieve a gain of  $\Omega(\eta T)$  for any  $\eta$ , the discrete optimizer has to be significantly better than the continuous-time optimizer. For that to happen the learner would have to change actions frequently. Indeed, the difference between the discrete and continuous learners is that the discrete learner is slower to change actions: they only change actions at integer times, whereas the continuous learner could change actions at each real-valued time. In order to change actions, they need to have at least two good actions, and this is what Assumption 1 guarantees. We derive the following statement:

**Proposition 4.** *For any zero-sum game  $A \in \mathbb{R}^{n \times m}$  that satisfies Assumption 1,  $\eta \in (0, 1)$  and  $T \in \mathbb{N}$ , there exists an optimizer, such that against a learner that uses MWU with step size  $\eta$ , achieves a utility of at least  $T \text{Val}(A) + \Omega(\eta T)$ , where the constant possibly depends on  $A$ .*

The proof can be found in Appendix B (Proposition 11).

Proposition 4 considers a strategy for the optimizer which takes the minmax strategy  $x$  from Assumption 1, and splits into two strategies:  $x', x'' \in \Delta(\mathcal{A})$ , such that  $(x' + x'')/2 = x$ . Further,  $x'^\top A b_{i_1} \neq x''^\top A b_{i_2}$ , where  $b_{i_1}, b_{i_2}$  are defined in Assumption 1. It plays  $x'$  in each odd round  $t$ , and  $x''$  in each even round. It is possible to show that in each two consecutive iterations, the reward for the optimizer is  $2 \text{Val}(A) + \Omega(\eta)$ . Summing over  $T/2$  consecutive pairs yields the final bound.

### 3 A computational lower bound for optimization in general-sum games

In this section, we present the first limitation on optimizing against a mean-based learner. Specifically, we study the algorithm which is termed *Best-Response* or *Fictitious Play* [10, 43]. At each time step  $t = 1, \dots, T$ , this algorithm selects an action  $y(t)$  that maximizes the cumulative utility for rounds  $1, \dots, t-1$ :

$$y(t) = \arg \max_{y \in \Delta(\mathcal{B})} \sum_{s=1}^{t-1} x(s)^\top B y \quad (9)$$

There is always a maximizer which corresponds to a pure action  $b_i \in \mathcal{B}$  of the learner and we assume that if there are ties, they are broken lexicographically. In this section, we constrain the optimizer to also play pure actions.

To put this algorithm in context, we note the following connections to general mean-based learners: (1) *mean-based* learners are any algorithms which select an action  $y(t)$  that approximately maximizes Eq. (9); (2) Best-Response is equivalent to MWU with  $\eta \rightarrow \infty$ .

We prove that there is no algorithm that approximates the optimal utility of the optimizer up to an approximation factor of  $1 - \epsilon$ , whose runtime is polynomial in  $n, m, T$  and  $1/\epsilon$ , as exemplified by the following (informal) Theorem:

**Theorem 2** (informal). *Let  $\text{Alg}$  be an algorithm that receives parameters  $\epsilon > 0, m, n, T \in \mathbb{N}$  and utility matrices  $A, B$  of dimension  $m \times n$  and entries in  $[0, 1]$ , and outputs a control sequence  $x(1), \dots, x(T) \in \mathcal{A}$ . Let  $\bar{U}$  denote the utility attained by the learner and let  $U^*$  denote the optimal possible utility. If  $U \geq (1 - \epsilon)U^*$  for any instance of the problem, and if  $\text{P} \neq \text{NP}$ , then  $\text{Alg}$  is not polynomial-time in  $m, n, T$  and  $1/\epsilon$ .*

A sketch of the proof can be found below, and a full proof can be found in the Appendix C. The proof is obtained via a reduction from the Hamiltonian cycle problem. We note two limitations of this result: (1) The lower bound is for  $T = n/2 + 1$ , and it shows hardness in distinguishing between the case that the optimal reward is  $T$  and the case that the optimal reward is at most  $T - 1$ . It is still open whether one could efficiently find a sequence that optimizes the reward up to  $o(T)$ ; (2) fictitious-play is a fundamental online learning algorithm. Yet, it does not possess the no-regret guarantee. It is still open whether one could obtain maximal utility against no-regret learners such as MWU with a small step size.

We continue by framing the problem of maximizing rewards against a Best-Response learner as a decision problem, called OCDP:

**Problem 1** (Optimal Control Discrete Pure (OCDP)). *An OCDP instance is defined by  $(A, B, n, m, k, T)$ , where  $n, m, k, T \in \mathbb{N}$ ,  $A \in \{0, 1\}^{n \times m}$  and  $B \in [0, 1]^{n \times m}$ . The numbers  $n$  and  $m$  correspond to the actions of the optimizer and learner in a game, where  $A$  is the utility matrix of the optimizer and  $B$  is the utility matrix of the learner. This instance is a 'YES' instance if the optimizer can achieve utility at least  $k$  after playing the game for  $T$  rounds with a learner that uses the Best Response Algorithm (Eq. (9)), and 'NO' if otherwise.*

We will prove that OCDP is NP-hard, using a reduction from the Hamiltonian cycle problem.

**Problem 2** (Hamiltonian cycle). *Given a directed graph  $G(V, E)$ , find whether there exists a Hamiltonian cycle, i.e. a cycle that starts from any vertex, visits every vertex exactly once and closes at the same vertex where it started.*

It is a known theorem that the Hamiltonian Cycle is an NP-complete problem, as it is one of Karp's 21 initial NP-Complete problems.

**Theorem 3** ([33]). *Hamiltonian Cycle is NP-complete.*

We conclude with the main result of this section, followed by a proof sketch. The full proof appears in Section C.

**Theorem 4.** *OCDP is NP-hard. That is, if  $\text{P} \neq \text{NP}$ , there exists no algorithm that runs in polynomial time in  $n, m$  and  $k$  which distinguishes between the case that a reward of  $k$  is achievable and the case that it is impossible to obtain reward more than  $k - 1$ .*

*Proof sketch.* Consider an instance of the Hamiltonian cycle problem:  $\pi_H = (V, E)$ , where  $V = \{v_1, \dots, v_n\}$ ,  $E = \{e_1, \dots, e_m\}$ . We create an instance of OCDP as follows. First, set  $T = k = n + 1$ . We will construct the instance such that the optimizer can receive reward  $n + 1$  if and only if there is a Hamiltonian cycle in the graph. Define the optimizer's actions to be  $\{a_1, \dots, a_m\}$ , and the learner's actions to be  $\{b_1, \dots, b_n, b'_1, \dots, b'_n\}$ . Namely, for each node  $v_i$  of the graph, the learner has two associated actions,  $b_i$  and  $b'_i$ . The details in the sketch differ slightly from the proof for clarity. The reduction is constructed to satisfy the following properties:

- The only way for the optimizer to receive maximal utility, is to play a strategy as follows: the first  $n$  actions should correspond to edges  $e_{i_1} - e_{i_2} - \dots - e_{i_n}$  that form a Hamiltonian cycle which starts and ends at the vertex  $v_1$ ; and the last action corresponds to  $e_{i_{n+1}}$  which is an outgoing edge from  $v_1$ .

- We define the utility matrix for the learner such that, if the optimizer is playing according to this strategy, then the learner's actions will correspond to the vertices of the same cycle, denoted as  $v_{j_1} - \dots - v_{j_n} - v_{j_1}$ . This is achieved by defining the learner's tie-breaking rule to play  $a_1$  in the first round; and defining the learner's utilities such that, if the optimizer has played in rounds  $1, \dots, t-1$  the edges along a path  $v_{j_1} - \dots - v_{j_t}$ , then the best response for the learner at time  $t$  would be to play  $a_{j_t}$ . To achieve this, we define for any edge  $e_k = (v_p, v_q)$ :  $B[a_k, b_p] = -1$ ,  $B[a_k, b_q] = 1$  and  $B[a_k, b] = 0$  for any other action  $b$ . Consequently, after observing the edges along  $b_{j_1} - \dots - b_{j_t}$ , the cumulative reward for the learner would be 1 for  $b_{j_t}$ , -1 for  $b_{j_1}$  and 0 for any other  $b_j$ . Consequently, the learner will best respond with  $b_{j_t}$  — we ignore the actions  $b'_1, \dots, b'_n$  of the learner at the moment.
- In order to guarantee that the above optimizer's strategy yields a utility of  $n+1$ , we define the optimizer's utility such that  $A[e_k, b_p] = 1$  if  $e_k = (v_p, v_q)$  and for all  $q \neq p$   $A[e_k, b_q] = 0$ . This will also force the optimizer to play edges that form a path. Indeed, if the optimizer has previously played the edges along a path  $v_{j_1} - \dots - v_{j_t}$ , then, as we discussed, the learner will play  $b_{j_t}$  in the next round. For the optimizer to gain a reward at the next round, they must play an edge outgoing from  $a_{j_t}$ . This preserves the property that the optimizer's actions form a path.
- Next, we would like to guarantee that the optimizer's actions align with a Hamiltonian cycle. Therefore, we need to ensure that the path played by the optimizer does not close too early. Namely, if the learner has played the edges along the path  $v_{j_1} - \dots - v_{j_t}$  and if  $t < n$ , then  $j_1, \dots, j_t$  are all distinct. For this purpose, the actions  $v'_1, \dots, v'_n$  are defined. We define for any edge  $e_k = (v_i, v_j)$ :  $B[a_k, b'_i] = 0.85$ . This will prevent the optimizer from playing the same vertex twice, for any round  $t = 1, \dots, n$  (recall that there are  $T = n+1$  rounds), as argued below.

Assume for the sake of contradiction that the first time the same vertex is visited is at  $t$ , where  $j_t = j_r$  for some  $r < t \leq n$ . Then, the cumulative utility of action  $b'_{j_r}$  for rounds  $1, \dots, t$  is 1.7, which is larger than any other action. This implies that at the next round, the learner will play action  $b'_{j_r}$ . We define the utility for the optimizer to be zero against any action  $b'_j$ . Hence, any scenario where the learner plays an action  $b'_j$ , prevents the optimizer from receiving a utility of  $n+1$ . This happens whenever  $i_t = i_j$  for some  $j < t \leq n$ . Hence, an optimizer that receives a reward of 1 at any round must play a path that does not visit the same vertex twice, in rounds  $t = 1, \dots, n$ .

- Lastly, notice that we do want the optimizer to play the same vertex  $a_1$  twice, at rounds 1 and  $n+1$ . This does not prevent the optimizer from receiving optimal utility. Indeed, if the optimizer would play the action  $a_1$  twice, then the learner would play  $b'_1$  at the next round. Yet, since there are only  $n+1$  rounds, there is no “next round” and no reward is lost. The details that force the learner to play  $b_1$  in round  $n+1$  appear in the full version of the proof.

The above explanation sketches why it is possible to get  $n+1$  reward if and only if there is a Hamiltonian cycle and this concludes the reduction.  $\square$

## 4 Conclusion and future directions

In this paper we studied how an optimizer should play in a two-player repeated game knowing that the other agent is a learner that is using a known mean-based algorithm. In zero-sum games we showed how they can gain optimal utility against the Replicator Dynamics and we further analyzed the utility that they could gain against MWU. In general sum games, we showed the first computational hardness result on optimizing against a mean-based learner, by reduction from Hamiltonian Cycle.

One interesting problem that remains is the open is analyzing the optimal reward in general sum games against the Replicator Dynamics (or the MWU), which was denoted as  $R_{cont}^*(\mathbf{0}, T, A, B)$ . In the fully general case with no restriction on the utility matrices  $A$  and  $B$ , we believe there is no closed form solution for the optimal utility, differently from the zero-sum case. However, it would be interesting to understand how  $R_{cont}^*(\mathbf{0}, T, A, B)$  behaves as a function of  $A$  and  $B$  and how the best strategy for the optimizer looks like. A conjecture in this direction was given by Deng et al. [24]. Perhaps it would be easier to study simpler scenarios, such as the one where  $\text{rank}(A + B) = 1$ ,

which has been explored in the context of computing equilibria and the convergence of learning algorithms to them ([1], [3]). Another direction is improving on the lower bound for general sum games. Currently, we prove that it is hard to distinguish between the case where the optimizer can achieve reward  $\alpha = T$  and the case where the optimizer cannot achieve more than  $\beta = T - 1$ . Is it also hard to distinguish between a reward of at least  $\alpha$  or at most  $\beta$  in cases where  $\alpha - \beta = \Omega(T)$ ? Are there lower bounds when the learner uses different learning algorithms, such as MWU? Other relevant open directions are extensions to multi-agent settings [13], analyzing how the learner's utility is impacted by interaction with the optimizer in general-sum games [30], which learning algorithms yield higher utilities against an optimizer, and what algorithms should be used to both learn and optimize at the same time?

**Societal impact.** The work studies multi-agent environments and how agents can benefit by anticipating the behavior of other agents. We believe that increasing the academic knowledge in this topic can help learning agents assess their risk of being utilized by other agents and can help to build algorithms that are more resilient to manipulation. As always with new technologies, there is a risk that malicious players will utilize ideas from this paper, which could cause a harmful effect on other agents.

## References

- [1] Bharat Adsul, Jugal Garg, Ruta Mehta, Milind Sohoni, and Bernhard Von Stengel. Fast algorithms for rank-1 bimatrix games. *Operations Research*, 69(2):613–631, 2021.
- [2] Ioannis Anagnostides, Constantinos Daskalakis, Gabriele Farina, Maxwell Fishelson, Noah Golowich, and Tuomas Sandholm. Near-optimal no-regret learning for correlated equilibria in multi-player general-sum games. In *ACM Symposium on Theory of Computing*, 2022.
- [3] Ioannis Anagnostides, Gabriele Farina, Ioannis Panageas, and Tuomas Sandholm. Optimistic mirror descent either converges to nash or to strong coarse correlated equilibria in bimatrix games. *Advances in Neural Information Processing Systems*, 35:16439–16454, 2022.
- [4] Ioannis Anagnostides, Ioannis Panageas, Gabriele Farina, and Tuomas Sandholm. On last-iterate convergence beyond zero-sum games. In *International Conference on Machine Learning*, 2022.
- [5] Eshwar Ram Arunachaleswaran, Natalie Collina, and Jon Schneider. Pareto-optimal algorithms for learning in games. *arXiv preprint arXiv:2402.09549*, 2024.
- [6] Maria-Florina Balcan, Avrim Blum, Nika Haghtalab, and Ariel D. Procaccia. Commitment without regrets: Online learning in stackelberg security games. In *Proceedings of the Sixteenth ACM Conference on Economics and Computation*, pages 61–78, 2015.
- [7] Amir Beck. *First-order methods in optimization*. SIAM, 2017.
- [8] Richard Bellman. Dynamic programming and a new formalism in the calculus of variations. *Proceedings of the national academy of sciences*, 40(4):231–235, 1954.
- [9] Mark Braverman, Jieming Mao, Jon Schneider, and Matt Weinberg. Selling to a no-regret buyer. In *Proceedings of the 2018 ACM Conference on Economics and Computation*, pages 523–538, 2018.
- [10] George W Brown. Iterative solution of games by fictitious play. *Act. Anal. Prod Allocation*, 13(1):374, 1951.
- [11] William Brown, Jon Schneider, and Kiran Vodrahalli. Is learning in games good for the learners? *Advances in Neural Information Processing Systems*, 36, 2024.
- [12] Sébastien Bubeck et al. Convex optimization: Algorithms and complexity. *Foundations and Trends® in Machine Learning*, 8(3-4):231–357, 2015.
- [13] Linda Cai, S Matthew Weinberg, Evan Wildenhain, and Shirley Zhang. Selling to multiple no-regret buyers. In *International Conference on Web and Internet Economics*, pages 113–129. Springer, 2023.
- [14] Emilio Calvano, Giacomo Calzolari, Vincenzo Denicolo, and Sergio Pastorello. Artificial intelligence, algorithmic pricing, and collusion. *American Economic Review*, 110(10):3267–3297, 2020.
- [15] Nicolo Cesa-Bianchi. Analysis of two gradient-based algorithms for on-line regression. In *Proceedings of the tenth annual conference on Computational learning theory*, pages 163–170, 1997.
- [16] Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.
- [17] Nicolo Cesa-Bianchi, Yoav Freund, David Haussler, David P Helmbold, Robert E Schapire, and Manfred K Warmuth. How to use expert advice. *Journal of the ACM (JACM)*, 44(3):427–485, 1997.
- [18] Yiling Chen and Tao Lin. Persuading a behavioral agent: Approximately best responding and learning. *arXiv preprint arXiv:2302.03719*, 2023.
- [19] Yiling Chen and Tao Lin. Persuading a behavioral agent: Approximately best responding and learning. *arXiv preprint arXiv:2302.03719*, 2023.

- [20] Natalie Collina, Eshwar Ram Arunachaleswaran, and Michael Kearns. Efficient stackelberg strategies for finitely repeated games. In *Proceedings of the 2023 International Conference on Autonomous Agents and Multiagent Systems*, pages 643–651, 2023.
- [21] Vincent Conitzer and Tuomas Sandholm. Computing the optimal strategy to commit to. In *Proceedings of the 7th ACM Conference on Electronic Commerce*, pages 82–90, 2006.
- [22] Constantinos Daskalakis, Maxwell Fishelson, and Noah Golowich. Near-optimal no-regret learning in general games. *Advances in Neural Information Processing Systems*, 34:27604–27616, 2021.
- [23] Yuan Deng, Jon Schneider, and Balasubramanian Sivan. Prior-free dynamic auctions with low regret buyers. *Advances in Neural Information Processing Systems*, 32, 2019.
- [24] Yuan Deng, Jon Schneider, and Balasubramanian Sivan. Strategizing against no-regret learners. *Advances in neural information processing systems*, 32, 2019.
- [25] Heng Du and Tiaojun Xiao. Pricing strategies for competing adaptive retailers facing complex consumer behavior: Agent-based model. *International Journal of Information Technology & Decision Making*, 18(06):1909–1939, 2019.
- [26] Gabriele Farina, Ioannis Anagnostides, Haipeng Luo, Chung-Wei Lee, Christian Kroer, and Tuomas Sandholm. Near-optimal no-regret learning dynamics for general convex games. In *Neural Information Processing Systems (NeurIPS)*, 2022.
- [27] S. Freund and L. Gittins. Strategic learning and teaching in games. *Econometrica*, 66(3):597–625, 1998.
- [28] Nick Gravin, Yuval Peres, and Balasubramanian Sivan. Tight lower bounds for multiplicative weights algorithmic families. In *44th International Colloquium on Automata, Languages, and Programming (ICALP 2017)*. Schloss-Dagstuhl-Leibniz Zentrum für Informatik, 2017.
- [29] Xinxiang Guo and Yifen Mu. The optimal strategy against hedge algorithm in repeated games. *arXiv preprint arXiv:2312.09472*, 2023.
- [30] Guru Guruganesh, Yoav Kolumbus, Jon Schneider, Inbal Talgam-Cohen, Emmanouil-Vasileios Vlatakis-Gkaragkounis, Joshua R Wang, and S Matthew Weinberg. Contracting with a learning agent. *arXiv preprint arXiv:2401.16198*, 2024.
- [31] Nika Haghtalab, Thodoris Lykouris, Sloan Nietert, and Alexander Wei. Learning in stackelberg games with non-myopic agents. In *Proceedings of the 23rd ACM Conference on Economics and Computation*, pages 917–918, 2022.
- [32] David Haussler, Jyrki Kivinen, and Manfred K Warmuth. Tight worst-case loss bounds for predicting with expert advice. In *European Conference on Computational Learning Theory*, pages 69–83. Springer, 1995.
- [33] Richard M Karp. *Reducibility among combinatorial problems*. Springer, 2010.
- [34] Yoav Kolumbus and Noam Nisan. Auctions between regret-minimizing agents. In *Proceedings of the ACM Web Conference 2022*, pages 100–111, 2022.
- [35] Yoav Kolumbus and Noam Nisan. How and why to manipulate your own agent: On the incentives of users of learning agents. In *Advances in Neural Information Processing Systems*, volume 35, pages 28080–28094, 2022.
- [36] Niklas Lauffer, Mahsa Ghasemi, Abolfazl Hashemi, Yagiz Savas, and Ufuk Topcu. No-regret learning in dynamic stackelberg games. In *arXiv preprint arXiv:2203.17184*, 2022.
- [37] Yishay Mansour, Mehryar Mohri, Jon Schneider, and Balasubramanian Sivan. Strategizing against learners in bayesian games. In *Conference on Learning Theory*, pages 5221–5252. PMLR, 2022.

- [38] Janusz Marecki, Gerry Tesauro, and Richard Segal. Playing repeated stackelberg games with unknown opponents. In *Proceedings of the 11th International Conference on Autonomous Agents and Multiagent Systems - Volume 2*, pages 821–828, 2012.
- [39] Denis Nekipelov, Vasilis Syrgkanis, and Eva Tardos. Econometrics for learning agents. In *Proceedings of the sixteenth ACM conference on economics and computation*, pages 1–18, 2015.
- [40] Gali Noti and Vasilis Syrgkanis. Bid prediction in repeated auctions with learning. In *Proceedings of the Web Conference 2021*, pages 3953–3964, 2021.
- [41] Binghui Peng, Weiran Shen, Pingzhong Tang, and Song Zuo. Learning optimal strategies to commit to. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 2149–2156, 2019.
- [42] Georgios Piliouras, Ryann Sim, and Stratis Skoulakis. Beyond time-average convergence: Near-optimal uncoupled online learning via clairvoyant multiplicative weights update. *Advances in Neural Information Processing Systems*, 35:22258–22269, 2022.
- [43] Julia Robinson. An iterative method of solving a game. *Annals of mathematics*, pages 296–301, 1951.
- [44] Vasilis Syrgkanis, Alekh Agarwal, Haipeng Luo, and Robert E. Schapire. Fast convergence of regularized learning in games. In *Advances in Neural Information Processing Systems*, volume 28, 2015.
- [45] J. Youn. Learning algorithms in games. *Journal of Game Theory*, 56(2):123–134, 2004.
- [46] Brian Hu Zhang, Gabriele Farina, Ioannis Anagnostides, Federico Cacciamani, Stephen Marcus McAleer, Andreas Alexander Haupt, Andrea Celli, Nicola Gatti, Vincent Conitzer, and Tuomas Sandholm. Computing optimal equilibria and mechanisms via learning in zero-sum extensive-form games. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

## A Additional Definitions

We introduce the following definitions that will help us in the future proofs.

**Definition 2** (Historical Rewards for the learner). *The historical rewards for the learner in continuous games at time  $t$ , denoted by  $h(t) \in \mathbb{R}^m$ , is an  $m$  dimensional vector that corresponds to the sum of rewards achieved by each action of the learner against the history of play in the game so far. We assume a general setting where the learner at  $t = 0$  might have non-zero historical rewards, i.e.  $h(0) \in \mathbb{R}^m$ . If the strategy of the optimizer is  $x : [0, t] \rightarrow \Delta(\mathcal{A})$  we get:*

$$h(t) = h(0) + \int_0^t B^\top x(t) dt$$

*In the discrete setting, given that the optimizer has played  $x_0, x_1, \dots, x_{t-1} \in \Delta(\mathcal{A})$ , we have for  $t \geq 1$ :*

$$h(t) = h(0) + \sum_{i=0}^{t-1} B^\top x_i$$

**Definition 3** (Value of the game). *In a zero sum case, where  $B = -A$  we define the value of the game for the optimizer as  $\text{Val}(A)$ :*

$$\text{Val}(A) = \min_{y \in \Delta(\mathcal{B})} \max_{x \in \Delta(\mathcal{A})} x^\top A y$$

**Definition 4** (Best Response Set). *For a given strategy of the optimizer  $x \in \Delta(\mathcal{A})$ , denote by  $\text{BR}(x)$  the set of best responses by the learner:*

$$\text{BR}(x) = \left\{ e_i \mid i \in \operatorname{argmin}_{i \in [m]} [x^\top A e_i] \right\}$$

**Definition 5** (Min-max strategies). *Denote with  $\text{MinMaxStrats}_{\text{opt}}(A)$  the set of strategies of the optimizer that achieve the value of the game.*

$$\text{MinMaxStrats}_{\text{opt}}(A) = \left\{ x \in \Delta(\mathcal{A}) \mid \text{Val}(A) = \min_{i \in [m]} x^\top A e_i \right\}$$

### A.1 Algorithms for the learner

For completeness, we provide pseudocode for the algorithms.

**Definition 6** (MWU Algorithm). *Fix the step size  $0 \leq \eta \leq \frac{1}{2}$ . In a game where the utility matrix of the learner is  $B$ , the MWU algorithm is as follows:*

---

#### Algorithm 1 MWU algorithm

---

```

1: procedure MWU( $B, T, \eta$ )
2:    $h(t) = (0, 0, \dots, 0)^\top$ 
3:   for  $t = 1, 2, \dots, T$  do
4:      $y \leftarrow \left( \frac{e^{\eta \cdot h_1(t)}}{\sum_{i=1}^m e^{\eta \cdot h_i(t)}}, \dots, \frac{e^{\eta \cdot h_m(t)}}{\sum_{i=1}^m e^{\eta \cdot h_i(t)}} \right)$ 
5:     Submit  $y$ 
6:     Observe  $x_t$ 
7:      $h(t+1) = h(t) + B^\top x_t$ 
8:   end for
9: end procedure

```

---

**Definition 7** (Replicator Dynamics). *Suppose  $x : [0, T] \rightarrow \Delta(\mathcal{A})$  is a strategy for the optimizer. If the learner is using the replicator dynamics, the strategy for the learner at time  $t$  is given by:*

$$y_i(t) = \frac{\exp \left( \eta \int_0^t x(s)^\top B e_i \right)}{\sum_{j=1}^n \exp \left( \eta \int_0^t x(s)^\top B e_j \right)}, i = 1, 2, \dots, m. \quad (10)$$

**Definition 8** (Best Response Algorithm). *In a game where the utility matrix of the learner is  $B$ , the best response algorithm is as follows, assuming we break ties lexicographically (i.e. if  $h_i(t) = h_j(t)$ , then  $b_i$  beats  $b_j$  if  $i < j$ ):*

---

**Algorithm 2** Best Response algorithm

---

```

1: procedure BRA( $B, T$ )
2:    $h(0) = (0, 0, \dots, 0)^\top$ 
3:   for  $t = 0, 1, \dots, T - 1$  do
4:      $y \leftarrow b_{\arg\max_{i \in [m]} h_i(t)}$ 
5:     Submit  $y$ 
6:     Observe  $x_t$ 
7:      $h(t + 1) = h(t) + B^\top x$ 
8:   end for
9: end procedure

```

---

## B Optimization in zero-sum games: missing proofs

In this section, we present some omitted proofs of lemmas mentioned in the main section.

### B.1 Continuous games

First, we prove that the optimal strategy for the optimizer in continuous games can be computed efficiently. From theorem 1 we know that the optimal strategy for the optimizer throughout the game, is just  $x(t) = x^*$ , where  $x^*$  minimizes  $f(x)$ :

$$f(x) = \ln \left( \sum_{i=1}^m e^{\eta(h_i(0) - T e_i^\top A^\top x)} \right)$$

The above is also known as the log-sum-exp function. This is a convex function, which allows us to compute its minima efficiently. We use the following two lemmas.

**Lemma 1** (Example 5.15 [7]). *The log-sum-exp function  $f : \mathbb{R}^n \rightarrow \mathbb{R}$ , for which  $f(\mathbf{x}) = \ln(\sum_{i=1}^m e^{x_i})$ , is 1-smooth with respect to the  $\ell_2, \ell_\infty$  norms and convex.*

**Lemma 2** (Theorem 3.8, [12]). *For a function  $f : \mathcal{X} \rightarrow \mathbb{R}$  that is convex and  $\beta$ -smooth with respect to a norm  $\|\cdot\|$  and where  $R = \sup_{x,y} \|x - y\|$ , the Frank-Wolfe algorithm can find  $x_t$  on  $O(t)$  time for which:*

$$f(x_t) - f(x^*) \leq \frac{2\beta R^2}{t + 1}$$

We can therefore compute the approximate optimal strategy for the optimizer efficiently:

**Proposition 5.** *For zero sum continuous games where the learner is using replicator dynamics, we can find an  $\epsilon$  approximate optimal strategy for the optimizer in  $O(\frac{1}{\epsilon\eta})$  time. That is we can find  $x$  for which:*

$$R_{cont}^*(\mathbf{0}, T, A, -A) - R_{cont}(x, \mathbf{0}, T, A, -A) \leq \epsilon$$

*Proof.* Consider the function  $g(x) = \ln \left( \sum_{i=1}^m e^{\eta(h_i(0) - T e_i^\top A^\top x)} \right)$ . From 1, we also know that  $g$  is 1-smooth with respect to the  $\ell_\infty$  norm. Note that since the strategies are on the simplex, we have that  $R = \sup_{x,y} \|x - y\| = 1$ . Using lemma 2, with  $t = \frac{2}{\epsilon\eta} - 1$  we can find  $x$  for which  $g(x) - g(x^*) \leq \epsilon\eta$ . Using theorem 1, we get:

$$R_{cont}^*(\mathbf{0}, T, A, -A) - R_{cont}(x, \mathbf{0}, T, A, -A) = \frac{g(x) - g(x^*)}{\eta} \leq \epsilon$$

as required.  $\square$

Next, we bound the range of the possibilities for the utility gained by the best optimizer:

**Proposition 6.** *In the zero-sum continuous setting, the optimizer's optimal utility is bounded as follows:*

$$\text{Val}(A) \cdot T \leq R_{\text{cont}}^*(\mathbf{0}, T, A, -A) \leq \text{Val}(A) \cdot T + \frac{\ln m}{\eta} \quad (11)$$

where  $\text{Val}(A)$  is as defined in 3.

*Proof.* We have:

$$\begin{aligned} R_{\text{cont}}^*(\mathbf{0}, T, A, -A) &= \frac{1}{\eta} \left( \ln(m) - \min_x \ln \left( \sum_{i=1}^m e^{-\eta \cdot e_i^\top A^\top x T} \right) \right) = \\ &= \frac{1}{\eta} \left( \ln(m) - \min_{x \in \Delta(\mathcal{A})} \max_{j \in \{1, 2, \dots, m\}} \ln \left( e^{-\eta e_j^\top A^\top x T} \sum_{i=1}^m e^{\eta(e_j^\top - e_i^\top) A^\top x T} \right) \right) = \\ &= \frac{1}{\eta} \left( \ln(m) - \min_{x \in \Delta(\mathcal{A})} \max_{j \in \{1, 2, \dots, m\}} \left( -\eta e_j^\top A^\top x T + \ln \left( \sum_{i=1}^m e^{\eta(e_j^\top - e_i^\top) A^\top x T} \right) \right) \right) = \\ &= \frac{1}{\eta} \left( \ln(m) + \max_{x \in \Delta(\mathcal{A})} \min_{j \in \{1, 2, \dots, m\}} \left( \eta e_j^\top A^\top x T - \ln \left( \sum_{i=1}^m e^{\eta(e_j^\top - e_i^\top) A^\top x T} \right) \right) \right) \end{aligned}$$

Note that picking out the  $j$  for which  $e_j^\top A^\top x \leq e_i^\top A^\top x$  for all  $i \in [m]$ , makes  $e^{\eta(e_j^\top - e_i^\top) A^\top x T} \leq 1$  for all  $i = 1, 2, \dots, m$ . On the other hand  $e^{\eta(e_j^\top - e_j^\top) A^\top x T} = 1$ . That gives  $\ln 1 = 0 \leq \ln \left( \sum_{i=1}^m e^{\eta(e_j^\top - e_i^\top) A^\top x T} \right) \leq \ln m$ . We finally have:

$$\max_{x \in \Delta(\mathcal{A})} \min_{j \in \{1, 2, \dots, m\}} e_j^\top A^\top x T \leq R_{\text{cont}}^*(\mathbf{0}, T, A, -A) \leq \max_{x \in \Delta(\mathcal{A})} \min_{j \in \{1, 2, \dots, m\}} e_j^\top A^\top x T + \frac{\ln m}{\eta} \quad (12)$$

which is exactly what we want.  $\square$

If we take the limit of the game as  $\eta T$  goes to infinity, we can lower bound the optimal rewards of the optimizer by considering a strategy from the set  $\text{MinMaxStrats}(A)$  as defined in 5, that has the least number of best responses.

**Proposition 7.** *As  $\eta T \rightarrow \infty$ ,*

$$\begin{aligned} R_{\text{cont}}^*(\mathbf{0}, T, A, -A) &\geq \text{Val}(A) \cdot T + \frac{1}{\eta} \left( \ln(m) - \ln \left( \min_{x^* \in \text{MinMaxStrats}_{\text{opt}}(A)} |\text{BR}(x^*)| \right) (1 + o_{\eta T}(1)) \right), \end{aligned}$$

where  $o_{\eta T}(1)$  denotes a terms that decays to 0 as  $\eta T \rightarrow \infty$ , and  $\text{BR}(x)$  is the set of best responses of a strategy  $x$  of the optimizer as defined in 4.

*Proof.* Fix some  $x \in \Delta(\mathcal{A})$ . Note that from Theorem 6 we have:

$$R_{\text{cont}}^*(\mathbf{0}, T, A, -A) \geq \frac{\ln(m)}{\eta} + \left( e_j^\top A^\top x T - \frac{1}{\eta} \ln \left( \sum_{i=1}^m e^{\eta(e_j^\top - e_i^\top) A^\top x T} \right) \right)$$

for any  $j \in [m]$ . Choose  $j \in \text{BR}(x)$  (thus it minimizes  $e_j^\top A^\top x$ ). Denote with  $d_i(j, x) = (e_j - e_i)^\top A^\top x$ . Note that for all  $i \in [m] / \text{BR}(x)$  we have  $d_i(j, x) = 0$  and for  $i \notin \text{BR}(x)$  we have

$d_i(j, x) < 0$ . We can rewrite this as:

$$\begin{aligned}
R_{cont}^*(\mathbf{0}, T, A, -A) &\geq \frac{\ln(m)}{\eta} + \left( e_j^\top A^\top x T - \frac{1}{\eta} \ln \left( |\text{BR}(x)| + \sum_{i \in [m] / \text{BR}(x)} e^{\eta T \cdot d_i(j, x)} \right) \right) \\
&= \frac{\ln(m)}{\eta} + \\
&+ \left( e_j^\top A^\top x T - \frac{1}{\eta} \ln(|\text{BR}(x)|) + \frac{1}{\eta} \ln \left( 1 + \frac{\sum_{i \in [m] / \text{BR}(x)} e^{\eta T \cdot d_i(j, x)}}{|\text{BR}(x)|} \right) \right) = \\
&= \frac{\ln(m)}{\eta} + \\
&+ \left( e_j^\top A^\top x T - \frac{1}{\eta} \ln(|\text{BR}(x)|) \left( 1 + \frac{\ln \left( 1 + \frac{\sum_{i \in [m] / \text{BR}(x)} e^{\eta T \cdot d_i(j, x)}}{|\text{BR}(x)|} \right)}{\ln(|\text{BR}(x)|)} \right) \right)
\end{aligned}$$

Note that as  $\eta T \rightarrow \infty$ , the term  $\frac{\ln \left( 1 + \frac{\sum_{i \in [m] / \text{BR}(x)} e^{\eta T \cdot d_i(j, x)}}{|\text{BR}(x)|} \right)}{\ln(|\text{BR}(x)|)}$  goes to 0, as  $d_i(j, x) < 0$ . Therefore this gives :

$$R_{cont}^*(\mathbf{0}, T, A, -A) \geq \frac{\ln(m)}{\eta} + \left( e_j^\top A^\top x T - \frac{1}{\eta} \ln(|\text{BR}(x)|)(1 + o_{\eta T}(1)) \right)$$

Choosing  $x \in \text{MinMaxStrats}_{opt}(A)$  that minimizes  $|\text{BR}(x)|$  yields the desired result.  $\square$

Below we provide an example of a game with multiple min-max strategies for the optimizer, however only one of them gives optimal rewards when used against MWU.

**Example 1.** Consider the zero sum game where:

	$b_1$	$b_2$	$b_3$	$\dots$	$b_n$	$b_{n+1}$	$b_{n+2}$	$b_{n+3}$
$a_1$	$n$	0	0	$\dots$	0	$n$	$n$	1
$a_2$	0	$n$	0	$\dots$	0	$n$	$n$	1
$a_3$	0	0	$n$	$\dots$	0	$n$	$n$	1
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\dots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$
$a_n$	0	0	0	$\dots$	$n$	$n$	$n$	1
$a_{n+1}$	$n$	$n$	$n$	$\dots$	$n$	2	0	1
$a_{n+2}$	$n$	$n$	$n$	$\dots$	$n$	0	2	1

Notice that the  $\text{Val}(A) = 1$ , and there are multiple strategies that achieve this result, that might have a different number of best responses. For example  $x_1 = (\frac{1}{n}, \frac{1}{n}, \dots, \frac{1}{n}, 0, 0)$  has  $n+1$  best responses;  $\text{BR}(x_1) = \{b_1, b_2, \dots, b_{n+3}\}$ . One of the strategies that achieves the value of the game with the least number of best responses is  $x^* = (0, 0, \dots, \frac{1}{2}, \frac{1}{2}, 0)$  that has only 1 best response, namely  $\text{BR}(x^*) = \{b_{n+3}\}$ . For this game playing  $x^*$  as  $\eta T \rightarrow \infty$  yields  $R_{cont}^*(\mathbf{0}, 0, A, -A) = T + \frac{\ln(n+3)}{\eta}$

## B.2 Discrete Case

We can have the analogous definitions of optimal rewards for the optimizer for the discrete case. Given the sequence of actions of the optimizer given by  $x : \{1, \dots, T\} \rightarrow \Delta(\mathcal{A})$ , we can define:

$$R_{disc}(x, h(0), T, A, B) = \sum_{t=1}^T \frac{\sum_{i=1}^m e^{\eta(h_i(0) + e_i^\top B^\top \sum_{s=1}^t x(s))} \cdot e_i^\top A^\top x(t)}{\sum_{i=1}^m e^{\eta(h_i(0) + e_i^\top B^\top \sum_{s=1}^t x(s))}} \quad (13)$$

Again, we are interested in finding the sequence of actions that maximize the rewards:

$$R_{disc}^*(h(0), T, A, B) = \max_{\substack{x(t) \\ 0 \leq t \leq T}} \sum_{t=1}^T \frac{\sum_{i=1}^m e^{\eta(h_i(0) + e_i^\top B^\top \sum_{s=1}^t x(s))} \cdot e_i^\top A^\top x(t)}{\sum_{i=1}^m e^{\eta(h_i(0) + e_i^\top B^\top \sum_{s=1}^t x(s))}} \quad (14)$$

We begin by proving that the rewards for the discrete game will always be more than the rewards for the analogous continuous game.

**Proposition 8.** *In the zero-sum discrete game, the optimizer can receive more utility compared to the continuous game. This can be done by playing discretely the same optimal strategy  $x^*$  for the continuous game (as defined in Theorem 1). Consequently,*

$$R_{disc}^*(\mathbf{0}, T, A, -A) \geq R_{cont}^*(\mathbf{0}, T, A, -A) \quad (15)$$

*Proof.* From Theorem 1 we know that  $R_{cont}^*(\mathbf{0}, 0, A, -A)$  is achieved by a constant strategy  $x^*$  for the optimizer throughout the game. We will prove that using that same strategy  $x^*$  will yield more reward for the discrete case than the continuous case, i.e.:

$$\begin{aligned} R_{disc}^*(\mathbf{0}, 0, A, -A) &\geq \sum_{t=0}^{T-1} \frac{\sum_{i=1}^m e^{-\eta e_i^\top A^\top x^* t} \cdot e_i^\top A^\top x^*}{\sum_{i=1}^m e^{-\eta e_i^\top A^\top x^* t}} \\ &\geq \int_{t=0}^T \frac{\sum_{i=1}^m e^{-\eta e_i^\top A^\top x^* t} \cdot e_i^\top A^\top x^*}{\sum_{i=1}^m e^{-\eta e_i^\top A^\top x^* t}} dt = R_{cont}^*(\mathbf{0}, 0, A, -A) \end{aligned}$$

All we need for the above to be true is that the function  $f(t) = \frac{\sum_{i=1}^m e^{-\eta e_i^\top A^\top x^* t} \cdot e_i^\top A^\top x^*}{\sum_{i=1}^m e^{-\eta e_i^\top A^\top x^* t}}$  is non-increasing. This is because  $R_{disc}(\mathbf{0}, 0, A, -A) - R_{cont}(\mathbf{0}, 0, A, -A) = \sum_{t=0}^{T-1} f(t) - \int_0^T f(t) dt = \sum_{t=0}^{T-1} (f(t) - \int_t^{t+1} f(s) ds)$ , so by proving  $f$  is non-increasing we can prove  $f(t) \geq \int_t^{t+1} f(s) ds$ . Therefore, all we are left with is showing that  $\frac{df(t)}{dt} < 0$  for all  $t \geq 0$ . Notice that:

$$f'(t) = -\frac{1}{\eta} \left( \frac{\sum_{i=1}^m e^{-c_i t} \cdot c_i^2}{\sum_{i=1}^m e^{-c_i t}} + \left( \frac{\sum_{i=1}^m e^{-c_i t} \cdot c_i}{\sum_{i=1}^m e^{-c_i t}} \right)^2 \right) < 0 \quad (16)$$

which concludes the proof.  $\square$

We continue by proving that there are instances of discrete games, that achieve reward for the optimizer approximately  $\frac{\eta T}{2}$  more than the analogous continuous game in Theorem 9. We also prove that this difference between rewards in continuous and discrete games cannot be greater than  $\frac{\eta T}{2}$  in Theorem 10.

**Proposition 9.** *There is a zero sum game instance for which we have that*

$$R_{disc}^*(\mathbf{0}, T, A, -A) \geq R_{cont}^*(\mathbf{0}, T, A, -A) + \frac{\tanh(\eta) \cdot T}{2}$$

*The above is achieved by taking the optimal strategy  $x^*$  for the continuous game (as defined in 1) and playing it over discrete time.*

*Proof.* Consider the matching pennies problem, where the utility matrices are:

$$A = \begin{array}{|c|c|c|} \hline & b_1 & b_2 \\ \hline a_1 & 1 & -1 \\ \hline a_2 & -1 & 1 \\ \hline \end{array}, B = \begin{array}{|c|c|c|} \hline & b_1 & b_2 \\ \hline a_1 & -1 & 1 \\ \hline a_2 & 1 & -1 \\ \hline \end{array}$$

Note that  $\text{Val}(A) = 0$ , and  $\text{MinMaxStrats}_{opt}(A) = \{(\frac{1}{2}, \frac{1}{2})\}$ , and  $|\text{BR}((\frac{1}{2}, \frac{1}{2}))| = 2$ . This gives  $R_{cont}^*(\mathbf{0}, T, A, -A) = 0$ . On the other hand, consider the sequence of actions of the optimizer for the discrete game to be as follows:

$$x(t) = \begin{cases} a_2, & \text{if } t \text{ is odd} \\ a_1, & \text{otherwise} \end{cases}$$

We claim that at round  $t$  where  $t$  is odd, the learner will play  $(\frac{1}{2}, \frac{1}{2})$ , while when  $t$  is even, the learner will play  $(\frac{e^\eta}{e^\eta + e^{-\eta}}, \frac{e^{-\eta}}{e^\eta + e^{-\eta}})$ . Indeed, at  $t = 1, 3, 5, \dots$ , the optimizer will have played equal times the action  $a_1$  and  $a_2$ , thus the rewards for actions  $b_1$  and  $b_2$  so far will be equal, leading the learner to play  $(\frac{1}{2}, \frac{1}{2})$ . When  $t$  is even, action  $a_2$  has been played one more time than  $a_1$  is played. That means that at time  $t$  for  $b_1$  the historical reward is 1, while the historical reward for  $b_2$  is  $-1$ . Thus the strategy for the learner for even  $t$  will indeed be  $(\frac{e^\eta}{e^\eta + e^{-\eta}}, \frac{e^{-\eta}}{e^\eta + e^{-\eta}})$ . Therefore the optimizer's reward will be:

$$R_{disc}(x(t), \mathbf{0}, T, A, -A) = \frac{T}{2} \cdot [0 \ 1] \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} \frac{1}{2} \\ \frac{1}{2} \end{bmatrix} + \frac{T}{2} \cdot \begin{bmatrix} \frac{e^\eta}{e^\eta + e^{-\eta}} & \frac{e^{-\eta}}{e^\eta + e^{-\eta}} \end{bmatrix} \begin{bmatrix} 1 & -1 \\ -1 & 1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \\ = 0 + \frac{T}{2} \left( \frac{e^\eta}{e^\eta + e^{-\eta}} - \frac{e^{-\eta}}{e^\eta + e^{-\eta}} \right) = \frac{T}{2} \cdot \tanh(\eta) \approx \frac{\eta T}{2} \quad (18)$$

Combining the two gives the desired result.  $\square$

**Proposition 10.** *In a zero-sum game where the entries of  $A$  satisfy  $A_{ij} \in [-1, 1]$ , we have that the optimal utility the optimizer can achieve in the discrete game is not more than  $\eta T/2$  compared to the continuous game:*

$$R_{disc}^*(\mathbf{0}, T, A, -A) - R_{cont}^*(\mathbf{0}, T, A, -A) \leq \eta T/2.$$

*Proof.* Suppose the optimal strategy for the discrete game is  $x_{disc}$ . We will construct a strategy  $x_{cont}$  for the continuous game, that approximates the utility of the discrete game pretty well. The strategy is as follows; if  $x_{disc}(t) = x_t \in \Delta(\mathcal{A})$ , then  $x_{cont}(s) = x_t, \forall a \in [t, t+1]$ . Note that the historical rewards of the learner for  $t = 0, 1, 2, \dots, T-1$  are the same for both the continuous and the discrete game for these specific strategies. Suppose the history at time  $t$  is  $h(t)$ , the strategy being played at time  $t$  be  $x$ , and  $u_i = e_i^\top A^\top x$ . We have:

$$R_{disc}^*(h(t), 1, A, -A) = \frac{\sum_{i=1}^m e^{\eta h_i(t)} u_i}{\sum_{i=1}^m e^{\eta h_i(t)}}$$

We also have:

$$R_{cont}^*(h(t), 1, A, -A) = \int_0^1 \frac{\sum_{i=1}^m e^{\eta h_i(t) - \eta u_i t} \cdot u_i}{\sum_{i=1}^m e^{\eta h_i(t) - \eta u_i t}} dt \quad (19)$$

Denote by  $f(t) = \frac{\sum_{i=1}^m e^{\eta h_i(t) - \eta u_i t} \cdot u_i}{\sum_{i=1}^m e^{\eta h_i(t) - \eta u_i t}}$ . Then,

$$R_{disc}^*(h(t), 1, A, -A) = f(0),$$

whereas

$$R_{cont}^*(h(t), 1, A, -A) = \int_0^1 f(t) dt. \quad (20)$$

Note that:

$$f'(t) = \frac{-\eta \sum_{i=1}^m e^{\eta h_i(t) - \eta u_i t} \cdot u_i^2}{\sum_{i=1}^m e^{\eta h_i(t) - \eta u_i t}} + \frac{\eta \left( \sum_{i=1}^m e^{\eta h_i(t) - \eta u_i t} \cdot u_i \right)^2}{\left( \sum_{i=1}^m e^{\eta h_i(t) - \eta u_i t} \right)^2}.$$

Since  $u_i$ 's are bounded by in  $[-1, 1]$ , we get that  $f'(t) \geq -\eta$ . Hence  $f(t) \geq f(0) - \eta t$ , consequently,

$$R_{cont}^*(h(t), 1, A, -A) = \int_0^1 f(t) dt \geq \int_0^1 (f(0) - \eta t) dt = f(0) - \frac{\eta}{2}.$$

Therefore summing up for all  $t = 0, 1, \dots, T-1$ :

$$R_{disc}^*(\mathbf{0}, T, A, -A) - R_{cont}^*(\mathbf{0}, T, A, -A) \leq \eta T/2$$

$\square$

We conclude with the last proposition of this section which states that in games that satisfy the following Condition, we get that the optimizer can achieve reward at least  $T \text{Val}(A) + \Omega(\eta T)$  against a MWU learner.

**Condition 2.** There exists a minmax strategy  $x$  for the optimizer such that there exists two best responses for the learner,  $b_{i_1}, b_{i_2} \in \text{BR}(x)$ , which do not identify on  $\text{support}(x)$ . Namely, there exists an action  $a_k \in \text{support}(x)$  such that  $a_k^\top A b_{i_1} \neq a_k^\top A b_{i_2}$ .

**Proposition 11.** For any zero-sum game  $A \in \mathbb{R}^{n \times n}$  that satisfies Condition 1,  $\eta \in (0, 1)$  and  $T \in \mathbb{N}$ , there exists an optimizer, such that against a learner that uses MWU with step size  $\eta$ , achieves a utility of at least

$$T \text{Val}(A) + C_A \eta T,$$

where  $C_A$  is a constant that depends only on the game matrix  $A$ .

*Proof.* Let  $x, b_{i_1}, b_{i_2}, a_k$  be the strategies and actions guaranteed from Condition 1. Since  $b_{i_1}$  and  $b_{i_2}$  are both best responses for  $x$ ,  $x^\top A e_{i_1} = x^\top A e_{i_2}$ . By the Condition that there exists an action  $a_k \in \text{support}(x)$  such that  $a_k^\top A e_{i_1} \neq a_k^\top A e_{i_2}$ , it follows that there exists two strategies  $x', x'' \in \Delta(A)$  such that  $(x' + x'')/2 = x$ ,  $x'^\top A e_{i_1} > x''^\top A e_{i_1}$  and  $x''^\top A e_{i_1} < x'^\top A e_{i_2}$ . We propose the following strategy for the learner: at any odd time  $t$ , play  $x'$  and at any even time  $t$ , play  $x''$ . We will prove that in iterations  $t$  and  $t+1$  the reward of the optimizer is at least  $2 \text{Val}(A) + C_A \eta$  by playing as described. Therefore, if we just sum up over all  $t$ , we will get the desired result. Now, fix some odd  $t$ , and we will lower bound the sum of utilities of the optimizer in iterations  $t$  and  $t+1$ :

$$x(t)^\top A y(t) + x(t+1)^\top A y(t+1) = x'^\top A y(t) + x''^\top A y(t+1)$$

Denote  $u_i = x'^\top A e_i$  and  $v_i = x''^\top A e_i$ , then

$$x'^\top A y(t) + x''^\top A y(t+1) = \sum_{i=1}^n u_i y_i(t) + \sum_{i=1}^n v_i y_i(t+1).$$

Notice that

$$y_i(t+1) = \frac{y_i(t) e^{-\eta x'^\top A b_i}}{\sum_{j=1}^n y_j(t) e^{-\eta x'^\top A b_j}} = \frac{y_i(t) e^{-\eta u_i}}{\sum_{j=1}^n y_j(t) e^{-\eta u_j}}.$$

Therefore

$$\sum_{i=1}^n u_i y_i(t) + \sum_{i=1}^n v_i y_i(t+1) = \sum_{i=1}^n u_i y_i(t) + \sum_{i=1}^n \frac{v_i(t) y_i(t) e^{-\eta u_i}}{\sum_{j=1}^n y_j(t) e^{-\eta u_j}}. \quad (21)$$

Notice that

$$\frac{u_i + v_i}{2} = \frac{(x' + x'')^\top}{2} A e_i = x^\top A e_i \geq \text{Val}(A).$$

Hence,  $v_i(t) \geq 2 \text{Val}(A) - u_i(t)$ , hence the right hand side of Eq. (21) is at least

$$\sum_{i=1}^n y_i(t) u_i + 2 \text{Val}(A) - \sum_{i=1}^n \frac{u_i y_i(t) e^{-\eta u_i}}{\sum_{j=1}^n y_j(t) e^{-\eta u_j}} = 2 \text{Val}(A) + \sum_{i=1}^n y_i(t) u_i \left( 1 - \frac{e^{-\eta u_i}}{\sum_{j=1}^n y_j(t) e^{-\eta u_j}} \right)$$

The right hand side equals  $2 \text{Val}(A)$  if  $\eta = 0$ . We will differentiate it wrt  $\eta$  to get

$$\frac{d}{d\eta} \left( - \sum_{i=1}^n y_i(t) u_i e^{-\eta u_i} \right) = \frac{\sum_{i=1}^n y_i(t) u_i^2 e^{-\eta u_i}}{\sum_{j=1}^n y_j(t) e^{-\eta u_j}} - \left( \frac{\sum_{i=1}^n y_i(t) u_i e^{-\eta u_i}}{\sum_{j=1}^n y_j(t) e^{-\eta u_j}} \right)^2.$$

The right hand side equals the variance of a random variable, which we denote by  $Z_\eta$ , such that

$$\Pr[Z_\eta = u] = \sum_{i: u_i = u} \frac{y_i e^{-\eta y_i}}{\sum_{j=1}^n y_j e^{-\eta y_j}}.$$

We obtain that the sum of utilities for the optimizer in steps  $t$  and  $t+1$  is at least  $2 \text{Val}(A) + \int_0^\eta \text{Var}(Z_r) dr$ . Recall that we assumed that  $\eta \leq 1$ . Hence, if we prove that there exists some constant  $C_A$  such that  $\text{Var}(Z_r) \geq C_A$  for all  $r \in [0, 1]$ , we will obtain that the sum of utilities in iterations  $t$  and  $t+1$  is at least  $2 \text{Val}(A) + \eta C_A$ . This will imply that the sum of utilities in iterations 1 through  $T$  is at least  $T \text{Val}(A) + \eta T C_A / 2$ , assuming that  $T$  is even (if  $T$  is odd then we can play a minmax strategy in the last round and get a similar bound). It remains to bound  $\text{Var}(Z_r)$ . Recall that  $b_{i_1}$  and  $b_{i_2}$  are two best responses to  $x$ , and since  $\sum_{i=1}^{t-1} x(i) = \frac{t}{2} x$ , it holds from the definition of MWU that  $y_{i_1}(t)$  and  $y_{i_2}(t)$  are among the largest entries of  $y(t)$ . Consequently,  $y_{i_1}(t), y_{i_2}(t) \geq 1/n$ . This implies that there exists some  $C > 0$  (that possibly depends on  $A$  and  $n$ ), such that for all  $r \in (0, 1)$ ,  $\Pr[Z_r = u_{i_1}] = \Pr[Z_r = u_{i_2}] \geq C$ . Further, by Condition,  $u_{i_1} = x'^\top A b_{i_1} \neq x''^\top A b_{i_2} = u_{i_2}$ . Consequently, there exists some  $C_A > 0$  such that  $\text{Var}(Z_r) \geq C_A$  for all  $r \in [0, 1]$ . This is what we wanted to prove.  $\square$

## C Computational lower bound

In this section, we present the full omitted proof of Theorem 4.

*Proof of theorem 4.* We'll show a polynomial time reduction from OCDP to the hamiltonian cycle problem. Let  $n = |V|$  and  $m = |E|$ . Given an instance  $\pi_{Ham} = G(V, E)$ , where  $V = \{v_1, v_2, \dots, v_n\}$  and  $E = \{e_1, \dots, e_m\}$  of the Hamiltonian cycle problem, we create an instance of the OCDP problem as follows: the optimizer has  $m$  actions, denoted by  $\mathcal{A} = \{a_1, \dots, a_m\}$  and the learner has  $2n$  actions, denoted by  $\mathcal{B} = \{b_1, \dots, b_{|V|}, b_{in_1}, \dots, b_{in_{|V|}}\}$ . The matrices  $A$  and  $B$ , which correspond to the optimizer and learner's utility matrices respectively are defined as follows:

$$A[a_i, b_j] = \begin{cases} 1, & \text{if there exists } u \in V \text{ such that } e_i = (v_j, u), \text{ i.e. } e_i \text{ is outgoing edge of node } v_j \\ 0, & \text{otherwise} \end{cases}$$

$$A[a_i, b_{in_j}] = 0$$

$$B[a_i, b_j] = \begin{cases} -0.1 & \text{if } j = 1 \text{ and } \exists u \in V \text{ s.t. } e_i = (v_1, u), \text{ i.e. } e_i \text{ is an outgoing edge of node } v_1 \\ -4, & \text{if } j \neq 1 \text{ and } \exists u \in V \text{ s.t. } e_i = (v_j, u), \text{ i.e. } e_i \text{ is an outgoing edge of node } v_j \\ 1, & \text{if } \exists u \in V \text{ such that } e_i = (u, v_j), \text{ i.e. } e_i \text{ is an incoming edge of node } v_j \\ 0, & \text{o.w.} \end{cases}$$

$$B[a_i, b_{in_j}] = \begin{cases} 0.85, & \text{if } \exists u \in V \text{ such that } e_i = (v_j, u), \text{ i.e. } e_i \text{ is an outgoing edge of node } v_j \\ 0, & \text{o.w.} \end{cases}$$

Finally, we set  $k = T = n + 1$ , constructing the instance  $\pi_{OCDP} = (A, B, m, 2n, k, T)$ . We conclude the reduction by proving that  $\pi_{Ham}$  is a 'Yes' instance if and only if  $\pi_{OCDP}$  is a 'Yes' instance.

1.  $\pi_{Ham} \implies \pi_{OCDP}$  : Suppose we have a Hamiltonian cycle that visits vertices  $v_1 = v_{u_1} \rightarrow v_{u_2} \rightarrow \dots \rightarrow v_{u_n} \rightarrow v_{u_{n+1}} = v_{u_1} = v_1$ . Suppose that edge  $e_{p_i}$  connects  $v_{u_i}$  to  $v_{u_{i+1}}$ . We will prove that the sequence of actions  $a_{p_1}, a_{p_2}, \dots, a_{p_{n-1}}, a_{p_n}, a_{p_{n+1}} = a_{p_1}$  achieve reward exactly  $n+1$  for the optimizer. To prove that, it is enough to argue that if the optimizer plays as we described above, the learner will respond with  $b_{u_1}, b_{u_2}, \dots, b_{u_n}, b_{u_{n+1}} = b_{u_1}$ . Indeed, notice that  $A[a_{p_i}, b_{u_i}] = 1, \forall i \in [n+1]$ , since  $e_{p_i}$  is an outgoing edge of  $v_{u_i}$ , therefore the optimizer will be getting reward 1 every round. Now, to prove that the learner best responds as described, let us look at the historical rewards of the learner. Suppose the rewards for each of the  $m = 2 \cdot |V|$  actions of the learner for rounds  $1, 2, \dots, t-1$  is denoted by  $h(t) = (h_1(t), \dots, h_n(t), h_{in_1}(t), \dots, h_{in_n}(t)) \in \mathbb{R}^{2n}$ . We will prove inductively that after  $t$  rounds we will have  $h(t) = H(t)$ , where  $H(t)$  is defined as:

$$H_1(t) = \begin{cases} 0, & \text{if } t = 1 \\ -0.1, & \text{if } 2 \leq t \leq n \\ 0.9, & \text{if } t = n+1 \end{cases}$$

$$H_{u_i}(t) = \begin{cases} 0, & \text{if } t < i \\ 1, & \text{if } t = i, i > 1 \\ -3, & \text{if } t > i \end{cases}$$

$$H_{in_{u_i}}(t) = \begin{cases} 0, & \text{if } t \leq i \\ 0.85, & \text{if } t > i \end{cases}, i = 1, 2, \dots, n$$

Notice that the cases  $t = 1, 2$  are trivial; at  $t = 1$  everything is equal to 0 since the learner has accumulated no reward yet. At time  $t = 2$  we only update  $h_{in_1} \rightarrow 0.85, h_1 \rightarrow -0.1, h_{u_2} \rightarrow 1$ , thus giving us  $h(2) = H(2)$ . Suppose that after some  $t$  rounds, where  $2 \leq t \leq n$  rounds we have  $h(t) = H(t)$ . After  $a_{p_t}$  is played by the optimizer, the learner has to update  $h_{in_{u_t}} \rightarrow 0.85, h_{u_t} \rightarrow 1 - 4 = -3$  and if  $t < n$  we update  $h_{u_{t+1}} \rightarrow 1$  otherwise if  $t = n$  we update  $h_1 \rightarrow -0.1 + 1 = 0.9$ . Either way, we get  $h(t+1) = H(t+1)$ . To complete the proof, note that for each  $t$ , given that the of the learner history is  $H(t)$ , best response for the learner is  $b_{u_t}$ , thus the sequence of actions  $a_{p_1}, a_{p_2}, \dots, a_{p_n}, a_{p_{n+1}}$  achieves reward  $n+1$  for the optimizer.

2.  $\pi_{\text{OCDP}} \implies \pi_{\text{Ham}}$ : Suppose there is a sequence of actions played by the optimizer that receive reward  $n + 1$  in  $n + 1$  rounds, with actions  $a_{p_1}, a_{p_2}, \dots, a_{p_{n+1}}$ . We will prove that  $a_{p_1}, \dots, a_{p_n}$  correspond to the edges of a Hamiltonian Cycle starting from node  $v_1 = 1$ , i.e. edges  $e_{p_1}, \dots, e_{p_n}$  make a Hamiltonian Cycle, by connecting nodes  $v_1 = v_{u_1}, v_{u_2}, \dots, v_{u_n}$ . We again denote the rewards of the learner for rounds  $1, 2, \dots, t - 1$  with  $h(t) = (h_1(t), \dots, h_n(t), h_{\text{in}_1}(t), \dots, h_{\text{in}_n}(t)) \in \mathbb{R}^{2n}$ .

- **Action at  $t = 1$ :** Start by observing that the learner will play  $b_1$  at  $t = 1$  as it is the tie breaking rule, so the optimizer needs to play an action that achieves reward 1 against that. Note that the way we constructed the utility matrix of the optimizer, the only actions that achieve reward against  $v_i$  correspond to outgoing edges from vertex  $v_i$ . So at  $t = 1$ , the optimizer is forced to play an action that corresponds to an outgoing edge of  $v_1 = 1$ . Suppose the first action of the optimizer is  $a_{p_1}$ , corresponding to the edge  $e_{p_1} = (v_1, v_{u_2})$ .
- **Action at  $t = 2$ :** The rewards of the optimizer will be updated as follows, after the optimizer plays  $a_{p_1}$  and the learner plays  $b_1$ :

$$H_i(t) = \begin{cases} -0.1, & \text{if } i = 1 \\ 1, & \text{if } i = u_2 \\ 0, & \text{o.w.} \end{cases}$$

$$H_{\text{in}_i}(t) = \begin{cases} 0.85, & \text{if } i = 1 \\ 0, & \text{o.w.} \end{cases}$$

We note that now new best response of the learner will be  $b_{u_2}$ . Thus, the only way for the optimizer to obtain reward 1 for the second round is to play an action corresponding to an outgoing edge of node  $v_{u_2}$ . Suppose the action the optimizer plays is  $b_{p_2}$ , corresponding to an edge  $e_{p_2} = (v_{u_2}, v_{u_3})$ .

- **Actions at  $2 < t < n$ :** We will prove inductively that the first  $t \leq n - 1$  actions  $a_{p_1}, \dots, a_{p_t}$  correspond to edges of the form  $e_{p_1} = (v_{u_1}, v_{u_2}), e_{p_2} = (v_{u_2}, v_{u_3}), \dots, e_{p_t} = (v_{u_t}, v_{u_{t+1}})$  where  $u_i \neq u_j$  for  $1 \leq i < j \leq n$ , i.e. edges that define a simple path in the graph, starting from node  $v_1$ . By playing actions in such a manner, the historical rewards of the learner will be of the form:

$$H_1(t) = \begin{cases} 0, & \text{if } t = 1 \\ -0.1, & \text{if } 2 \leq t \leq n \end{cases}$$

$$H_{u_i}(t) = \begin{cases} 0, & \text{if } t < i \\ 1, & \text{if } t = i, i > 1 \\ -3, & \text{if } t > i \end{cases}$$

$$H_{\text{in}_{u_i}}(t) = \begin{cases} 0, & \text{if } t \leq i \\ 0.85, & \text{if } t > i \end{cases}, i = 1, 2, \dots, n$$

We proved that the first  $t$  actions correspond to edges of a simple path for  $t \leq 2$ . We assume for the inductive step that it is true for the first  $t < n - 1$  actions  $a_{p_1}, \dots, a_{p_t}$  of the optimizer. We will prove that the next action  $a_{p_{t+1}}$  by the optimizer corresponds to an edge  $e_{p_{t+1}}$  that extends this simple path. Since the first  $t$  actions are corresponding to a simple path, at round  $t + 1$  the learner's historical rewards are going to be equal to  $H(t + 1)$ . Note that the best response for the learner is going to be  $b_{u_{t+1}}$ . In order for the optimizer to gain reward 1 the  $t + 1$ 'th round, they have to be playing an action  $a_{p_{t+1}}$  for which  $e_{p_{t+1}}$  is an outgoing edge of the node  $v_{u_{t+1}}$ . Suppose the optimizer chooses to play  $a_{p_{t+1}}$ , where  $e_{p_{t+1}} = (v_{u_{t+1}}, v_{u_{t+2}})$ , and  $u_{t+2} = u_l$  for some  $1 \leq l \leq t + 1$ . At round  $t + 2$  the optimizer will still have to play an action that corresponds to an outgoing edge of  $v_{u_l}$ . However, after that, we will have  $h_{\text{in}_{u_l}}(t + 3) = 1.70$ , which will make  $b_{u_l}$  the best response. However, there is no action the optimizer can play against  $b_{u_l}$  in which they receive reward 1, making it impossible for the optimizer to obtain  $n + 1$  reward in  $n + 1$  rounds. Thus the optimizer needs to play  $e_{p_{t+1}} = (v_{u_{t+1}}, v_{u_{t+2}})$ , and  $u_{t+2} \neq u_l, l \in [t + 1]$ , completing the inductive step. Since we also proved that the first 2 actions are forming a simple path starting from  $v_1$ , using the inductive step, we conclude that the first  $n - 1$  actions have to correspond to a simple path in the graph starting from  $v_1$ .

- **Actions at  $t = n, n + 1$ :** Suppose the last two actions of the optimizer are  $a_{p_n}, a_{p_{n+1}}$ . We want to prove that  $e_{p_n} = (v_{u_n}, v_1), e_{p_{n+1}} = (v_1, u)$ , where  $u$  is some node connected to  $v_1$ . We know that the history of the learner at time  $t = n$ , as proven in the previous bullet point, is as follows:

$$h_{u_i}(n) = \begin{cases} -0.1, & \text{if } i = 1 \\ 1, & \text{if } i = n, i > 1 \\ -3, & \text{o.w.} \end{cases}$$

$$h_{\text{in}_{u_i}}(n) = \begin{cases} 0, & \text{if } i = n \\ 0.85, & \text{o.w.} \end{cases}, i = 1, 2, \dots, n$$

Note that the best response for the learner at time  $t = n$  will be  $b_{u_n}$ . Thus the optimizer will have to play an action  $a_{p_n}$  corresponding to an outgoing edge of  $v_{u_n}$ . Suppose  $e_{p_n} = (v_{u_n}, x)$ . Assume  $x \neq 1$ . Then the history is updated to  $h_{u_n} \rightarrow -3, h_x \rightarrow -2, h_{\text{in}_{u_n}} \rightarrow 0.85$ . Note that the best response is any action  $b_{u_{\text{in}_i}}$  for which the optimizer cannot get reward 1 from. Thus if  $x \neq 1$  it is impossible to get reward 1 every round. On the other hand, if  $x = 1$ , then the best response at time  $n + 1$  will be  $b_1$  since the  $h_1(n + 1) = -0.1 + 1 = 0.9 > 0.85$ , and thus playing any action corresponding to an outgoing edge of  $v_1$  will obtain reward 1 for the learner at that round.

Lastly, notice that the problem statement requests the entries of the weight matrix  $B$  to be in  $[0, 1]$ . While our construction have weights in  $[-4, 4]$ , scaling and shifting all the entries by the same amount does not affect the identity of the best response.  $\square$

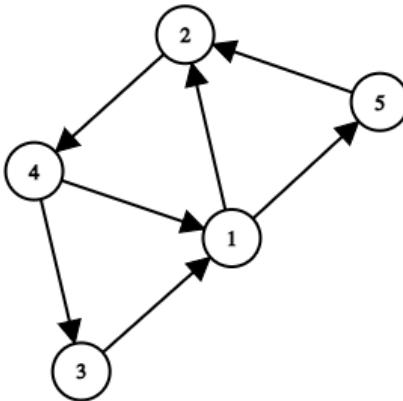


Figure 1: Graph  $G$

We will show an example of how we can make the reduction from Hamiltonian Cycle to OCDP. Imagine we have an instance of Hamiltonian Cycle that corresponds to the graph shown in 1, and we want to reduce to OCDP. Since we have  $|V| = 5$  vertices and  $|E| = 7$  edges, we instantiate the following instance of OCDP:  $(A, B, n = |E|, m = 2|V|, k = |V| + 1, T = |V| + 1)$ , where the actions sets are  $\mathcal{A} = \{e_1 = (1, 5), e_2 = (5, 2), e_3 = (1, 2), e_4 = (2, 4), e_5 = (4, 1), e_6 = (4, 3), e_7 = (3, 1)\}$   $\mathcal{B} = \{v_1, v_2, v_3, v_4, v_5, v_{in_1}, v_{in_2}, v_{in_3}, v_{in_4}, v_{in_5}\}$ , and the utility matrices  $A$  and  $B$  as shown in the tables below.

$$A =$$

	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_{in_1}$	$v_{in_2}$	$v_{in_3}$	$v_{in_4}$	$v_{in_5}$
$e_1$	1	0	0	0	0	0	0	0	0	0
$e_2$	0	0	0	0	1	0	0	0	0	0
$e_3$	1	0	0	0	0	0	0	0	0	0
$e_4$	0	1	0	0	0	0	0	0	0	0
$e_5$	0	0	0	1	0	0	0	0	0	0
$e_6$	0	0	0	1	0	0	0	0	0	0
$e_7$	0	0	1	0	0	0	0	0	0	0

$$B =$$

	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_{in_1}$	$v_{in_2}$	$v_{in_3}$	$v_{in_4}$	$v_{in_5}$
$e_1$	-1	0	0	0	1	0.85	0	0	0	0
$e_2$	0	1	0	0	-4	0	0	0	0	0.85
$e_3$	-1	1	0	0	0	0.85	0	0	0	0
$e_4$	0	-4	0	1	0	0	0.85	0	0	0
$e_5$	1	0	0	-4	0	0	0	0	0.85	0
$e_6$	0	0	1	-4	0	0	0	0	0.85	0
$e_7$	1	0	-4	0	0	0	0	0.85	0	0

Notice that the only way for the optimizer to achieve reward  $|V| + 6 = 5 + 1 = 6$  is if he plays the following actions:

$e_1, e_2, e_4, e_6, e_7, e_1$

and the learner's actions are the following:

$v_1, v_5, v_2, v_4, v_3, v_1$

The following table shows the rewards history of the learner during this game:

$$r(t) =$$

	$v_1$	$v_2$	$v_3$	$v_4$	$v_5$	$v_{in_1}$	$v_{in_2}$	$v_{in_3}$	$v_{in_4}$	$v_{in_5}$
$t = 0$	0	0	0	0	0	0	0	0	0	0
$t = 1$	-0.1	0	0	0	1	0.85	0	0	0	0
$t = 2$	-0.1	1	0	0	-3	0.85	0	0	0	0.85
$t = 3$	-0.1	-3	0	1	-3	0.85	0.85	0	0	0.85
$t = 4$	-0.1	-3	1	-3	-3	0.85	0.85	0	0.85	0.85
$t = 5$	-0.1	-3	1	-3	-3	0.85	0.85	0	0.85	0.85
$t = 5$	0.9	-3	-3	-3	-3	0.85	0.85	0.85	0.85	0.85
$t = 6$	0.8	-3	-3	-3	-3	1.70	0.85	0.85	0.85	0.85

## NeurIPS Paper Checklist

### 1. Claims

Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? NEUR

Answer: **[Yes]**

Justification: The introduction and abstract accurately reflect the main scope of the paper. We have proofs for the propositions and theorems of the paper in the appendix.

Guidelines:

- The answer NA means that the abstract and introduction do not include the claims made in the paper.
- The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A No or NA answer to this question will not be perceived well by the reviewers.
- The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
- It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

### 2. Limitations

Question: Does the paper discuss the limitations of the work performed by the authors?

Answer: **[Yes]**

Justification: The assumptions made in each claim are clearly described in the statement of the claim. We discuss possible future directions in section 4.

Guidelines:

- The answer NA means that the paper has no limitation while the answer No means that the paper has limitations, but those are not discussed in the paper.
- The authors are encouraged to create a separate "Limitations" section in their paper.
- The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
- The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
- The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
- The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
- If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
- While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

### 3. Theory Assumptions and Proofs

Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: **[Yes]**

Justification: The assumptions are stated clearly in each theorem, lemma or proposition. Full proofs of what was omitted in the main section, are included in the appendix.

Guidelines:

- The answer NA means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

#### 4. Experimental Result Reproducibility

Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

Answer: **[NA]**

Justification: This paper does not include any experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- If the paper includes experiments, a No answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
- If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
- Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general, releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
- While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
  - (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
  - (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
  - (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
  - (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

#### 5. Open access to data and code

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [NA]

Justification: This paper does not include any experiments requiring code.

Guidelines:

- The answer NA means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so “No” is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (<https://nips.cc/public/guides/CodeSubmissionPolicy>) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

## 6. Experimental Setting/Details

Question: Does the paper specify all the training and test details (e.g., data splits, hyper-parameters, how they were chosen, type of optimizer, etc.) necessary to understand the results?

Answer: [NA]

Justification: This paper does not include any experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

## 7. Experiment Statistical Significance

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [NA]

Justification: This paper does not include any experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The authors should answer “Yes” if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)

- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g. negative error rates).
- If error bars are reported in tables or plots, The authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

## 8. Experiments Compute Resources

Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

Answer: [NA]

Justification: This paper does not include any experiments.

Guidelines:

- The answer NA means that the paper does not include experiments.
- The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
- The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
- The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

## 9. Code Of Ethics

Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics <https://neurips.cc/public/EthicsGuidelines>?

Answer: [Yes]

Justification: The paper is compliant with the code of NeurIPS Code of Ethics.

Guidelines:

- The answer NA means that the authors have not reviewed the NeurIPS Code of Ethics.
- If the authors answer No, they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

## 10. Broader Impacts

Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

Answer: [Yes]

Justification: Discussed in the end of page 9.

Guidelines:

- The answer NA means that there is no societal impact of the work performed.
- If the authors answer NA or No, they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.

- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

## 11. Safeguards

Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pretrained language models, image generators, or scraped datasets)?

Answer: [NA]

Justification: This paper does not contain data models and therefore does not pose such risks.

Guidelines:

- The answer NA means that the paper poses no such risks.
- Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
- Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
- We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

## 12. Licenses for existing assets

Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

Answer: [NA]

Justification: This paper does not use any existing assets.

Guidelines:

- The answer NA means that the paper does not use existing assets.
- The authors should cite the original paper that produced the code package or dataset.
- The authors should state which version of the asset is used and, if possible, include a URL.
- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, [paperswithcode.com/datasets](https://paperswithcode.com/datasets) has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.

- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

### 13. New Assets

Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

Answer: [NA]

Justification: This paper does not use any existing assets.

Guidelines:

- The answer NA means that the paper does not release new assets.
- Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
- The paper should discuss whether and how consent was obtained from people whose asset is used.
- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

### 14. Crowdsourcing and Research with Human Subjects

Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

Answer: [NA]

Justification: This paper does not include crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

### 15. Institutional Review Board (IRB) Approvals or Equivalent for Research with Human Subjects

Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

Answer: [NA]

Justification: This paper does not include crowdsourcing nor research with human subjects.

Guidelines:

- The answer NA means that the paper does not involve crowdsourcing nor research with human subjects.
- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.