Computational Intractability of Strategizing against Online Learners

Angelos Assos ASSOS@MIT.EDU
MIT CSAIL

Yuval Dagan YDAGAN@TAUEX.TAU.AC.IL
Tel Aviv University, CS

Nived Rajaraman NIVED.RAJARAMAN@BERKELEY.EDU
Berkeley EECS

Editors: Nika Haghtalab and Ankur Moitra

Abstract

Online learning algorithms are widely used in strategic multi-agent settings, including repeated auctions, contract design, and pricing competitions, where agents adapt their strategies over time. A key question in such environments is how an optimizing agent can best respond to a learning agent to improve its own long-term outcomes. While prior work has developed efficient algorithms for the optimizer in special cases—such as structured auction settings or contract design—no general efficient algorithm is known.

In this paper, we establish a strong computational hardness result: unless $\mathsf{P} = \mathsf{NP}$, no polynomial-time optimizer can compute a near-optimal strategy against a learner using a standard no-regret algorithm, specifically Multiplicative Weights Update (MWU). Our result proves an $\Omega(T)$ hardness bound, significantly strengthening previous work that only showed an additive $\Theta(1)$ impossibility result. Furthermore, while the prior hardness result focused on learners using fictitious play—an algorithm that is not no-regret—we prove intractability for a widely used no-regret learning algorithm. This establishes a fundamental computational barrier to finding optimal strategies in general game-theoretic settings.

Keywords: repeated games, online learning, computational complexity

1. Introduction

Online learning algorithms are often used in scenarios containing other learning agents. This includes repeated auctions where a seller and buyers learn how to construct auctions and to bid, respectively (Nekipelov et al., 2015; Noti and Syrgkanis, 2021); Repeated contracts where a contractor learns how to set up contracts and agents learn how to take actions which maximize their revenue given these contracts (Guruganesh et al., 2024); competing retailers that learn how to set prices and to adapt to the demand (Du and Xiao, 2019; Calvano et al., 2020); Information design, where a principle wants to persuade an agent to take actions by deciding which information to share (Chen and Lin, 2023); and many more examples exist. In all these scenarios, the agents are operating in the same environment and influence each other's utilities.

In this paper we assume that there are two agents, a *learner* and an *optimizer*, and we ask:

Given that the learner is using a well-known learning algorithm to decide on his course of action, what is the best strategy for the optimizer to maximize her own reward (a.k.a. utility)?

We formalize this setting under the lens of *game theory*: we assume that there exists some *finite normal-form game* between a learner and an optimzer, as explained below. Concretely, this means that the optimizer can take actions from some set finite $\mathcal{A}$, the learner can take actions from some finite set $\mathcal{B}$, and there are functions A and B which determine the reward of the agents: if the optimizer plays action $\mathbf{x} \in \mathcal{A}$ and the learner plays action $\mathbf{y} \in \mathcal{B}$, the optimizer receives the reward $A(\mathbf{x}, \mathbf{y}) \in [-1, 1]$ and the learner receives the reward $B(\mathbf{x}, \mathbf{y}) \in [-1, 1]$. We assume *repeated game playing*: this means that in every iteration $t = 1, \ldots, T$, the optimizer chooses an action $\mathbf{x}(t)$, the learner chooses an action $\mathbf{y}(t)$, and the agents receives reward $A(\mathbf{x}(t), \mathbf{y}(t))$ and $B(\mathbf{x}(t), \mathbf{y}(t))$, respectively. The goal of each agent is to maximize their cumulative reward, summed over all the T iterations. For this purpose, the learner is using a standard online learning algorithm which determines which action to take in each iteration, whereas the optimizer's goal is to play according to a strategy that maximizes her cumulative reward given the learner's algorithm.

Perhaps the most fundamental notion by which online learning algorithms are measured is *regret*. That is, the difference between the cumulative reward of the learner and the reward that they would gain if they played a fixed action and if the optimizer played the same sequence of actions, which is defined as

$$\sum_{t=1}^{T} B(\mathbf{x}(t), \mathbf{y}(t)) - \max_{\mathbf{y} \in \mathcal{B}} \sum_{t=1}^{T} B(\mathbf{x}(t), \mathbf{y}) \ .$$

A learning algorithm is no-regret if its regret behaves as $o(T)$ and no-regret is a desirable property. A comprehensive theory has been developed around no-regret learning algorithms, and this includes the fundamental Hedge algorithm, also known as Multiplicative Weights Update (MWU), which is a simple algorithm that attains the asymptotically-optimal regret (Littlestone and Warmuth, 1994; Freund and Schapire, 1997; Arora et al., 2012).

A disadvantage of the notion of regret is that it does not take into account scenarios where the other agent (optimizer, in our case) utilizes an adaptive algorithm. In spite of that, standard no-regret learners are still commonly used. This raises the question of how an optimizing agent can behave in environments where other agents use no-regret learning algorithms to make decisions, to maximize its reward. Braverman et al. (2018) showed that in the context of repeated auctions between one seller and one buyer, if the buyer is using a mean-based no-regret learning algorithm (such as Multiplicative Weights Update) then the seller can extract full welfare, which is more than what they could get if the learner was strategic. This line of work has been generalized in scenarios with one (Deng et al., 2019a; Rubinstein and Zhao, 2024) or multiple no-regret buyers (Cai et al., 2023). Beyond mechanism design, similar interactions between a learner and an optimizer were studied in repeated contracts (Guruganesh et al., 2024) and in repeated information design (Bayesian Persuation) (Lin and Chen, 2024). In all these scenarios, an asymptotically-optimal strategy for the optimizer was developed. However, the solution in each instance was limited to the special game structure at hand. In general games, Deng et al. (2019b) has shown that the general optimization problem is essentially equivalent to solving a high-dimensional optimal control problem, however they do not provide an efficient algorithm. In this paper we show, in fact, that this is no coincidence: there is no efficient algorithm for the optimizer in general games, thereby resolving an open question of Guruganesh et al. (2024).

1.1. Our Contribution

Unlike prior work that we discussed above that has provided efficient algorithms only in structured settings or restricted cases, we establish a general computational hardness result. We prove that unless $\mathsf{P} = \mathsf{NP}$, no polynomial-time optimizer can compute a near-optimal strategy against a learner that uses a standard no-regret algorithm, specifically the Hedge/MWU. This formally establishes that no general efficient strategy exists for the optimizer in repeated game settings.

We formalize the optimization problem that we study. We assume that the learner's algorithm is Hedge (MWU), and it is parameterized by a step-size $\eta \geqslant 0$ (cf. Definition 2). Any sequence of optimizer's actions $\mathbf{x}(1), \ldots, \mathbf{x}(T)$ together with the learning rate η determine the learner's actions. This deremines the optimizer's reward: $R(\{\mathbf{x}(t) : t \geqslant 1\}) = \sum_{t=1}^{T} A(\mathbf{x}(t), \mathbf{y}(t))$. The optimizer's goal is:

Task 1 *Given $T \in \mathbb{N}$, given truth tables of the reward functions $A, B \colon \mathcal{A} \times \mathcal{B} \to [-1, 1]$ and given a step-size η for the learner, find a sequence of actions of the optimizer $x(1), \ldots, x(T) \in \mathcal{B}$ that approximately maximizes its reward $R\{\mathbf{x}(t) : t \geqslant 1\}$.*

Theorem 1 *Fix absolute constants $\alpha, \beta \in (0, 1]$. Let T denote the horizon, suppose the learner plays Hedge with learning rate parameterized as $\eta = 1/T^{1-\alpha}$ (cf. Definition 2) and suppose that the number of strategies per player is bounded by $|\mathcal{A}|, |\mathcal{B}| \leqslant T^{\beta}$. If $\mathsf{P} \neq \mathsf{NP}$, there exists no algorithm whose runtime is polynomial in T for Task 1 that outputs $\mathbf{x}(1), \ldots, \mathbf{x}(T)$ such that:*

$$R(\{x(t) : t \geqslant 1\}) \geqslant \max_{\{x^\star(t) : t \in [T]\}} R(\{x^\star(t) : t \in [T]\}) - cT, \tag{1}$$

for an absolute constant $c > 0$.

Our result significantly strengthens the hardness result of Assos et al. (2024), who show a computational hardness result for attaining the optimal reward in general two-player games. However their result had two significant limitations: (1) their impossibility result is based on approximating the reward of the optimizer up to an additive $\Theta(1)$-factor, whereas ours hold even for approximation up to $\Omega(T)$; (2) Their result assumed that the learner uses *follow-the-leader* (a.k.a. fictitious play), an algorithm that is not no-regret.

There are other differences which are not merely conceptual: the structure of the game considered in Assos et al. (2024) has learner action space of size T. Therefore establishing any (even $\Omega(T)$) hardness for such instances, does not preclude, for instance, the existence of an efficient algorithm which collects cumulative reward within $O(\sqrt{|\mathcal{B}|T})$ of that of the best optimizer. Further, it relied on the instability of the learner, which can change actions in each iteration, whereas we consider the MWU that is slow-changing, as is the case of most no-regret learners.

In contrast, our results show that even when the size of the game is an arbitrarily small (but constant) power of T, the regret scaling is $\Omega(T)$, thus ruling out the existence of such algorithms. In a sense, this is the best achievable: if the action space were to be any smaller, say polylogarithmic, the optimizer is now allowed to carry out superpolynomial compute in the size of the game. This essentially rules out approaches which prove hardness based on reducing to NP-complete problems of size poly($|\mathcal{B}|$), under $\mathsf{P} \neq \mathsf{NP}$. It is conceivable that under stronger assumptions, like ETH, using the same techniques, hardness results may be established even when the size of the game is yet smaller than even a polynomial in T.

From a technical point of view, our hardness reduction appeals to a special case of the traveling salesman problem with integral weights, $(1, 2)$-TSP, for which constant factor hardness of approximation is known Karpinski and Schmied (2012). Our result shows $\Omega(T)$ hardness of optimizing against MWU, even when the learning rate is as small as $1/T^{0.99}$, which is the regime where the learner is very slow changing[1]. When the learning rate is 0, the problem becomes easy, since the learner is no longer adaptive.

1.2. Related work

Multi-Agent Learning and Strategic Optimization The study of learning in multi-agent environments has a long history in game theory (Cesa-Bianchi and Lugosi, 2006; Youn, 2004; Freund and Gittins, 1998). A central question in this field is whether repeated interactions between learning agents lead to convergence to equilibrium (Robinson, 1951; Cesa-Bianchi and Lugosi, 2006). Classical results establish that when agents repeatedly play against each other using learning algorithms, their average play can converge to various equilibrium concepts. More recent work has shown that when all agents use similar learning algorithms, play can converge significantly faster to equilibria compared to classical results (Syrgkanis et al., 2015; Daskalakis et al., 2021; Anagnostides et al., 2022a,b; Farina et al., 2022; Piliouras et al., 2022; Zhang et al., 2023a). However, a different challenge arises when a strategic optimizer interacts with a learning agent: what is the best strategy for the optimizer to maximize long-term reward?

Optimizing Against Learning Agents in Structured Settings A series of works have studied how an optimizer should act when interacting with a learning agent in structured settings. In auction design, it has been shown that a seller interacting with a no-regret buyer—whether a single buyer or multiple buyers—can extract nearly the entire welfare, achieving an additional $\Omega(T)$ reward compared to what would be possible against a fully strategic buyer (Braverman et al., 2018; Deng et al., 2019a; Cai et al., 2023; Rubinstein and Zhao, 2024). A similar setting was studied in contract design, where a contractor optimally sets contracts for an agent using a learning algorithm (Guruganesh et al., 2024), and in Bayesian persuasion, where an information designer persuades a learning agent by strategically revealing information (Lin and Chen, 2024). In each of these cases, efficient algorithms were developed for the optimizer, leveraging the structure of the specific problem.

General Games: Lack of Efficient Optimizer Algorithms Beyond structured problems, several works have studied optimization against learning agents in general games, where the optimizer's goal is to maximize its reward in an arbitrary repeated game. Deng et al. (2019b) showed how to find an approximately optimal strategy for the optimizer. However, their work only provided a reduction to an optimal control problem and did not provide an efficient algorithm to solve it. Other works have explored similar settings in Bayesian games (Mansour et al., 2022), Pareto-optimal interactions (Arunachaleswaran et al., 2024), and optimizers facing arbitrary no-regret learners (Brown et al., 2024), but none of these provide a general efficient algorithm for the optimizer.

Computational Hardness in Special Cases While no efficient algorithm is known in general settings, some restricted cases have been solved efficiently. Guo and Mu (2023) developed a

1. 0.99 can be made any constant arbitrarily close to 1

polynomial-time algorithm for an optimizer interacting with an MWU learner in two-player games where each agent has only two actions. Masoumian and Wright (2024) provided an efficient algorithm for an optimizer interacting with a learner that has small bounded memory. Collina et al. (2023) showed that if a learner best-responds to a known optimizer policy, then the optimizer can achieve more than the per-round Stackelberg value, and they provided efficient algorithms to do so. In first-price auctions, Kumar et al. (2024) constructed a bidding algorithm that is both no-regret and robust against a strategic seller.

Learning the Stackelberg Equilibrium from Repeated Interactions The Stackelberg equilibrium represents the best possible reward an optimizer can obtain in a one-shot game, assuming the learner best-responds to the optimizer's actions. The problem of computing the Stackelberg equilibrium has been widely studied (Conitzer and Sandholm, 2006; Peng et al., 2019; Collina et al., 2023). In repeated interactions, prior work has studied whether an optimizer can learn the Stackelberg equilibrium when it does not know the learner's reward function in advance. Ananthakrishnan et al. (2024) showed an impossibility result when the learner is strategic and the optimizer has missing information. Other works have developed algorithms for learning a Stackelberg strategy in repeated games under similar uncertainty (Balcan et al., 2015; Marecki et al., 2012; Lauffer et al., 2022; Haghtalab et al., 2022; Brown et al., 2024). Additionally, Zhang et al. (2023b) explored how a principal can influence online learners to converge to specific equilibria.

2. Preliminaries

Repeated Games Notation. Consider a two-player game, where the optimizer's and learner's strategy sets are $\mathcal{A}$ and $\mathcal{B}$, respectively. The players are allowed to play distributions over their action sets (a.k.a. mixed strategies). At time t the optimizer and the learner play $\mathbf{x}(t) \in \Delta(\mathcal{A})$ and $\mathbf{y}(t) \in \Delta(\mathcal{B})$, respectively. The reward functions for the learner and the optimizer are parameterized by matrices $\boldsymbol{A}, \boldsymbol{B} \in [-1,1]^{|\mathcal{A}|\times|\mathcal{B}|}$. The reward collected by the optimizer and the learner at time t is $\mathbf{x}(t)^\top \boldsymbol{A}\mathbf{y}(t)$ and $\mathbf{x}(t)^\top \boldsymbol{B}\mathbf{y}(t)$, respectively. The notation $\mathbf{x}(a,t)$ denotes the probability of action $a \in \mathcal{A}$ at time t under $\mathbf{x}(t)$. Likewise, the notation $\mathbf{y}(b,t)$ is defined. The repeated game is assumed to be played over a horizon of length T. Furthermore, for $t \in [T]$, let,

$$\forall b \in \mathcal{B}, \quad r_{\text{learner}}(b,t) = \sum_{t'=1}^{t} \mathbf{x}(t')^\top \boldsymbol{B}\delta_b \tag{2}$$

denote the cumulative rewards of the actions of the learner at time t.

Definition 2 (Hedge/MWU learner) *Fix the step size $\eta \geqslant 0$ of the algorithm. In a repeated game with matrices $\boldsymbol{A}, \boldsymbol{B}$ for optimizer and learner, and an initial reward history $r_0 \in \mathbb{R}^m$, plays a mixed strategy, where at time t, for any $b \in \mathcal{B}$,*

$$\boldsymbol{y}(b,t) = \frac{\exp\big(\eta(r_{learner}(b,t) + r_0(b))\big)}{\sum_{b' \in \mathcal{B}} \exp\big(\eta(r_{learner}(b',t) + r_0(b'))\big)} \tag{3}$$

The parameter η, is referred to as the learning rate.

With notation in place, we define the main problem we study in this paper.

Definition 3 (Optimizing against MWU) *For a sequence of pure optimizer strategies $\{\boldsymbol{x}(t) : t \in [T]\}$, we will denote $R_{\eta,r_0}(\{\boldsymbol{x}(t) : t \geqslant 1\})$ as the cumulative reward collected by the optimizer, when playing against*

multiplicative weights with learning rate and initialization r_0. The problem of optimizing against MWU is defined as: find the sequence of pure strategies $\{\boldsymbol{x}(t) : t \in [T]\}$ which maximizes $R_{\eta,r_0}(\{\boldsymbol{x}(t) : t \geqslant 1\})$.

We notice that the standard definition of MWU is with $r_0 = 0$ and indeed we will prove hardness for this case. To show this, we will first prove a hardness when $r_0 > 0$ and provide a reduction between these two scenarios.

2.1. Reduction Workhorse: $(1,2)$-TSP

To prove the hardness result (Theorem 1) we will reduce the problem of optimizing against MWU, to an NP hard problem with an approximation hardness guarantee: we consider a variant of the traveling salesman problem, where the edge weights of the graph are restricted to be either 1 or 2.

Definition 4 ($(1,2)$-maxTSP) *An instance of $(1,2)$-TSP specified by $G = (V, W)$ is in one-one correspondence with an instance of $(1,2)$-maxTSP on $\widetilde{G} = (V, \widetilde{W})$ by the relation $\widetilde{W}(e) = 3 - W(e)$. This corresponds to replacing all the weight-2 edges by weight-1 edges, and vice versa. The objective is to find an Hamiltonian cycle in $\widetilde{G}$ with maximum weight.*

The following theorem shows constant approximation factor hardness for $(1,2)$-maxTSP and is proved in the Appendix (this follows trivially from Karpinski and Schmied (2012)).

Theorem 5 *Unless $\mathsf{P} = \mathsf{NP}$, there is no polynomial time approximation algorithm for $(1,2)$-maxTSP achieving a multiplicative approximation factor of $1067/1068 + \epsilon$ for any $\epsilon > 0$.*

3. Lower bound against MWU with non-zero reward history

The main result we show in this section is an $\Omega(T)$ hardness result for optimizing against MWU when initialized with an arbitrary reward history. In the subsequent section, we will extend this result to optimizing against MWU in the absence of any initialization.

3.1. Structure of the reduction

Define $E = (V \times V) \setminus \{(v, v) : v \in V\}$, the set of directed edges of the complete digraph on V. Given an instance of $(1,2)$-maxTSP on the vertex set V, specified by a set of edge weights $\{W_e : e \in E\}$, define the following game structure for both agents:

$$\textbf{Optimizer's action space } \mathcal{A}_{\text{init}} = E \textbf{ of size } m_{\text{init}} = |V|^2 - |V| \tag{4a}$$

$$\textbf{Learner's action space } \mathcal{B}_{\text{init}} = E \cup V_{\text{in}} \cup V_{\text{out}} \textbf{ of size } n_{\text{init}} = |V|^2 + |V| \tag{4b}$$

where V_{in} and V_{out} are auxiliary copies of V. For any vertex $v \in V$, we will use v_{in} and v_{out} to denote the corresponding vertex in V_{in} and V_{out}, and vice versa. Recall that we parameterize MWU's learning rate as $\eta = 1/T^{1-\alpha}$ for some $\alpha \in (0, 1]$. We will later choose V to be of size $n = T^{\min\{\alpha,\beta\}/3}$. Thus, the learner's and optimizer's action spaces are of size $O(T^{2\beta/3})$.

The reduction's idea is to create a game instance, where the optimal strategy for the optimizer is to play along some approximate solution of the maxTSP problem. Specifically, some edge e^+ is forced as the first edge to play, and the optimizer has to proceed with an approximate solution to the TSP problem, playing each edge along this path for k rounds. The reward matrix will be

constructed such that, when the optimizer plays along such strategy, it receives a utility that is proportional to its path weight, hence it would like to find a maximal-weight path. Notice that the optimizer does not have to play according to some path. To ensure that it cannot gain more utility deviating from playing each edge in a path for k rounds, the historical rewards are defined such that if the learner significantly diverges from such a play-structure, the learner will play actions in V_{in} and V_{out} from that point onwards, and these yield no utility for the optimizer.
The game matrices are defined as follows:

$$\text{For } (w,x) \in E, \quad \boldsymbol{A}(\underbrace{(u,v)}_{\text{opt.}}, \underbrace{(w,x)}_{\text{lr.}}) = \begin{cases} W_{(w,x)} & \text{if } (u,v) = (w,x) & \text{(5a)} \\ W_{(w,x)} & \text{if } u = x, & \text{(5b)} \\ 0 & \text{otherwise.} & \text{(5c)} \end{cases}$$

$$\text{For } a \in \mathcal{A}_{\text{init}},\ b \in \mathcal{B}_{\text{init}} \setminus E, \quad \boldsymbol{A}(a,b) = 0. \tag{6}$$

Thus, if the learner plays an edge e, the optimizer collects W_e units of reward for either responding with the same edge or responding with one which stems from the out-vertex of this edge. The optimizer surely collects no reward if the learner had chosen an action in V_{in} or V_{out}. On the other hand, for some $\varepsilon > 0$ and an arbitrary edge $e^\dagger = (u^\dagger, v^\dagger) \in E$, the game matrix for the learner is,

$$\text{For } (w,x) \in E, \quad \boldsymbol{B}(\underbrace{(u,v)}_{\text{opt.}}, \underbrace{(w,x)}_{\text{lr.}}) = \begin{cases} 0 & \text{if } (u,v) = e^\dagger & \text{(7a)} \\ 1 & \text{if } (u,v) \neq e^\dagger \text{ and } (w,x) = (u,v) & \text{(7b)} \\ -\varepsilon & \text{if } x = u, & \text{(7c)} \\ 0 & \text{otherwise.} & \text{(7d)} \end{cases}$$

$$\text{For } b \in V_{\text{in}}, \quad \boldsymbol{B}((u,v),b) = 2\mathbb{I}(b = v) \tag{8}$$

$$\text{For } b \in V_{\text{out}}, \quad \boldsymbol{B}((u,v),b) = 2\mathbb{I}(b = u) \tag{9}$$

Thus, the cumulative reward of an edge for the learner is the number of times the same edge was played by the optimizer minus the number of times the optimizer chooses an edge emanating from its out vertex scaled by ε.

In this section, we will prove the result assuming that the cumulative rewards of actions played by MWU are initially non-zero. This "initial reward" influences the probability of actions chosen by MWU. In particular, for $k > 0$ to be determined later, suppose the cumulative rewards are initialized as follows,

$$\text{For } v_{\text{in}} \in V_{\text{in}},\ r_0(v_{\text{in}}) = -k \tag{10}$$

$$\text{For } v_{\text{out}} \in V_{\text{out}},\ r_0(v_{\text{out}}) = -k \tag{11}$$

$$\text{For } (u,v) \in E \setminus \{e^\dagger\},\ r_0((u,v)) = 0 \tag{12}$$

$$r_0(e^\dagger) = k \tag{13}$$

In particular, when $k \gg \log(n)/\eta$, the initial distribution determined by MWU places most of its mass on the edge $e^\dagger$. As we will see later, the presence of the edge $e^\dagger$ is vital: it will ensure that there always exists an action with high cumulative reward ($\approx k$) for the learner. On the other

hand, the vertices $v_{\text{in}} \in V_{\text{in}}$ and $v_{\text{out}} \in V_{\text{out}}$ will serve to keep a handle on the number of edges the optimizer may play sinking / emanating out of a single vertex. These vertices begin with a high negative reward, $-k$, but gain reward at twice the rate of edges in $\mathcal{B}_{\text{init}}$. Once any of these vertices gain too much reward, MWU will displace most of its mass toward such actions, preventing the optimizer from gaining significant reward then onward.

3.2. Proof of Theorem 1 when learner has non-zero historical rewards

We consider some horizon length $T_{\text{init}} \leqslant T$. We will also implement the proof so that the only constraint on the value of k is that it falls into some range depending on other fixed problem parameters. When reducing the case without initialization to the case with initialization, the choice of T_{init} will be made by the optimizer, although it must essentially be $\approx \varepsilon T$ for it to be able to collect the most reward.

We start by arguing that the optimizer should not play an edge significantly more than k times, otherwise she will stop gaining reward. Define $V_{\text{excess}}(\Delta, t)$ as the set of vertices v for which there exists an incident edge that has been played more than $k + \Delta$ times by the optimizer by time t, where Δ is small compared to k.

Definition 6 *For $v \in V$ and $t = 1, \ldots, T$ denote by $d_{in}(v, t)$ (resp. $d_{out}(v, t)$) the number of times that the optimizer played an edge incoming (resp. outgoing) v, in iterations $1, \ldots, t$. We refer to $d_{in}(v, t)$ and $d_{out}(v, t)$ as in-degree and out-degree of v throughout the play. Let $V_{excess}(\Delta, t)$ denote the set of vertices which have in-degree or out-degree exceeding $k + \Delta$. Namely,*

$$V_{excess}(\Delta, t) = \{v \in V : \max\{d_{in}(v, t), d_{out}(v, t)\} \geqslant k + \Delta\} . \tag{14}$$

In the following lemma we prove that the optimizer will be punished for playing edges that emanate or sink in a vertex v too many times. Specifically, if at time $t_{\max}$, $V_{\text{excess}}(\Delta, t)$ is non empty, then from that point onwards the rewards the optimizer can obtain are limited. The idea is that if the optimizer exceeds the threshold of $k + \Delta$ for a vertex v, one of the special actions $v_{\text{in}}, v_{\text{out}}$ will receive much higher reward than the other actions of the optimizer. The MWU learner will then place almost all of its probability mass on this action, for which the optimizer earns no reward.

Corollary 7 *Suppose that the first time $V_{excess}(\Delta, t)$ becomes non-empty is at time $t_{\max}$, and so far the optimizer has collected reward $R(t_{\max})$. Then, by the end of the repeated interaction the optimizer can earn at most $R(t_{\max}) + 1$ rewards (for appropriate value of Δ).*

Next, we argue that the optimizer will not gain much reward playing edges emnating non-heavy vertices — where a non-heavy vertex is one that the optimizer played at least $k - \Delta$ times some edge incoming to it. Define $S_{\text{heavy}}(\Delta, t)$, the set of *heavy vertices* at time t. This is the union of $\{v^\dagger\}$ with the vertices $v \in V$ which satisfy the condition that there exists an edge of the form $e = (\cdot, v)$ such that $E(e, t) \geqslant k(1 - \varepsilon) - \Delta$. Since we restrict ourselves to time $t_{\max}$, vertices have in-degree at most $k + \Delta$ (as induced by the edges played by the optimizer). Thus, the heavy vertices are those with high in-degree, induced almost entirely by $\approx k$ copies of a single incoming edge.

Definition 8 *Define $S_{\text{heavy}}(\Delta, t)$ to be the set of vertices $v \in V$ for which there exists an edge $(\cdot, v)$ for which this edge has been used by the optimizer at least $k(1 - \varepsilon) - \Delta$ times plus $v^\dagger$. Precisely:*

$$S_{\text{heavy}}(\Delta, t) = \{v^\dagger\} \cup \{v \in V | \exists u \in V, Q((u, v), t) \geqslant k(1 - \varepsilon) - \Delta\} \tag{15}$$

where $Q((u,v),t)$ is the number of time that (u,v) was played by the optimizer in iterations $1,\dots,t$.

In Lemma 9, we argue that if we consider the set of heavy vertices, $S_{\text{heavy}}(\Delta, t_{\max})$, the optimizer essentially only collects reward for the edges played which emanate from a vertex in this set. We will denote this cumulative reward by $r_{\text{heavy}}(\Delta, t_{\max})$. The idea is that any vertex $v \notin S_{\text{heavy}}(\Delta, t_{\max})$ will never have a single edge played more than $k(1-\varepsilon)-\Delta$ times incoming into it. The optimizer may only collect reward for an edge emanating from this vertex if:

1. MWU places high mass on some edge incoming into this vertex. This is not possible because every edge incoming into v has low frequency (as induced by the optimizer).

2. MWU places high mass on the same edge. This is not possible until the same edge has been pulled many (at least approximately $k(1-\varepsilon)$) times by the optimizer. After this point, the same edge can be played at most until its frequency hits $k+\Delta$. In the small interim interval is when any reward can be collected for picking this edge.

Lemma 9 *Let $r_{\text{heavy}}(\Delta, t)$ count the contribution of the total reward collected by the optimizer at time t arising only from edges which emanate from a vertex in $S_{\text{heavy}}(t_{\max})$. Then,*

$$r_{optimizer}(t_{\max}) \leqslant r_{\text{heavy}}(\Delta, t_{\max}) + 2n(k\varepsilon + 2\Delta) + 3 \tag{16}$$

The above assertions have a few consequences. First, we have that the optimizer should be very careful not to exceed playing one action more than $k+\Delta$ times, and risks receiving almost no reward for the rest of the game (Corollary 7). We then define the 'heavy' vertices to be the vertices for which an edge incident to them have been played a significant amount of times. Then, with Lemma 9, we argued that it is enough to look at the reward of the edges emanating from 'heavy' vertices of the optimizer, $r_{\text{heavy}}(\Delta, t_{\max})$, as they make up most of the reward of the optimizer.

We will now try to unpack and upper bound $r_{\text{heavy}}(\Delta, t_{\max})$. Notice that for every $v \in S_{\text{heavy}}(\Delta, t_{\max})$, there is a unique vertex u for which the action (u,v) is played more than $k(1-\varepsilon)-\Delta$ times. There cannot be more than one, otherwise we would have $d_{\text{in}}(v,t) = 2(k(1-\varepsilon)-\Delta) > k+\Delta$, which is a contradiction to the fact that V_{excess} is empty prior to reaching time $t_{\max}$. Similarly, one can prove that for every vertex in v there can be at most one action (v,w) having reward at least $k(1-\varepsilon)-\Delta$. This motivates the following graph construction of G'.

Definition 10 *Construct a graph G' using the vertices of V as follows; add an edge from $u \longrightarrow v$ if $v \in S_{\text{heavy}}(\Delta, t_{\max})$ and $E((u,v), t_{\max}) \geqslant k(1-\varepsilon)-\Delta$.*

The first observation, as already hinted, is that G' is a graph in which all vertices have in-degree and out-degree at most one. That means that we may split G' into disjoint paths and cycles, which we denote as $C_1, C_2, \cdots C_s$. Our goal will now be to bound the rewards obtained by actions emanating from the paths (Lemma 11) and cycles (Lemma 12) of the Graph G' and combining the two to get a global upper bound of the reward of the optimizer (Lemma 13), emanating from vertices in $S_{\text{heavy}}(\Delta, t)$. We begin by bounding the rewards of actions/edges that emanate from vertices of a specific path of the graph G'; We argue that the total reward collected by the optimizer for edges emanating from these vertices is approximately upper bounded by the sum of the weights (in the maxTSP instance) of the edges along this path along with a free edge pointing from the last vertex to an arbitrary vertex, multiplied by k. Without loss of generality, we assume that the component C_1 contains the edge $e^{\dagger}$.

Lemma 11 *Consider any path C_i in G' composed of vertices $z_1, z_2, \cdots, z_m \in V$. The total reward collected by the optimizer for edges which stem from vertices on C_i is upper bounded by,*

$$\sum_{i=1}^{m}(k(1+\varepsilon)+3\Delta)W(z_i, z_{i+1}) + 2(k+\Delta)\mathbb{I}(i=1). \tag{17}$$

where z_{i+1} is an arbitrary (free) vertex.

We continue by bounding the rewards of actions/edges that emanate from vertices of a specific cycle of the graph G'; we show that the total reward collected by the optimizer for edges emerging from vertices in C_i is approximately upper bounded by the weight of edges in some $|C_i|-1$ length path within C_i up to a factor of k.

Lemma 12 *Consider any cycle C_i in G' composed of vertices $z_1, z_2, \cdots, z_m$. The total reward collected by the optimizer for edges which stem from vertices on C_i is upper bounded by,*

$$(k(1+3\varepsilon)+7\Delta)\sum_{i\in[m]\setminus\{i^\star\}} W(z_i, z_{i+1}) + 2(k+\Delta)\mathbb{I}(i=1) + \frac{1}{n} \tag{18}$$

for some $i^\star \in [m]$, where $z_{m+1} \triangleq z_1$.

We now combine the above to get an upper bound on the total reward heavy edges can contribute. We argue that we can combine the edges of graph G' to form a Hamiltonian Cycle without losing any reward. Using this fact, we can then upper bound the contribution of edges emanating from the heavy vertices with the reward of the Hamiltonian Cycle we obtain when combining all the heavy vertices. This, together with Corollary 7 and Lemma 9 gives the following result.

Lemma 13 *Assume $\Delta \geqslant \frac{1}{\eta}\log\left(2n^2T_{init}\right)$. Then, the reward collected by the optimizer at the end of the repeated interaction is at most,*

$$(k(1+5\varepsilon)+9\Delta)\sum_{i=1}^{n} W(z_i, z_{i+1}) + (2k+2\Delta+4) \tag{19}$$

where $z_1 \to z_2 \to \cdots \to z_n \to z_1$ is some Hamiltonian cycle on the vertex set V, and $z_{n+1} = z_1$. This upper bound applies for any value of horizon T_{init}, as long as Δ is suitably large.

The last step is to show that there exists an optimizer strategy which collects reward approximately lower bounded by the weight of the maximum weight Hamiltonian cycle (i.e., the solution to the maxTSP instance) multiplied by k. This is established in Lemma 14.

Lemma 14 (Bounding the reward of the best optimizer) *Consider any maximum weight Hamiltonian cycle in G, defined by the sequence of vertices $z_1^\star \to z_2^\star \to \cdots \to z_n^\star \to z_1^\star$. As long as $T_{init} \geqslant nk$, there exists an optimizer strategy such that the total reward collected is at least,*

$$(1-\varepsilon)^2 k \times \left(1-\frac{2}{n}\right)\sum_{i=1}^{n} W(z_i^\star, z_{i+1}^\star), \tag{20}$$

where $z_{n+1}^\star = z_1^\star$, and assuming that $\eta\varepsilon k \geqslant \frac{1}{\varepsilon-\varepsilon^2}\log\left(n^3\right)$.

We now have all we need to complete the proof.

Proof sketch of the theorem. Suppose we could approximate the optimal rewards of the optimizer; by Lemma 13 we know that the optimizer can achieve rewards up to $k(1+\epsilon_1)$ times the length of a Hamiltonian cycle in the maxTSP instance, where ϵ_1 can get arbitrarily close to 0. On the other hand, we know by Lemma 14 that the best the optimizer can do is at least $k(1-\epsilon_2)$ times the length of the heaviest Hamiltonian cycle in the maxTSP instance, where ϵ_2 can also get arbitrarily close to 0. By Theorem 5, we know that unless $\mathsf{P} = \mathsf{NP}$, it is impossible to approximate the weight of the optimal Hamiltonian cycle to within a $(1+c)$-factor for some $c > 0$. This implies that the rewards of the optimizer cannot be to close to optimal, and thereby leads to the $\Omega(T_{\text{init}})$ hardness guarantee.

4. Lower bound against MWU: removing the initialization

In this section, we will show how to reduce the case of MWU without initialization to the case with initialization. To do this, we will modify the structure of the game discussed in Section 3.1 for optimizing against MWU with the initialization of the form eqs. (10) to (13) and add a few additional actions. For $p \in \mathbb{N}$ to be determined later,

$$\textbf{Optimizer's action space } \mathcal{A} = \mathcal{A}_{\text{init}} \cup \{\blacklozenge\} \textbf{ of size } m = |V|^2 - |V| + 1 \tag{21a}$$

$$\textbf{Learner's action space } \mathcal{B} = \mathcal{B}_{\text{init}} \cup \widetilde{\mathcal{B}}_{\text{init}} \textbf{ of size } n = (p+1)(|V|^2 + |V|) \tag{21b}$$

The optimizer's action space is augmented by a single special action $\blacklozenge$. On the other hand, the learner's action space is augmented by $\widetilde{\mathcal{B}}_{\text{init}}$, which contains p copies of each action in $\mathcal{B}_{\text{init}}$. Each of these p copies will behave symmetrically with respect to the game matrices $\boldsymbol{A}$ and $\boldsymbol{B}$, and so we will simply refer to the "counterpart" of any $b \in \mathcal{B}_{\text{init}}$ as any one of its copies in $\widetilde{\mathcal{B}}_{\text{init}}$. We will choose $p = T^{\beta/3}$, which will bring the size of the learner's action space to be $O(T^\beta)$.

The game matrices are amended as follows. For any learner edge $\widetilde{b} \in \widetilde{\mathcal{B}}_{\text{init}}$ and any action $a \in \mathcal{A}_{\text{init}}$,

$$\boldsymbol{A}(a, \widetilde{b}) = 0. \tag{22}$$

Furthermore, the optimizer collects no reward for playing $\blacklozenge$. Namely,

$$\text{For all } b \in \mathcal{B},\ \boldsymbol{A}(\blacklozenge, b) = 0. \tag{23}$$

From the point of view of the learner, the actions in $\mathcal{B}_{\text{init}}$ and $\widetilde{\mathcal{B}}_{\text{init}}$ are not symmetric. For a small constant $\beta > 0$ to be determined later, and any $b \in \mathcal{B}_{\text{init}}$,

$$\boldsymbol{B}(\blacklozenge, b) = \begin{cases} k^\star/T & \text{if } b = e^\dagger \\ -k^\star/T & \text{if } b \in V_{\text{in}} \cup V_{\text{out}} \\ 0 & \text{otherwise.} \end{cases}, \tag{24}$$

Where $k^\star = \varepsilon(1-\varepsilon)^{-1}T^{1-\min\{\alpha,\beta\}/3}$. For any $b \in \mathcal{B}_{\text{init}}$ and its counterpart $\widetilde{b} \in \widetilde{\mathcal{B}}_{\text{init}}$,

$$\boldsymbol{B}(\blacklozenge, \widetilde{b}) = \boldsymbol{B}(\blacklozenge, b) - \gamma, \tag{25}$$

where $\gamma = \beta/3(1-\varepsilon)^{-1}T^{-\alpha}\log(T)$. On the other hand, for any learner action $b \in \mathcal{B}_{\text{init}}$ and its set of counterparts $\widetilde{b} \in \widetilde{\mathcal{B}}_{\text{init}}$, and any optimizer edge $a \in \mathcal{A}_{\text{init}}$,

$$\boldsymbol{B}(a, \widetilde{b}) = \boldsymbol{B}(a, b) \tag{26}$$

4.1. Proof sketch of Theorem 1

The intuition behind the amended game structure is as follows. In the initial iteration, observe that the presence of $p \gg 1$ copies of each action in $\mathcal{B}_{\text{init}}$ ensures MWU does not place a significant amount of probability mass on actions in $\mathcal{B}_{\text{init}}$. This in turn ensures that the optimizer cannot collect much reward in the first step. In order to mitigate this issue in future steps, notice that whenever the optimizer plays $\blacklozenge$, the reward on all actions in $\widetilde{\mathcal{B}}_{\text{init}}$ decreases a little bit. Once $\blacklozenge$ is played sufficiently many times, the actions in $\widetilde{\mathcal{B}}_{\text{init}}$ are downweighted enough to the point where MWU no longer places a significant amount of mass on them; the optimizer can begin to collect reward after this point. How many times should the optimizer play $\blacklozenge$ before this starts to happen?

It will turn out that the answer to this question will be $\approx (\eta\beta)^{-1}\log(p)$ times. Furthermore, the choice of γ is reverse engineered from the equation $(\eta\beta)^{-1}\log(p) = (1-\varepsilon)^2 T$. Thus, a good optimizer would need to play the action $\blacklozenge \approx (1-\varepsilon)^2 T$ times before the effect of the actions in $\widetilde{\mathcal{B}}_{\text{init}}$ is nullified. This leaves out a small $\approx 2\varepsilon T$ portion of the horizon where the optimizer may collect rewards. By virtue of the structure of the game, however, the action $e^{\dagger}$ has collected $\approx k^{\star}$ cumulative reward by this point, and likewise, actions in $V_{\text{in}} \cup V_{\text{out}}$ have cumulative reward $\approx -k^{\star}$. Observe the resemblance to the initialization in eqs. (10) to (13): we may now use techniques for bounding the reward of the optimizer for the case of MWU with initialization. In particular, the rewards collected by the optimizer are essentially bounded by Lemma 13.

Lemma 15 *Assume that $\Delta \geqslant \frac{1}{\eta}\log(2n^2T)$. The cumulative reward collected by the optimizer, when playing against MWU is upper bounded by,*

$$2T^{1-\beta\varepsilon/3} + (k^{\star}(1+5\varepsilon)+9\Delta)\sum_{i=1}^{n} W(z_i, z_{i+1}) + (2k^{\star}+2\Delta+4) \tag{27}$$

for some Hamiltonian cycle $z_1 \to z_2 \to \cdots \to z_n \to z_1$ on G.

The same argument will apply for lower bounding the total reward collected by the best optimizer strategy. We will consider the optimizer which plays $\blacklozenge$, $(1-\varepsilon)T$ times first, prior to following the optimizer strategy outlined in Lemma 14. Note that the remaining duration in the epoch is εT which is at least $nk^{\star}$.

Lemma 16 *Consider any maximum weight Hamiltonian cycle in G, defined by the sequence of vertices $z_1^{\star} \to z_2^{\star} \to \cdots \to z_n^{\star} \to z_1^{\star}$. For any absolute constant $\gamma \in (0,1)$, there exists an optimizer strategy such that the total reward collected is at least,*

$$(1-\varepsilon)^3 k^{\star} \times \frac{1}{1+p^{-\varepsilon}}\left(1-\frac{2}{n}\right)\sum_{i=1}^{n} W(z_i^{\star}, z_{i+1}^{\star}). \tag{28}$$

We may now combine Lemmas 15 and 16 to prove Theorem 1, using the fact that both equations, up to an $\approx k^{\star}$ factor, capture the weight of a Hamiltonian cycle on G.

5. Conclusion and future questions

We show that it is hard in general to optimize against a no-regret learner: specifically, against Hedge/MWU. This implies that positive results can only be obtained for games with specific

structure, as was extensively studied in prior literature. It remains open what is the best runtime for the optimizer. While it is not polynomial in the game size unless $\mathsf{P} = \mathsf{NP}$, can it be polynomial in T? Can the runtime depend exponentially only on $\min(|\mathcal{A}|, |\mathcal{B}|)$? Concretely, is there an algorithm that runs in time $\text{poly}(|\mathcal{A}|, |\mathcal{B}|, T)e^{O(\min(|\mathcal{A}|,|\mathcal{B}|))}$? Further, beyond the special cases studied, is there a broader class of games in which it is possible to efficiently find optimize the optimizer's reward? Beyond MWU, there is an efficient optimization algorithm against no-swap regret learners (Deng et al., 2019b), but what about other learners? Taking the learner's perspective, can one analyze how well it performs against optimizers and which learning algorithms are desired in such a scenario? See related work for some studies in this direction.

References

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*, 2022a.

Ioannis Anagnostides, Ioannis Panageas, Gabriele Farina, and Tuomas Sandholm. On last-iterate convergence beyond zero-sum games. In *International Conference on Machine Learning*, 2022b.

Nivasini Ananthakrishnan, Nika Haghtalab, Chara Podimata, and Kunhe Yang. Is knowledge power? on the (im) possibility of learning from strategic interaction. *arXiv preprint arXiv:2408.08272*, 2024.

Sanjeev Arora, Elad Hazan, and Satyen Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory of computing*, 8(1):121–164, 2012.

Eshwar Ram Arunachaleswaran, Natalie Collina, and Jon Schneider. Pareto-optimal algorithms for learning in games. *arXiv preprint arXiv:2402.09549*, 2024.

Angelos Assos, Yuval Dagan, and Constantinos Costis Daskalakis. Maximizing utility in multi-agent environments by anticipating the behavior of other learners. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=0uGlKYS7a2.

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.

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.

William Brown, Jon Schneider, and Kiran Vodrahalli. Is learning in games good for the learners? *Advances in Neural Information Processing Systems*, 36, 2024.

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.

Emilio Calvano, Giacomo Calzolari, Vincenzo Denicolo, and Sergio Pastorello. Artificial intelligence, algorithmic pricing, and collusion. *American Economic Review*, 110(10):3267–3297, 2020.

Nicolo Cesa-Bianchi and Gábor Lugosi. *Prediction, learning, and games*. Cambridge university press, 2006.

Yiling Chen and Tao Lin. Persuading a behavioral agent: Approximately best responding and learning. *arXiv preprint arXiv:2302.03719*, 2023.

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.

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.

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.

Yuan Deng, Jon Schneider, and Balasubramanian Sivan. Prior-free dynamic auctions with low regret buyers. *Advances in Neural Information Processing Systems*, 32, 2019a.

Yuan Deng, Jon Schneider, and Balasubramanian Sivan. Strategizing against no-regret learners. *Advances in neural information processing systems*, 32, 2019b.

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.

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.

S. Freund and L. Gittins. Strategic learning and teaching in games. *Econometrica*, 66(3):597–625, 1998.

Yoav Freund and Robert E Schapire. A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of computer and system sciences*, 55(1):119–139, 1997.

Xinxiang Guo and Yifen Mu. The optimal strategy against hedge algorithm in repeated games. *arXiv preprint arXiv:2312.09472*, 2023.

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.

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.

Marek Karpinski and Richard Schmied. On approximation lower bounds for tsp with bounded metrics. *arXiv preprint arXiv:1201.5821*, 2012.

Rachitesh Kumar, Jon Schneider, and Balasubramanian Sivan. Strategically-robust learning algorithms for bidding in first-price auctions. *arXiv preprint arXiv:2402.07363*, 2024.

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.

Tao Lin and Yiling Chen. Persuading a learning agent. *arXiv preprint arXiv:2402.09721*, 2024.

Nick Littlestone and Manfred K Warmuth. The weighted majority algorithm. *Information and computation*, 108(2):212–261, 1994.

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.

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.

Alireza Masoumian and James R Wright. Model selection for average reward rl with application to utility maximization in repeated games. *arXiv preprint arXiv:2411.06069*, 2024.

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.

Gali Noti and Vasilis Syrgkanis. Bid prediction in repeated auctions with learning. In *Proceedings of the Web Conference 2021*, pages 3953–3964, 2021.

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.

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.

Julia Robinson. An iterative method of solving a game. *Annals of mathematics*, pages 296–301, 1951.

Aviad Rubinstein and Junyao Zhao. Strategizing against no-regret learners in first-price auctions. In *Proceedings of the 25th ACM Conference on Economics and Computation*, pages 894–921, 2024.

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.

J. Youn. Learning algorithms in games. *Journal of Game Theory*, 56(2):123–134, 2004.

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*, 2023a.

Brian Hu Zhang, Gabriele Farina, Ioannis Anagnostides, Federico Cacciamani, Stephen Marcus McAleer, Andreas Alexander Haupt, Andrea Celli, Nicola Gatti, Vincent Conitzer, and Tuomas Sandholm. Steering no-regret learners to optimal equilibria. 2023b.

Appendix A. Supplementary results

A.1. Hardness for maxTSP: Proof Theorem 5

First, we define the minimization version of $(1,2)$-TSP that was defined in literature:

Definition 17 ($(1,2)$-TSP) *The $(1,2)$-TSP problem is specified by a weighted complete graph $G = (V, W)$ on n vertices where $W \in \{1,2\}^{\binom{V}{2}}$. The weight of a Hamiltonian cycle (i.e., each vertex is visited exactly once) $v_1 \to v_2 \to \cdots \to v_n \to v_1$ is $\sum_{i=1}^{n} W(v_i, v_{i+1})$. The objective is to find a Hamiltonian cycle in V having minimum weight.*

The following is known about the hardness of $(1,2)$-TSP:

Theorem 18 (Hardness of $(1,2)$-TSP (Karpinski and Schmied, 2012)) *Unless $\mathsf{P} = \mathsf{NP}$, there is no polynomial time approximation algorithm for $(1,2)$-TSP achieving a multiplicative approximation factor of $535/534 - \epsilon$ for any $\epsilon > 0$.*

As a consequence of Theorem 18, hardness of approximation for $(1,2)$-maxTSP is established below.

Consider any instance $G = (V, W)$ of $(1,2)$-TSP and the associated instance $\widetilde{G} = (V, \widetilde{W})$ of $(1,2)$-maxTSP. Consider any Hamiltonian cycle C on V. By the relation between W and $\widetilde{W}$, we have that $W(C) = 3n - \widetilde{W}(C)$. Taking the minimum over Hamiltonian cycles C, for any cycle C,

$$\frac{3n - W(C)}{3n - \min_{C'} W(C')} = \frac{\widetilde{W}(C)}{\max_{C'} \widetilde{W}(C)}$$

Unless $\mathsf{P} = \mathsf{NP}$, it is not possible to find in polynomial time a Hamiltonian cycle C such that $W(C)/\min_{C'} W(C) \leqslant 535/534 - \epsilon$. By contradiction this implies that we cannot find a Hamiltonian cycle C such that $\widetilde{W}(C)/\max_{C'} \widetilde{W}(C) \geqslant \frac{1067}{1068} + \varepsilon$ in polynomial time, unless $\mathsf{P} = \mathsf{NP}$.

Appendix B. Optimizing against MWU with initialization

In this section, we prove a lower bound on optimizing when the learner has non-zero reward history, formulated as follows:

Theorem 19 *Fix absolute constants $\alpha, \beta \in (0,1]$. Suppose the learner plays MWU (Definition 2) with a specific non-zero initialization of reward history, and learning rate parameterized as $\eta = 1/T^{1-\alpha}$. There exist game matrices $\boldsymbol{A}, \boldsymbol{B} \in [-1,1]^{|\mathcal{A}| \times |\mathcal{B}|}$, where $|\mathcal{A}|, |\mathcal{B}| \leqslant T^{\beta}$, such that unless $\mathsf{P} = \mathsf{NP}$, there exists no polynomial time optimizer which finds a sequence of pure strategies for the optimizer $\{\boldsymbol{x}(t) : t \in [T]\}$ satisfying,*

$$R(\{\boldsymbol{x}(t) : t \in [T]\}) \geqslant \max_{\{\boldsymbol{x}^\star(t) : t \in [T]\}} R(\{\boldsymbol{x}^\star(t) : t \in [T]\}) - cT \tag{29}$$

For a sufficiently small absolute constant $c > 0$.

Furthermore, note that for any directed edge $(w, x) \in E$, the cumulative reward of the learner at time t is,

$$r_{\text{learner}}((w,x), t) = r_0((w,x)) + \sum_{t'=1}^{t} \boldsymbol{x}(t')^\top \boldsymbol{B} \delta_{(w,x)} \tag{30}$$

$$= \begin{cases} E((w,x),t) - \varepsilon d_{\text{out}}(x,t) & \text{if } (w,x) \neq e^{\dagger}, \\ k - \varepsilon d_{\text{out}}(x,t) & \text{otherwise.} \end{cases} \tag{31}$$

where $d_{\text{out}}(x,t)$ is the out-degree of vertex x at time t (as induced by the edges played by the optimizer) and $E(e,t)$ counts the number of times the edge e was played by the optimizer up to an including time t. eq. (30) results from the fact that each time the edge $(w,x) \neq e^{\dagger}$ is played by the optimizer the cumulative reward increases by 1, while each edge leaving x removes a reward of ε. On the other hand, for any vertex $v_{\text{in}} \in V_{\text{in}}$ or $v_{\text{out}} \in V_{\text{out}}$,

$$r_{\text{learner}}(v_{\text{in}},t) = -k + 2d_{\text{in}}(v_{\text{in}},t) \tag{32}$$

$$r_{\text{learner}}(v_{\text{out}},t) = -k + 2d_{\text{out}}(v_{\text{out}},t) \tag{33}$$

Lemma 20 *Suppose $\Delta \geqslant \frac{1}{\eta}\log\left(2n^2T_{init}\right)$. If at any time $t > 0$, $V_{excess}(\Delta,t)$ is nonempty, then, the optimizer can collect at most $\frac{1}{T}$ the next round.*

B.1. Proof of Lemma 20

Recall that the reward of the actions of the learner at time t is as follows:

$$r_{\text{learner}}((w,x),t) = \begin{cases} E((w,x),t) - \varepsilon d_{\text{out}}(x,t) & \text{if } (w,x) \neq e^{\dagger}, \\ k - \varepsilon d_{\text{out}}(x,t) & \text{otherwise.} \end{cases} \tag{34}$$

$$r_{\text{learner}}(v_{\text{in}},t) = -k + 2d_{\text{in}}(v,t) \tag{35}$$

$$r_{\text{learner}}(v_{\text{out}},t) = -k + 2d_{\text{out}}(v,t) \tag{36}$$

Take $u \in V_{\text{excess}}(\Delta,t)$ for which $\max\{d_{\text{in}}(u,t), d_{\text{out}}(u,t)\}$ is maximized, and supposed without loss of generality that $d_{\text{in}}(u,t) = \max\{d_{\text{in}}(u,t), d_{\text{out}}(u,t)\}$ (the proof still goes through if $d_{\text{out}}(u,t) = \max\{d_{\text{in}}(u,t), d_{\text{out}}(u,t)\}$ in an identical way). We claim that for each action (w,x) of the learner in E we have:

$$r_{\text{learner}}(u_{\text{in}},t) \geqslant \Delta + r_{\text{learner}}((w,x),t)$$

Notice that

$$r_{\text{learner}}(u_{\text{in}},t) = -k + 2d_{\text{in}}(u,t) = -k + d_{\text{in}}(u,t) + d_{\text{in}}(u,t) \tag{37}$$

$$\geqslant -k + d_{\text{in}}(u,t) + k + \Delta = d_{\text{in}}(u,t) + \Delta \tag{38}$$

We also have for $(w,x) \neq e^{\dagger}$

$$r_{\text{learner}}((w,x),t) = E((w,x),t) - \varepsilon d_{\text{out}}(x,t) \leqslant E((w,x),t) \leqslant d_{\text{in}}(x,t) \leqslant d_{\text{in}}(u,t) \tag{39}$$

and for $e^{\dagger}$

$$r_{\text{learner}}(e^{\dagger},t) < k < d_{\text{in}}(u,t) \tag{40}$$

thus the claim is proven. Now let us look at the probability mass the learner will put on actions of $V_{\text{in}} \cup V_{\text{out}}$ in the next round, compared to the actions of the learner in E. Let us denote the latter probability with p_E. We have:

$$p_E = \frac{\sum_{(w,x)\in E} \exp\left(\eta r_{\text{learner}}((w,x),t)\right)}{\sum_{(w,x)\in E} \exp\left(\eta r_{\text{learner}}((w,x),t)\right) + \sum_{v\in V} \exp\left(\eta r_{\text{learner}}(v_{\text{in}},t)\right) + \exp\left(\eta r_{\text{learner}}(v_{\text{out}},t)\right)} \tag{41}$$

We have that the probability mass of the actions in $V_{\text{in}} \cup V_{\text{out}}$ is at least:

$$\sum_{v \in V} \exp\left(\eta r_{\text{learner}}(v_{\text{in}}, t)\right) + \exp\left(\eta r_{\text{learner}}(v_{\text{out}}, t)\right) > \exp\left(\eta r_{\text{learner}}(u_{\text{in}}, t)\right) \geqslant \tag{42}$$

$$\geqslant \exp\left(\eta d_{\text{in}}(u, t)\right) \cdot \exp\left(\eta \Delta\right) = \exp\left(\eta d_{\text{in}}(u, t)\right) \cdot \exp\left(\log\left(2n^2 T_{\text{init}}\right)\right) \tag{43}$$

$$= 2\exp\left(\eta d_{\text{in}}(u, t)\right) \cdot n^2 T_{\text{init}} \tag{44}$$

while the probability mass in the actions in E are:

$$\sum_{(w,x) \in E} \exp\left(\eta r_{\text{learner}}((w, x), t)\right) \leqslant \sum_{(w,x) \in E} \exp\left(\eta d_{\text{in}}(u, t)\right) < n^2 \exp\left(\eta d_{\text{in}}(u, t)\right) \tag{45}$$

Thus we get that:

$$p_E \leqslant \frac{\exp\left(\eta d_{\text{in}}(u, t)\right) \cdot n^2}{\exp\left(\eta d_{\text{in}}(u, t)\right) \cdot n^2 + 2\exp\left(\eta d_{\text{in}}(u, t)\right) \cdot n^2 T_{\text{init}}} \leqslant \frac{1}{2T_{\text{init}} + 1} < \frac{1}{2T_{\text{init}}} \tag{46}$$

Since the optimizer can earn reward only from actions in E, his reward in the next round will be upper bounded by $\frac{1}{T_{\text{init}}}$ as wanted.

B.2. Proof of Corollary 7

This follows directly from Lemma 20. Since at time $t_{\max}$ we have that V_{excess} is non-empty, that means that for rounds $t_{\max} + 1, t_{\max} + 2, \ldots, T_{\text{init}}$ the optimizer can earn reward at most $\frac{1}{T_{\text{init}}}$. Summing up gives that the total reward that can be obtained by the optimizer:

$$R(t_{\max}) + \frac{T_{\text{init}} - t_{\max}}{T_{\text{init}}} < R(t_{\max}) + 1$$

as desired.

B.3. Proof of Lemma 21 and Corollary 22

The following results will be useful for the proof of lemma 9:

Lemma 21 *At any time t, there will always exist an action for the learner having cumulative reward at least $k(1 - \varepsilon)$.*

Proof of Lemma 21 Notice that initially, we have $r_{\text{learner}}(e^\dagger, 0) = r_{\text{learner}}((u^\dagger, v^\dagger), 0) = k$. At some time t we will have that $r_{\text{learner}}(e^\dagger, t) = k - \varepsilon d_{\text{in}}(v^\dagger, t)$. At time t we also have for the action $v^\dagger_{\text{in}}$ that, $r_{\text{learner}}(v^\dagger_{\text{in}}, t) = -k + 2d_{\text{in}}(v^\dagger, t)$. Now consider $\max(r_{\text{learner}}(e^\dagger, t), r_{\text{learner}}(v^\dagger_{\text{in}}, t)$. This quantity is always greater than $k(1 - \varepsilon)$, thus one of the two actions will always have reward greater than $k(1 - \varepsilon)$, which proves the claim.

Corollary 22 *Consider any edge $(u, v) \neq (u^\dagger, v^\dagger)$ which was chosen fewer than $k(1 - \varepsilon) - \Delta$ times until time t by the optimizer. MWU places mass of at most $1/n^2 T_{init}$ on the corresponding learner edge (u, v).*

Proof of Corollary 22 By lemma 21 we know that at any time t, there exists an action for the learner having cumulative reward at least $k(1-\varepsilon)$ and we know that action is not (u,v). The probability mass MWU will then place on action (u,v) will be at most,

$$\frac{\exp\big(\eta(k(1-\varepsilon)-\Delta)\big)}{\exp\big(\eta(k(1-\varepsilon)-\Delta)\big)+\exp\big(\eta k(1-\varepsilon)\big)} = \frac{1}{1+\exp\big(\eta\Delta\big)} < \frac{1}{1+2n^2T_{\text{init}}} < \frac{1}{n^2T_{\text{init}}} \tag{47}$$

as required.

B.4. Proof of Lemma 9

Consider actions/edges (v,w) played by the optimizer, that contribute to his reward but do not contribute to $r_{\text{heavy}}(\Delta,t)$, i.e., where $v \notin S_{\text{heavy}}(\Delta,t)$. We will try to upper bound the contribution of these actions to the optimizer's rewards. If we take a look at the utility matrix of the optimizer, when the optimizer is plays such an action reward can be obtained in only two cases:

1. The first case corresponds to when the learner plays an edge of the form (u,v) for some $u \in V$. Since $v \notin S_{\text{heavy}}(\Delta,t)$, no edge that is incident to v has been played enough to be considered heavy, i.e., $Q((u,v),t_{\max}) \leqslant k(1-\varepsilon)-\Delta$. Since this action has not been played enough, it will be dominated by the action that has reward at least $k(1-\varepsilon)$ as explained by Lemma 21, and therefore MWU will allocate very little probability mass on it and we can upper bound its contribution. Let us call the total reward the optimizer earns in this case, i.e., the reward the optimizer earns when playing edges of the form (v,w) when the learner plays edges of the form (u,v), be R_1.

2. The second type of reward, results from when the learner also plays the same edge (v,w). As the optimizer plays (v,w) more and more, the equivalent action (v,w) of the learner begins to accumulates reward. However, this reward will be relevant only when the edge (v,w) has been played enough times from the optimizer that (v,w) is no longer dominated by the action that has reward at least $k(1-\varepsilon)$ (cf. Lemma 21); this is the point when the combination $((v,w),(v,w))$ will start earning reward for the optimizer. However, note that the optimizer is constrained in playing (v,w) a limited number of times, specifically less than $k+\Delta$ times since we are considering times $t \leqslant t_{\max}$, thus the reward in this case for the optimizer will also be limited. Let us denote the total reward the optimizer earns in this case, i.e. the reward the optimizer earns when playing edges of the form (v,w) and the learner plays edges of the form (v,w) as R_2.

Let us now look at the reward the optimizer can earn in the above two cases. We make two claims; firstly, that $R_1 \leqslant 2$ and secondly that $R_2 \leqslant 2n(k\varepsilon+2\Delta)+1$. Putting the two claims together gives that:

$$r_{\text{optimizer}}(t_{\max}) \leqslant r_{\text{heavy}}(\Delta,t_{\max}) + R_1 + R_2 \leqslant r_{\text{heavy}}(\Delta,t_{\max}) + 2n(k\varepsilon+2\Delta)+3 \tag{48}$$

as desired. It remains to prove the two claims. For the first claim, note that the actions of the form (u,v) will have $r_{\text{learner}}((u,v),t) \leqslant k(1-\varepsilon)-\Delta$, since by assumption, $v \notin S_{\text{heavy}}(\Delta,t)$. Using Corollary 22 we can deduce that the mass assigned to these actions is upper bounded by $\frac{1}{n^2T_{\text{init}}}$. There can be at most n^2 such choices of actions, and all these actions, across all time steps can earn reward at most,

$$R_1 \leqslant 2 \cdot n^2T_{\text{init}} \cdot \frac{1}{n^2T_{\text{init}}} \leqslant 2 \tag{49}$$

which proves the first claim. For the second claim, take again an edge (v, w) for the optimizer that is played more than $k(1-\varepsilon)-\Delta$. Note that for a fixed choice of v, there is only at most one edge (v, w) that is played more than $k(1-\varepsilon)-\Delta$ times. If there were more than one, then $d_{\text{out}}(v, t)$ would exceed $k+\Delta$ contradicting that we are in time $t \leqslant t_{\max}$. We thus want to upper bound the reward earned by the optimizer when playing (v, w). In the worst case the optimizer plays the action (v, w), $k+\Delta$ times. Let us split the reward earned by that action in two time periods; the reward of the optimizer the first $k(1-\varepsilon)-\Delta$ the optimizer plays the action and the last $k\varepsilon+2\Delta$ times. In the first interval, at times when the optimizer plays (v, w), note that the learner has not built significant reward for this action (v, w) action yet, specifically $r_{\text{learner}}((v, w), t) \leqslant k(1-\varepsilon)-\Delta$. Since when playing (v, w) the optimizer earns reward only when the learner is playing the same action. According again to Corollary 22, the learner will place mass at most $\frac{1}{n^2 T_{\text{init}}}$ in this action, and therefore the optimizer can earn reward at most $\frac{1}{n^2 T_{\text{init}}} \cdot 2 \cdot (k(1-\varepsilon)-\Delta) < \frac{1}{n^2} \cdot 2 \cdot T_{\text{init}} = \frac{2}{n^2}$. For the second part, the optimizer can earn reward at most 2 for the remaining $\varepsilon k + 2\Delta$ rounds, thus making the total reward for the specific action $\frac{2}{n^2} + \varepsilon k + 2\Delta$. Summing up over all possible v's we get that the total reward R_2 for the optimizer is:

$$R_2 \leqslant n \cdot \left(\frac{2}{n^2} + \varepsilon k + 2\Delta\right) \leqslant 1 + n(\varepsilon k + 2\Delta) \tag{50}$$

which proves the second claim and therefore the lemma.

B.5. Proof of Lemma 11

Since C_i is assumed to be a path, the vertex z_i has no incident edge in G'. We will prove that the following quantity is an upper bound on the total reward the optimizer collects via playing edges emanating from vertices in C_i,

$$\underbrace{\sum_{i=2}^{m-1} \left\{\alpha_i W(z_i, z_{i+1}) + 2(k+\Delta-\alpha_i)\right\}}_{(i)} + \underbrace{2\sum_{v \in V} Q((z_m, v), t_{\max})}_{(ii)} + \underbrace{2(k+\Delta)\mathbb{I}(i=1)}_{(iii)} \tag{51}$$

where $\alpha_i \geqslant k(1-\varepsilon)-\Delta$ for all i. We will later simplify this bound to prove the lemma. The first term (i) can be attributed to the fact that for every vertex in the path z_{i+1} for $i \geqslant 1$, there is a high number of edges from the previous vertex in the path, z_i. The number of such edges is captured by $\alpha_i \geqslant k(1-\varepsilon)-\Delta$. This vertex z_{i+1} may have $(k+\Delta)-\alpha_i$ remaining edges emanating out from it, and these edges may collect the maximum possible reward of 2 (since each edge weight is in $\{1, 2\}$). Thus the total reward for edges coming out of z_{i+1} is $\alpha_i W(z_i, z_{i+1}) + 2(k+\Delta-\alpha_i)$ The term (ii) accounts for the reward the last vertex in the path, z_m, can collect from edges emanating out from it. Since $z_m \in S_{\text{heavy}}(\Delta, t_{\max})$ every edge emanating from it can collect a reward of at most 2. Term (iii) accounts for the fact that specifically for C_1, the vertex $v^\dagger$ is always in S_{heavy} by definition. The reason for this choice is that even if there are no in-edges to $v^\dagger$, MWU associates high initial reward to the edge $(u^\dagger, v^\dagger)$ (so this vertex "behaves" as though the optimizer had chosen many in-edges incident on it). Simplifying eq. (51) further, assuming the in-degree bound of $k+\Delta$ on vertices, we arrive at the following upper bound on the total reward collected by the optimizer from playing

edges emanating from $\{z_1, \cdots, z_m\}$,

$$\sum_{i=2}^{m-1} \underbrace{\Big\{(k(1-\varepsilon)-\Delta)W(z_i, z_{i+1}) + 2(k\varepsilon + 2\Delta)\Big\}}_{(i)} + \underbrace{2(k+\Delta)}_{(ii)} + \underbrace{2(k+\Delta)\mathbb{I}(i=1)}_{(iii)} \tag{52}$$

$$\leqslant \sum_{i=2}^{m-1} (k(1+\varepsilon)+3\Delta)W(z_i, z_{i+1}) + 2(k+\Delta) + 2(k+\Delta)\mathbb{I}(i=1) \tag{53}$$

$$\leqslant \sum_{i=1}^{m} (k(1+\varepsilon)+3\Delta)W(z_i, z_{i+1}) + 2(k+\Delta)\mathbb{I}(i=1). \tag{54}$$

Where in the last inequality we upper bound the middle term $2(k+\Delta) \leqslant (k(1+\varepsilon)+3\Delta)W(z_1, z_2) + (k(1+\varepsilon)+3\Delta)W(z_m, z_{m+1})$ for any arbitrary vertex z_{m+1}, since $W(\cdot,\cdot)$ is pointwise in $\{1,2\}$. This results in an the overall upper bound on the total reward collected from all edges the optimizer plays emanating from the vertices $z_1, z_2, \cdots, z_m$.

B.6. Proof of Lemma 12

Consider the smallest time t_i for each i at which the vertices z_i first appear in $S_{\text{heavy}}(\Delta, t_i)$. As the optimizer plays edges, suppose the vertex z_{i^*} is the last to appear last in S_{heavy} among all the z_i's in this cycle. For all the edges in the graph G' that belong in this cycle, we can bound their contribution in a similar way we did in Lemma 11, except the contribution coming from edges emanating from z_{i^*}. This contribution is:

$$(k(1+\varepsilon)+3\Delta) \sum_{i \in [m] \setminus \{i^\star\}} W(z_i, z_{i+1}) + 2(k+\Delta)\mathbb{I}(i=1) \tag{55}$$

It ultimately remains to bound the contribution of the actions emanating from i^*. We will prove that the first $k(1-\varepsilon)-\Delta$ times the optimizer plays action (z_{i^*}, z_{i^*+1}) very little reward is collected. Notice that the optimizer can earn reward either if the learner places probability mass on the same edge or on an edge that is of the form $(\cdot, z_{i^*})$. For the first case, the maximum reward the optimizer can obtain is:

$$\sum_{j=0}^{k(1-\varepsilon)-\Delta} \frac{\exp(\eta j)}{\exp(\eta j) + \exp\big(\eta(k(1-\varepsilon)-\Delta)\big)} \leqslant (k(1-\varepsilon)-\Delta) \cdot \exp\big(-\eta\Delta\big) \leqslant \frac{1}{n^2} \tag{56}$$

For the second case, notice that each edge $(\cdot, z_{i^*})$ has been played at most $k(1-\varepsilon)-\Delta$ by the optimizer, since i^* is the last node to become heavy, thus meaning that the learner will not place a lot of mass on these edges. This bounds the reward of the optimizer as follows:

$$n \cdot 2 \cdot (k(1-\varepsilon)-\Delta) \cdot \frac{\exp\big(\eta(k(1-\varepsilon)-\Delta)\big)}{\exp\big(\eta(k(1-\varepsilon)-\Delta)\big) + \exp\big(\eta(k(1-\varepsilon))\big)} < n \cdot 2 \cdot (k(1-\varepsilon)-\Delta) \cdot \exp\big(\eta\Delta\big) < \frac{1}{2n} \tag{57}$$

For the remaining at most $\varepsilon k + 2\Delta$ times the optimizer plays an edge emanating from z_{i^*} we may assume the maximum reward of 2 is collected. The optimizer plays at most $\varepsilon k + 2\Delta$ such edges, and thereby earns reward at most $2(\varepsilon k + 2\Delta)$ in this case. This bounds the reward emanating from z_{i^*} to:

$$\frac{1}{n^2} + \frac{1}{2n} + 2(\varepsilon k + 2\Delta) < \frac{1}{n} + 2(\varepsilon k + 2\Delta) \tag{58}$$

and thus the total reward that can be earned from the cycle C_i is:

$$(k(1+\varepsilon)+3\Delta)\sum_{i\in[m]\setminus\{i^\star\}} W(z_i,z_{i+1}) + 2(k+\Delta)\mathbb{I}(i=1) + \frac{1}{n} + 2(\varepsilon k + 2\Delta) < \tag{59}$$

$$(k(1+3\varepsilon)+7\Delta)\sum_{i\in[m]\setminus\{i^\star\}} W(z_i,z_{i+1}) + 2(k+\Delta)\mathbb{I}(i=1) + \frac{1}{n} \tag{60}$$

as required.

B.7. Proof of Lemma 13

We will show that the total value collected by the optimizer is at most,

$$(1) + (2n(k\varepsilon+2\Delta)+3) + \left((k(1+3\varepsilon)+7\Delta)\sum_{i=1}^{n} W(z_i,z_{i+1}) + 2(k+\Delta)+1\right) \tag{61}$$

which is upper bounded by the quantity in the statement of the lemma. Let us recall the different ways the optimizer collects rewards.

1. If time $t_{\max}$ is reached, by Corollary 7 we argued that then onward, the optimizer will collect reward at most 1. This accounts for the first term in eq. (61).

2. From time 0 up until time $t_{\max}$, we argued through Lemma 9 that only at most $2n(k\varepsilon+2\Delta)$ reward can come from edges whose vertices do not emanated from vertices in $S_{\text{heavy}}(\Delta,t)$. This accounts for the second term in Eq. 61.

3. Finally, we use Lemmas 11 and 12 to argue that the optimizer can make at most $(k(1+3\varepsilon)+7\Delta)\sum_{i=1}^{n} W(z_i,z_{i+1})+2(k+\Delta)+1$ where $z_1\to z_2\to\cdots\to z_n\to z_1$ is some Hamiltonian cycle. Suppose the optimizer's play induces the graph G' as defined previously. We will prove that the paths and cycles formed can be combined in order to create a Hamiltonian cycle, the weight of which will upper bound (up to some multiplicative and additive constants) the reward that can be obtained in this scenario by the optimizer. Suppose $C_1, C_2, \ldots, C_l$ are all paths and $C_{l+1}, C_{l+2}, \ldots, C_t$ are cycles, and C_1 contains the special edge e^+ - we will later argue how to prove the claim even if the connected component of the special edge is a cycle. We first combine all the paths into one. Suppose path C_i is $z_1^i\to z_2^i\to\cdots\to z_{m_i}^i$. We will combine the paths by attaching the last vertex of path C_i to the first vertex of path C_{i+1}, i.e. $z_{m_i}^i\to z_1^{i+1}$. The combination of all paths will then be:

$$z_1^1\to z_2^1\to\cdots\to z_{m_1}^1\to z_1^2\to z_2^2\to\ldots z_{m_2}^2\to\cdots\to z_1^l\to\cdots\to z_{m_l}^l$$

If we sum up the rewards of the paths independently and use Lemma 11, the paths will earn rewards at most

$$(k(1+\varepsilon)+3\Delta)\sum_{i=1}^{l}\sum_{j=1}^{m_i} W(z_j^i,z_{j+1}^i) + 2(k+\Delta)$$

while the newly constructed path will earn reward

$$(k(1+\varepsilon)+3\Delta)\sum_{i=1}^{l}\left(\left(\sum_{j=1}^{m_i-1} W(z_j^i,z_{j+1}^i)\right) + \mathbb{I}[i\neq l]W(z_{m_i}^i,z_1^{i+1})\right) + 2(k+\Delta)$$

which is a clear upper bound. Now we continue by combining the cycles. Suppose for cycle C_i where $t \geqslant i > l$ and the cycle is $z_1^i \to z_2^i \to \cdots \to z_{m_i}^i \to z_1^i$ the vertex i^* as described in Lemma 12 is denoted by the vertex $z_{m_i}^i$. We will combine the cycles as follows, to create a big cycle:

$$z_1^l \to z_2^l \to \cdots \to z_{m_l}^l \to z_1^{l+1} \to \ldots z_{m_{l+1}}^{l+1} \to \cdots \to z_{m_t}^t \to z_1^l$$

We essentially delete the last edge of each cycle and redirect the last vertex to the first vertex of the next cycle. Note that the reward of this big cycle according to Lemma 12 is:

$$(k(1+3\varepsilon)+7\Delta)\sum_{i=l+1}^{t}\left(\left(\sum_{j=1}^{m_i-1}W(z_j^i,z_{j+1}^i)\right)+\mathbb{I}[i\neq t]W(z_{m_i}^i,z_1^{i+1})\right) \tag{62}$$

which is upper bounding the rewards -when adding a constant 1 to cover for $\frac{t}{n}$ - of the individual cycles, which is:

$$(k(1+3\varepsilon)+7\Delta)\sum_{i=l+1}^{t}\sum_{j=1}^{m_i-1}W(z_j^i,z_{j+1}^i)+\frac{t}{n} \tag{63}$$

Lastly, now we connect the big path and the big cycle together as follows: direct $z_{m_l}^l \to z_1^{l+1}$ and redirect $z_{m_t}^t \to z_1^1$. This will close the cycle and upper bound the rewards obtained by the paths and cycles together. The reward of the Big cycle will then be:

$$(k(1+3\varepsilon)+7\Delta)\sum_{i=1}^{n}W(z_i,z_{i+1})+2(k+\Delta)+1$$

as we wanted. In the case where C_1 was not a path but a cycle, we would have begun with the cycles instead of the paths and do the same thing with the only difference being that we would move over the $2(k+\Delta)$ factor to the cycles instead of the paths.

B.8. Proof of Lemma 14

Without loss of generality assume $z_1^\star = v^\dagger$ and we define $z_{-1}^\star = u^\dagger$ and $z_{n+1}^\star = z_1^\star$. Define Z as the set of edges $\{(z_{i-1}^\star, z_i^\star) : i \in [n+1]\}$. Consider the sequence of optimizer pure strategies which plays the edge $(z_i^\star, z_{i+1}^\star)$ for $k(1-\varepsilon+\varepsilon^2)$ rounds for each i, and repeats across $i \in [n]$ (assuming $z_{n+1}^\star = z_1^\star$). For the last $nk(\varepsilon-\varepsilon^2)$ we lower bound the rewards of the optimizer by 0 - basically disregarding those rounds. The total reward collected by this sequence of pure strategies is lower bounded next. At initialization, MWU associates a cumulative reward of k with the action $(u^\dagger, v^\dagger)$ and $\leqslant 0$ for the remaining actions. We argue that in each epoch $i \leqslant n-1$, the total mass MWU places on the actions $(z_{i-1}^\star, z_i^\star)$ and $(z_i^\star, z_{i+1}^\star)$ is close to 1 in most of the iterations within this epoch.

At the end of the i^{th} epoch (i.e., $t = k(1-\varepsilon+\varepsilon^2)\times i + 1$), the cumulative rewards on the actions are,

$$B_0(b) + \sum_{s=1}^{t} B((u_s, v_s), b) = \begin{cases} k(1-\varepsilon)(1-\varepsilon+\varepsilon^2) & \text{if } b = (z_j^\star, z_{j+1}^\star) \text{ for } j \leqslant i-1, \\ k(1-\varepsilon+\varepsilon^2) & \text{if } b = (z_i^\star, z_{i+1}^\star), \\ 0 & \text{if } b = (z_j^\star, z_{j+1}^\star) \text{ for } j \geqslant i+1, \\ 0 & \text{if } b \in E \setminus Z, \\ k(1-2\varepsilon+2\varepsilon^2) & \text{if } b = (z_j^\star)' \text{ for } j \leqslant i \\ -k & \text{if } b = (z_j^\star)' \text{ for } j \geqslant i+1 \\ k(1-2\varepsilon+2\varepsilon^2) & \text{if } b = (z_j^\star)'' \text{ for } j \leqslant i+1 \\ -k & \text{if } b = (z_j^\star)'' \text{ for } j \geqslant i+2 \end{cases} \tag{64}$$

Here, for $v \in V$, we denote $(v)'$ as its corresponding vertex in V_{out}, and $(v)''$ as its corresponding vertex in V_{in}. In the $(i+1)^{\text{th}}$ epoch, the edge $(z_{i+1}^\star, z_{i+2}^\star)$ is chosen a number of times. Having chosen it p times thus far, the cumulative rewards of actions are updated as,

$$B_0(b) + \sum_{s=1}^{t} B((u_s, v_s), b) = \begin{cases} k(1-\varepsilon)(1-\varepsilon+\varepsilon^2) & \text{if } b = (z_j^\star, z_{j+1}^\star) \text{ for } j \leqslant i-1, \\ k(1-\varepsilon+\varepsilon^2) - \varepsilon p & \text{if } b = (z_i^\star, z_{i+1}^\star), \\ p & \text{if } b = (z_{i+1}^\star, z_{i+2}^\star), \\ 0 & \text{if } b = (z_j^\star, z_{j+1}^\star) \text{ for } j \geqslant i+2, \\ 0 & \text{if } b \in E \setminus Z, \\ k(1-2\varepsilon+2\varepsilon^2) & \text{if } b = (z_j^\star)' \text{ for } j \leqslant i \\ -k+2p & \text{if } b = (z_{i+1}^\star)' \\ -k & \text{if } b = (z_j^\star)' \text{ for } j \geqslant i+2 \\ k(1-2\varepsilon+2\varepsilon^2) & \text{if } b = (z_j^\star)'' \text{ for } j \leqslant i \\ -k+2p & \text{if } b = (z_{i+2}^\star)'' \\ -k & \text{if } b = (z_j^\star)'' \text{ for } j \geqslant i+3 \end{cases} \tag{65}$$

It is easy to verify plugging in $p = k(1-\varepsilon+\varepsilon^2)$, we arrive at the cumulative rewards at the end of the $(i+1)^{\text{th}}$ epoch, which matches what is observed plugging in $i = i+1$ in eq. (64).

In the $(i+1)^{\text{th}}$ epoch, for any $p \leqslant k(1-\varepsilon+\varepsilon^2)$, observe that the total mass MWU associates with the edges $(z_i^\star, z_{i+1}^\star)$ and $(z_{i+1}^\star, z_{i+2}^\star)$ is lower bounded by,

$$\frac{e^{\eta(k(1-\varepsilon+\varepsilon^2)-\varepsilon p)} + e^{\eta p}}{e^{\eta(k(1-\varepsilon+\varepsilon^2)-\varepsilon p)} + e^{\eta p} + (|\mathcal{B}_{\text{init}}| - 2)e^{\eta k(1-2\varepsilon+2\varepsilon^2)}} \tag{66}$$

This uses the fact that for the learner, the maximum cumulative reward on any action which is not one of $(z_i^\star, z_{i+1}^\star)$ or $(z_{i+1}^\star, z_{i+2}^\star)$ is upper bounded by $\max\{k(1-\varepsilon)(1-\varepsilon+\varepsilon^2), k(1-2\varepsilon+2\varepsilon^2)\} \leqslant k(1-2\varepsilon+2\varepsilon^2)$ assuming that ε is sufficiently small and ε is bounded away from 1. For any value of $p \leqslant (1-\varepsilon)^2 k$, using the fact that $|\mathcal{B}_{\text{init}}| \leqslant 2\binom{n}{2} + 2n = n^2 + n$, eq. (66) is further lower bounded by,

$$\frac{e^{\eta(k(1-\varepsilon+\varepsilon^2)-\varepsilon p)} + e^{\eta p}}{e^{\eta(k(1-\varepsilon+\varepsilon^2)-\varepsilon p)} + e^{\eta p} + (|\mathcal{B}_{\text{init}}| - 2)e^{\eta k(1-2\varepsilon+2\varepsilon^2)}} = \left(\frac{e^{\eta(k(1-\varepsilon+\varepsilon^2)-\varepsilon p)} + e^{\eta p} + (|\mathcal{B}_{\text{init}}| - 2)e^{\eta k(1-2\varepsilon+2\varepsilon^2)}}{e^{\eta(k(1-\varepsilon+\varepsilon^2)-\varepsilon p)} + e^{\eta p}}\right)^{-1}$$

$$\geqslant \left(1+\frac{(n^2+n)e^{\eta k(1-2\varepsilon+2\varepsilon^2)}}{e^{\eta(k(1-\varepsilon+\varepsilon^2)-\varepsilon p)}+e^{\eta p}}\right)^{-1} \geqslant 1-\frac{(n^2+n)e^{\eta k(1-2\varepsilon+2\varepsilon^2)}}{e^{\eta(k(1-\varepsilon+\varepsilon^2)-\varepsilon p)}+e^{\eta p}} \geqslant 1-\frac{(n^2+n)e^{\eta k(1-2\varepsilon+2\varepsilon^2)}}{e^{\eta(k(1-\varepsilon+\varepsilon^2)-\varepsilon p)}}$$
$$\geqslant 1-(n^2+n)e^{\eta(k(-\varepsilon+\varepsilon^2)+\varepsilon p)} \geqslant 1-(n^2+n)e^{\eta(k(-\varepsilon+\varepsilon^2)+\varepsilon(1-\varepsilon)^2 k)} = 1-(n^2+n)e^{\eta k(-\varepsilon+\varepsilon^2+\varepsilon-2\varepsilon^2+\varepsilon^3)}$$
$$= 1-(n^2+n)e^{-\eta(\varepsilon-\varepsilon^2)\varepsilon k}.$$

Note that $(\varepsilon-\varepsilon^2)\eta\varepsilon k \geqslant \log(n^3)$, by assumption, thus the total probability MWU associates with the edges $(z_i^\star, z_{i+1}^\star)$ and $(z_{i+1}^\star, z_{i+2}^\star)$ in the first $\Delta_1 \triangleq (1-\varepsilon^2)k$ steps of the $(i+1)^{\text{th}}$ epoch is at least $1-2/n$.

Note that in the $(i+1)^{\text{th}}$ epoch, the optimizer always plays the edge $(z_{i+1}^\star, z_{i+2}^\star)$. This edge collects reward for the optimizer based on the probability mass MWU associates with edges of the form $(\cdot, z_{i+1}^\star)$ and with the edge $(z_{i+1}^\star, z_{i+2}^\star)$. MWU places at least $1-2/n$ mass on these kinds of edges in the first Δ_1 iterations of this epoch. Therefore, in the $(i+1)^{\text{th}}$ epoch, the optimizer collects reward of at least,

$$\Delta_1\left(1-\frac{2}{n}\right)W(z_{i+1}^\star, z_{i+2}^\star) \tag{67}$$

Summing this over all values of $i = 0, 1, \cdots, n-1$, the total reward collected by the optimizer is at least,

$$\Delta_1\left(1-\frac{2}{n}\right)\sum_{i=1}^{n} W(z_i^\star, z_{i+1}^\star) \tag{68}$$

Up to a multiplicative factor of $\Delta_1(1-2/n) = (1-\varepsilon)^2 k(1-\frac{2}{n}) \approx k$, the reward collected by this sequence of optimizer pure strategies matches the weight of the heaviest Hamiltonian cycle in G.

B.9. Proof of Theorem 19

We set the parameters as follows: $T_{\text{init}} = \varepsilon T, n = T^{\min\{\alpha,\beta\}/3}, \eta = 1/T^{1-\alpha}, k = \varepsilon T^{1-\min\{\alpha,\beta\}/3}, \Delta = T^{1-\alpha}\log(2n^2T)$. Notice that $nk = T_{\text{init}}$, $k \gg \Delta$. We also have $\eta\varepsilon k = T^{\alpha-1}\cdot\varepsilon\cdot\varepsilon T^{1-\min\{\alpha,\beta\}/3} \geqslant \varepsilon^2 T^{2\alpha/3}$ and since ε is a constant we get $\eta\varepsilon k \geqslant \frac{\varepsilon-\varepsilon^2}{\log}(n^3)$ for large enough T. Thus, the assumptions of Lemmas 14 and 13 are satisfied. Suppose the maximum reward that can be obtained by the optimizer is R_{opt}, and is upper bounded by Lemma 13,

$$R_{\text{opt}} \leqslant (k(1+5\varepsilon)+9\Delta)\sum_e W(e) + (2k+2\Delta+4) \leqslant (k(1+5\varepsilon)+11\Delta+4)\sum_e W(e) + (2k+4)$$

We know that the best optimizer achieves reward at least R_{lb}, which is given by Lemma 14,

$$R_{\text{lb}} \geqslant k(1-\varepsilon)^2\left(1-\frac{2}{n}\right)\sum_{i=1}^{n} W(z_i^\star, z_{i+1}^\star)$$

Suppose $R_\star$ is the optimal reward the optimizer can get in this instance. We will prove that if the optimizer can guarantee that $\frac{R_{\text{opt}}}{R_\star} \geqslant \rho$ for some $\rho > \frac{1067}{1068}$, then it can also guarantee a ρ approximation for the maxTSP problem: this contradicts Theorem 5 which shows that unless $\mathsf{P} = \mathsf{NP}$, there is no polynomial time approximation algorithm for $(1,2)$-maxTSP achieving an

approximation factor of $1067/1068+\varepsilon$ for any $\varepsilon>0$. Taking the ratio now, and note that $R_\star \geqslant R_{\text{lb}}$. Thus:

$$\frac{R_{\text{opt}}}{R_\star} \geqslant \frac{R_{\text{opt}}}{R_{\text{lb}}} \geqslant \frac{(k(1+5\varepsilon)+11\Delta+4)\sum_e W(e)+(2k+4)}{k(1-\varepsilon)^2\left(1-\frac{2}{n}\right)\sum_{i=1}^n W(z_i^\star,z_{i+1}^\star)} = \tag{69}$$

$$\frac{(k(1+5\varepsilon)+11\Delta+4)\sum_e W(e)}{k(1-\varepsilon)^2\left(1-\frac{2}{n}\right)\sum_{i=1}^n W(z_i^\star,z_{i+1}^\star)} + \frac{2k+4}{k(1-\varepsilon)^2\left(1-\frac{2}{n}\right)\sum_{i=1}^n W(z_i^\star,z_{i+1}^\star)} \geqslant \tag{70}$$

$$\geqslant \frac{(k(1+5\varepsilon)+11\Delta+4)\sum_e W(e)}{k(1-\varepsilon)^2\left(1-\frac{2}{n}\right)\sum_{i=1}^n W(z_i^\star,z_{i+1}^\star)} \tag{71}$$

$$= \left(1+\frac{(k(1+5\varepsilon)+11\Delta+4)-k(1-\varepsilon)^2\left(1-\frac{2}{n}\right)}{k(1-\varepsilon^2)\left(1-\frac{2}{n}\right)}\right)\frac{\sum_e W(e)}{\sum_{i=1}^n W(z_i^\star,z_{i+1}^\star)} \geqslant \rho \tag{72}$$

Now, all that is left to do it prove that $\left(1+\frac{(k(1+5\varepsilon)+11\Delta+4)-k(1-\varepsilon)^2\left(1-\frac{2}{n}\right)}{k(1-\varepsilon^2)\left(1-\frac{2}{n}\right)}\right)^{-1}\rho \geqslant \rho_{TSP}$, or equivalently that the ratio can get arbitrarily close to 1. We can therefore just focus on making $\frac{(k(1+5\varepsilon)+11\Delta+4)-k(1-\varepsilon)^2\left(1-\frac{2}{n}\right)}{k(1-\varepsilon)^2\left(1-\frac{2}{n}\right)}$ go as close to 0 as possible. As we take ε to be very close to 0.

$$\frac{(k(1+5\varepsilon)+11\Delta+4)-k(1-\varepsilon)^2\left(1-\frac{2}{n}\right)}{k(1-\varepsilon)^2\left(1-\frac{2}{n}\right)} = \frac{11\Delta+4-k\varepsilon^2+7k\varepsilon+\frac{2k}{n}(1-\varepsilon)^2}{k(1-\varepsilon)^2\left(1-\frac{2}{n}\right)} \tag{73}$$

$$= O\left(\frac{\Delta+\frac{2k}{n}}{k}\right) = O\left(\frac{\Delta}{k}+\frac{1}{n}\right) \tag{74}$$

The above term approaches 0, since $k/\Delta \geqslant \widetilde{\Omega}(T^{2\alpha/3})$. This concludes the proof.

Appendix C. Optimizing against MWU without initialization: Proof of Theorem 1

C.1. A comment on the initialization

Note that the proof in the previous section considers the specific initialization of the form eqs. (10) to (13). Consider a sequence of (non-distinct) edges E_{init} on the vertex set V and let $d_{\text{init,in}}$ and $d_{\text{init,out}}$ denote the induced in-degrees and out-degrees. Consider a new initialization,

$$\text{For } v_{\text{in}} \in V_{\text{in}},\ r_0(v_{\text{in}}) = -k + 2d_{\text{init,in}}(v_{\text{in}}) \tag{75}$$

$$\text{For } v_{\text{out}} \in V_{\text{out}},\ r_0(v_{\text{out}}) = -k + 2d_{\text{init,out}}(v_{\text{out}}) \tag{76}$$

$$\text{For } e \in E \setminus \{e^\dagger\},\ r_0(e) = E_{\text{init}}(e) \tag{77}$$

$$r_0(e^\dagger) = k \tag{78}$$

When $E_{\text{init}} = \emptyset$, we go back to the one in eqs. (10) to (13). With such an initialization, the upper bound on the total reward of the optimizer strategy in Lemma 13 still remains true, where T_{init} is replaced by $T_{\text{init}}+|E_{\text{init}}|$ everywhere. The idea is simple: the initialization in eqs. (75) to (78) may be realized by starting out with the one in eqs. (10) to (13), and allowing the optimizer to first playing the edges in E_{init} in sequence (where no reward is collected), and then playing its original strategy. Thus, we may upper bound the reward collected by the optimizer by considering what it would have collected over the total horizon (including that collected when edges in E_{init} are played). This results in the following corollary of Lemma 13.

Corollary 23 *Suppose* $\Delta \geqslant \frac{1}{\eta}\log\left(2n^2(T_{init}+|E_{init}|)\right)$ *The total reward collected by an optimizer strategy when MWU's initialization is of the form eqs. (75) to (78) is upper bounded by,*

$$(k(1+5\varepsilon)+9\Delta)\sum_{i=1}^{n} W(z_i, z_{i+1}) + (2k+2\Delta+4) \tag{79}$$

for some Hamiltonian cycle $z_1 \to z_2 \to \cdots \to z_1 \to z_1$ *on* G.

C.2. Proof of Theorem 1

Definition 24 (Reduced MWU) *Consider some sequence of actions* $\{\boldsymbol{x}(t)\}_{t\geqslant 1}$ *played by the optimizer. Let* $\mathcal{T}_{\blacklozenge}(t)$ *denote the set of time-points until and including time* t *where the optimizer played* $\blacklozenge$ *and* $\mathcal{T}_{\times\blacklozenge}(t)$ *denote the set of time-points until and including time* t *where the optimizer did not play* $\blacklozenge$. *Let* $\boldsymbol{y}_{red}(t)$ *denote the "reduced MWU" where we assume that cumulative rewards of actions are computed across time points where* $\blacklozenge$ *is not played. Namely, using the definition,*

$$\forall b \in \mathcal{B}_{init},\ \widetilde{r}_{learner}(b,t) = \frac{k^\star}{T}|\mathcal{T}_{\blacklozenge}(t)|\cdot\mathbb{I}(b=e^\dagger) + \sum_{t'\in\mathcal{T}_{\times\blacklozenge}(t)} \boldsymbol{x}(t')^\top \boldsymbol{B}\delta_b \tag{80}$$

$$\forall b \in \widetilde{\mathcal{B}}_{init},\ \widetilde{r}_{learner}(\widetilde{b},t) = \frac{k^\star}{T}|\mathcal{T}_{\blacklozenge}(t)|\cdot\mathbb{I}(\widetilde{b}=\widetilde{e}^\dagger) + \sum_{t'\in\mathcal{T}_{\times\blacklozenge}(t)} \boldsymbol{x}(t')^\top \boldsymbol{B}\delta_{\widetilde{b}} \tag{81}$$

Lemma 25 *For any action* $b \in \mathcal{B}_{init}$,

$$\mathbf{Pr}_{Y\sim\boldsymbol{y}(t)}(Y=b) = \frac{1}{1+p\cdot\exp\left(-\eta\gamma|\mathcal{T}_{\blacklozenge}(t)|\right)}\cdot\mathbf{Pr}_{Y\sim\boldsymbol{y}_{red}(t)}(Y=b) \tag{82}$$

Proof Note that the cumulative rewards $r_{\text{learner}}(b,t)$ of MWU can be written as,

$$\forall b \in \mathcal{B}_{\text{init}},\ r_{\text{learner}}(b,t) = \widetilde{r}_{\text{learner}}(b,t) + \sum_{t'\in\mathcal{T}_{\blacklozenge}(t)} \boldsymbol{x}(t')^\top \boldsymbol{B}\delta_b - \frac{k^\star}{T}|\mathcal{T}_{\blacklozenge}(t)|\cdot\mathbb{I}(b=e^\dagger). \tag{83}$$

For every action $b \in \mathcal{B}_{\text{init}} \setminus \{e^\dagger\}$, the second term is 0 by the structure of the game (eq. (24)), as is the third term. For the action $b = e^\dagger$, the second and third terms cancel each out, by the structure of the game (also by eq. (24)). Therefore,

$$\forall b \in \mathcal{B}_{\text{init}},\ r_{\text{learner}}(b,t) = \widetilde{r}_{\text{learner}}(b,t) \tag{84}$$

On the other hand, for actions in $\widetilde{\mathcal{B}}_{\text{init}}$, we have,

$$\forall \widetilde{b} \in \widetilde{\mathcal{B}}_{\text{init}},\ r_{\text{learner}}(\widetilde{b},t) = \widetilde{r}_{\text{learner}}(\widetilde{b},t) + \sum_{t'\in\mathcal{T}_{\blacklozenge}(t)} \boldsymbol{x}(t')^\top \boldsymbol{B}\delta_{\widetilde{b}} - \frac{k^\star}{T}|\mathcal{T}_{\blacklozenge}(t)|\cdot\mathbb{I}(\widetilde{b}=\widetilde{e}^\dagger) \tag{85}$$

$$\overset{(i)}{=} \widetilde{r}_{\text{learner}}(\widetilde{b},t) + \sum_{t'\in\mathcal{T}_{\blacklozenge}(t)} \boldsymbol{x}(t')^\top \boldsymbol{B}\delta_b - \gamma|\mathcal{T}_{\blacklozenge}(t)| - \frac{k^\star}{T}|\mathcal{T}_{\blacklozenge}(t)|\cdot\mathbb{I}(b=e^\dagger) \tag{86}$$

$$\overset{(ii)}{=} \widetilde{r}_{\text{learner}}(\widetilde{b},t) - \gamma|\mathcal{T}_{\blacklozenge}(t)| \tag{87}$$

where in (i) we use b to denote the counterpart of $\widetilde{b}$ in $\mathcal{B}_{\text{init}}$, and use the structure of the game (eq. (25)). In (ii) we use the same argument in going from eq. (83) to eq. (84). Therefore for any action $b \in \mathcal{B}_{\text{init}}$,

$$\mathbf{Pr}_{Y\sim \boldsymbol{y}(t)}(Y=b) = \frac{\exp\left(\eta r_{\text{learner}}(b,t)\right)}{\sum_{b'\in\mathcal{B}_{\text{init}}}\exp\left(\eta r_{\text{learner}}(b',t)\right)+\sum_{b'\in\widetilde{\mathcal{B}}_{\text{init}}}\exp\left(\eta r_{\text{learner}}(b',t)\right)} \tag{88}$$

$$= \frac{\exp\left(\eta \widetilde{r}_{\text{learner}}(b,t)\right)}{\sum_{b'\in\mathcal{B}_{\text{init}}}\exp\left(\eta \widetilde{r}_{\text{learner}}(b',t)\right)+p\sum_{b'\in\mathcal{B}_{\text{init}}}\exp\left(\eta \widetilde{r}_{\text{learner}}(b',t)\right)\cdot\exp\left(-\eta\gamma|\mathcal{T}_{\blacklozenge}(t)|\right)} \tag{89}$$

$$= \frac{1}{1+p\cdot\exp\left(-\eta\gamma|\mathcal{T}_{\blacklozenge}(t)|\right)}\cdot\frac{\exp\left(\eta \widetilde{r}_{\text{learner}}(b,t)\right)}{\sum_{b'\in\mathcal{B}_{\text{init}}}\exp\left(\eta \widetilde{r}_{\text{learner}}(b',t)\right)} \tag{90}$$

Using the definition of reduced MWU completes the proof. ■

Definition 26 (Critical time) *Define $T^{\star}_{\blacklozenge} = \frac{1}{\eta\gamma}\log(p)$ as the critical time. By choice of γ and p, $T^{\star}_{\blacklozenge} = (1-\varepsilon)^2 T$. Define the smallest time t where $|\mathcal{T}_{\blacklozenge}(t)| \geqslant (1-\varepsilon)^3 T$ as the lower critical time.*

If at any time t, $|\mathcal{T}_{\blacklozenge}(t)| < (1-\varepsilon)^3 T$, then $p\cdot\exp\left(-\eta\gamma|\mathcal{T}_{\blacklozenge}(t)|\right) \geqslant p^{\varepsilon} \gg 1$. Likewise, if at any time t, $|\mathcal{T}_{\blacklozenge}(t)| \geqslant (1-\varepsilon)T$, then $p\cdot\exp\left(-\eta\gamma|\mathcal{T}_{\blacklozenge}(t)|\right) \leqslant p^{-\varepsilon} \ll 1$.

Lemma 27 *Consider some sequence of actions $\{\boldsymbol{x}(t)\}_{t\geqslant 1}$ played by the optimizer. Let $\mathcal{T}^{1-\varepsilon}_{init}(t)$ denote the collection of timepoints, $\left\{t' \in \mathcal{T}_{\times\blacklozenge}(t) : |\mathcal{T}_{\blacklozenge}(t')| \geqslant (1-\varepsilon)^3 T\right\}$: the set of time points where the optimizer did not play $\blacklozenge$, accrued after $\blacklozenge$ itself has been played at least $(1-\varepsilon)^3 T$ times. Then, the cumulative reward of the optimizer is at most,*

$$\frac{2T}{p^{\varepsilon}} + \sum_{t\in\mathcal{T}^{1-\varepsilon}_{init}(t)} \boldsymbol{x}(t)^{\top}\boldsymbol{A}\boldsymbol{y}_{red}(t). \tag{91}$$

Proof The cumulative reward of the optimizer can be written as,

$$\sum_{t=1}^{T}\boldsymbol{x}(t)^{\top}\boldsymbol{A}\boldsymbol{y}(t) = \sum_{t\in\mathcal{T}_{\times\blacklozenge}(T)}\frac{1}{1+p\cdot\exp\left(-\eta\gamma|\mathcal{T}_{\blacklozenge}(t)|\right)}\boldsymbol{x}(t)^{\top}\boldsymbol{A}\boldsymbol{y}_{\text{red}}(t) \tag{92}$$

where we use the fact that the optimizer collects no reward for picking $\blacklozenge$ or if the learner picks an action in $\widetilde{\mathcal{B}}_{\text{init}}$, and on the remaining actions we use Lemma 25 to relate $\boldsymbol{y}(t)$ and the reduced MWU, $\boldsymbol{y}_{\text{red}}(t)$. At any time t prior to the lower critical time, we have that $p\cdot\exp\left(-\eta\gamma|\mathcal{T}_{\blacklozenge}(t)|\right) \geqslant p^{\varepsilon} \gg 1$. And therefore, by splitting the summation over $t \in \mathcal{T}_{\times\blacklozenge}(T)$ into $\{t : |\mathcal{T}_{\blacklozenge}(t)| < (1-\varepsilon)^3 T\}$ and the complement (which is nothing but $\mathcal{T}^{1-\varepsilon}_{\text{init}}(T)$), and noting that $\|\operatorname{Vec}(\boldsymbol{A})\|_{\infty} \leqslant 2$, we get the upper bound. ■

C.3. Proof of Lemma 15

Proof This is the cumulative reward collected by the optimizer when playing against MWU over a reduced game which ignores the steps where the optimizer plays $\blacklozenge$, accrued prior to the lower critical time. Prior to the lower critical time, the optimizer may have played actions which were

not ♦. Thus, at the lower critical time, the reduced MWU learner essentially begins with an "initialization" (in the same sense as eqs. (75) to (78)) corresponding to whichever non-♦ optimizer edges were played. Formally, at time any time t after the lower critical time, $|\mathcal{T}_{\blacklozenge}(t)| \geqslant (1-\varepsilon)T^{\star}_{\blacklozenge}$, and any $b \in \mathcal{B}_{\text{init}}$,

$$\widetilde{r}_{\text{learner}}(b,t) = \frac{k^{\star}}{T}|\mathcal{T}_{\blacklozenge}(t)| \cdot \mathbb{I}(b = e^{\dagger}) + \sum_{t' \in \mathcal{T}_{\times\blacklozenge}(t)} \boldsymbol{x}(t')^{\top}\boldsymbol{B}\delta_b \tag{93}$$

$$= \frac{k^{\star}}{T}|\mathcal{T}_{\blacklozenge}(t)| \cdot \mathbb{I}(b = e^{\dagger}) + \sum_{t' \in \mathcal{T}_{\times\blacklozenge}(t) \setminus \mathcal{T}^{1-\varepsilon}_{\text{init}}(t)} \boldsymbol{x}(t')^{\top}\boldsymbol{B}\delta_b + \sum_{t' \in \mathcal{T}^{1-\varepsilon}_{\text{init}}(t)} \boldsymbol{x}(t')^{\top}\boldsymbol{B}\delta_b \tag{94}$$

$$= r_0(b) + \sum_{t' \in \mathcal{T}^{1-\varepsilon}_{\text{init}}(t)} \boldsymbol{x}(t')^{\top}\boldsymbol{B}\delta_b. \tag{95}$$

Where, $r_0(b)$ is an initialization reward precisely of the form eqs. (75) to (78), where the induced value of k is $\frac{k^{\star}}{T}|\mathcal{T}_{\blacklozenge}(t)|$: this is not difficult to see, since $\sum_{t' \in \mathcal{T}_{\times\blacklozenge}(t) \setminus \mathcal{T}^{1-\varepsilon}_{\text{init}}(t)} \boldsymbol{x}(t')^{\top}\boldsymbol{B}\delta_b$ is an initial reward obtained by playing some set of optimizer edges E_{init} (corresponding to those played at time $\mathcal{T}_{\times\blacklozenge}(t) \setminus \mathcal{T}^{1-\varepsilon}_{\text{init}}(t)$: the non-♦ actions played prior to the lower critical time). While the induced value of k depends on the sequence of actions played thus far, the fact that $(1-\varepsilon)^3 T \leqslant |\mathcal{T}_{\blacklozenge}(t)| \leqslant T$ constrains it very tightly. The former induces $k = (1-\varepsilon)^3 T \times k^{\star}/T = (1-\varepsilon)^2 k^{\star}$ and the latter induces $k = T \times k^{\star}/T = k^{\star}$.

With this discussion, we may bound the value collected by the optimizer using Corollary 23. Noting the fact that the optimizer has played ♦ at least $(1-\varepsilon)^3 T$ times, the remaining actions must have been played at most $T - (1-\varepsilon)^3 T$ times. Note that the reduced MWU is only determined at the time points the optimizer does not play ♦, which is a very small fraction of the overall horizon, covering at most $\leqslant T - (1-\varepsilon)^3 T \leqslant 3\varepsilon T$ steps. In particular, for the induced initialization at the lower critical time, the set of edges in the initialization, E_{init}, is $\mathcal{T}_{\times\blacklozenge}(T) \setminus \mathcal{T}^{1-\varepsilon}_{\text{init}}(T)$, and the remaining duration of the horizon, T_{init} is at most $|\mathcal{T}^{1-\varepsilon}_{\text{init}}(T)|$. The overall effective horizon (cf. Corollary 23), $|E_{\text{init}}| + T_{\text{init}}$ is upper bounded by $T - (1-\varepsilon)^3 T \leqslant 3\varepsilon T$ assuming that $\mathcal{T}^{1-\varepsilon}_{\text{init}}(T)$ is non-empty. In particular, as a consequence of Corollary 23, with the choice $k = k^{\star}$, results in the upper bound: for any $\Delta \geqslant \frac{1}{\eta}\log(2n^2 T) \geqslant \frac{1}{\eta}\log\big(2n^2(1-\varepsilon)T\big) = \frac{1}{\eta}\log\big(2n^2(T_{\text{init}} + |E_{\text{init}}|)\big)$,

$$\sum_{t \in \mathcal{T}^{1-\varepsilon}_{\text{init}}(T)} \boldsymbol{x}(t)^{\top}\boldsymbol{A}\boldsymbol{y}_{\text{red}}(t) \leqslant (k^{\star}(1+5\varepsilon) + 9\Delta)\sum_{i=1}^{n} W(z_i, z_{i+1}) + (2k^{\star} + 2\Delta + 4) \tag{96}$$

The proof concludes by combining with Lemma 27 and the choice $p = T^{\beta/3}$. ■

C.4. Proof of Lemma 16

Consider the optimizer strategy which plays the action ♦, precisely $(1-\varepsilon)T$ times. On the remaining iterations, which is of length εT, the optimizer follows the strategy outlined in Lemma 14, on a horizon of length εT. Firstly, observe that, by Lemma 25, at any time $t > (1-\varepsilon)T$,

$$\mathbf{Pr}_{Y \sim \boldsymbol{y}(t)}(Y = b) = \frac{1}{1 + p \cdot \exp\left(-\eta\gamma|\mathcal{T}_{\blacklozenge}(t)|\right)} \cdot \mathbf{Pr}_{Y \sim \boldsymbol{y}_{\text{red}}(t)}(Y = b) \tag{97}$$

$$= \frac{1}{1+p^{-\varepsilon}} \cdot \mathbf{Pr}_{Y \sim \boldsymbol{y}_{\text{red}}(t)}(Y = b) \tag{98}$$

where in the last equation, we use the choice of γ and p. At time $t = (1-\varepsilon)T + 1$, the cumulative reward on actions is of the form,

$$\text{For } v_{\text{in}} \in V_{\text{in}},\ r_{\text{learner}}(v_{\text{in}}, t) = -(1-\varepsilon)k^\star = \varepsilon T^{1-\min\{\alpha,\beta\}/3} \tag{99}$$

$$\text{For } v_{\text{out}} \in V_{\text{out}},\ r_{\text{learner}}(v_{\text{out}}, t) = -\varepsilon T^{1-\min\{\alpha,\beta\}/3} \tag{100}$$

$$\text{For } (u,v) \in E \setminus \{e^\dagger\},\ r_{\text{learner}}((u,v), t) = 0 \tag{101}$$

$$r_{\text{learner}}(e^\dagger, t) = \varepsilon T^{1-\min\{\alpha,\beta\}/3} \tag{102}$$

Note that this optimizer has $T_{\text{init}} = \varepsilon T$ time remaining in the horizon, and the induced value of $k = \varepsilon T^{1-\min\{\alpha,\beta\}/3}$: in particular, since, $n \times k = \varepsilon T \triangleq T_{\text{init}}$, and therefore the horizon is sufficiently long to obtain the reward in of the optimizer policy in Lemma 14.

The cumulative reward of the optimizer can be written as,

$$\sum_{t=1}^{T} \boldsymbol{x}(t)^\top \boldsymbol{A}\boldsymbol{y}(t) = \frac{1}{1+p^{-\varepsilon}} \sum_{t=(1-\varepsilon)T+1}^{T} \boldsymbol{x}(t)^\top \boldsymbol{A}\boldsymbol{y}_{\text{red}}(t) \tag{103}$$

$$\geqslant (1-\varepsilon)^2 \varepsilon T^{1-\min\{\alpha,\beta\}/3} \times \frac{1}{1+p^{-\varepsilon}} \left(1 - \frac{2}{n}\right) \sum_{i=1}^{n} W(z_i^\star, z_{i+1}^\star) \tag{104}$$

$$\geqslant (1-\varepsilon)^3 k^\star \times \frac{1}{1+p^{-\varepsilon}} \left(1 - \frac{2}{n}\right) \sum_{i=1}^{n} W(z_i^\star, z_{i+1}^\star) \tag{105}$$

which completes the proof.

C.5. Proof of Theorem 1

The proof of this result follows very similarly to that of Theorem 19. The ratio of the optimizer's reward R_{opt}, to that of the best optimizer is at least,

$$\frac{R_{\text{opt}}}{R_\star} \geqslant \frac{R_{\text{opt}}}{R_{\text{lb}}} \geqslant \frac{2T^{1-\beta\varepsilon/3} + (k^\star(1+5\varepsilon) + 9\Delta)\sum_{i=1}^{n} W(z_i, z_{i+1}) + (2k + 2\Delta + 4)}{(1-\varepsilon)^3 k^\star \times \frac{1}{1+p^{-\varepsilon}} \left(1 - \frac{2}{n}\right) \sum_{i=1}^{n} W(z_i^\star, z_{i+1}^\star)} \tag{106}$$

$$\geqslant \left(1 - C\varepsilon - o_T(1)\right) \cdot \frac{\sum_{i=1}^{n} W(z_i, z_{i+1})}{\sum_{i=1}^{n} W(z_i^\star, z_{i+1}^\star)} \tag{107}$$

for a sufficiently small large absolute constant $C > 0$ in the last inequality. Here we use the fact that $\Delta = O(T^{1-\alpha}\log(T))$, while $k^\star = \Omega(T^{1-\alpha/3})$, and therefore $k^\star/\Delta = T^{\Omega(\alpha)}$ and the fact that $T^{1-\beta\varepsilon/3} \ll nk^\star = \varepsilon T$. By Theorem 5, this implies that unless $\mathsf{P} = \mathsf{NP}$, a polynomial time learner must satisfy,

$$\frac{R_{\text{opt}}}{R_\star} \leqslant \frac{1067}{1068} + \epsilon \tag{108}$$

for any $\epsilon > 0$. Since $R_\star \geqslant R_{\text{lb}}$ is at least $(1 - c_0\varepsilon)\varepsilon T - o(T)$ for some $c_0 > 0$, and noting that ε is a small constant, this implies that unless $\mathsf{P} = \mathsf{NP}$, a polynomial time optimizer must satisfy,

$$R_\star - R_{\text{opt}} \geqslant c_1 \varepsilon T - o(T), \tag{109}$$

for some constant $c_1 > 0$, thereby proving Theorem 1.