ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

PROJECT: ALGOLINC – Algorithmic Learning Theory and Incentives: Synergies in Optimization and Mechanism Design

Project ID: 15877

Greece 2.0 NATIONAL RECOVERY AND RESILIENCE PLAN "BASIC RESEARCH FINANCING" (Horizontal support for all Sciences)

WP4: Games, Optimization and Online Learning

Deliverable D.4.1: Initial report on algorithms for optimization under perfect gradient information

Prepared by the team members:

Michail Fasoulakis, Evangelos Markakis, Panagiotis Tsamopoulos

ΕΛ.Ι.Δ.Ε.Κ.

Ελληνικό Ίδρυμα Έρευνας & Καινοτομίας

Deliverable D.4.1: Initial report on algorithms for optimization under perfect gradient information

Overview

This deliverable concerns WP4, and was completed on time by Month M12 according to the timeline of the project. The goal of this report is to provide an overview of our initial progress regarding the algorithmic research questions that form the focus of WP4.

We briefly recall the research topic that we are interested in. Task 4.1 of WP4 is centered around the problem of finding Nash equilibria in bilinear zero-sum games. Zero-sum games have played a fundamental role in both game theory, being among the first classes of games formally studied, and in optimization. In particular, it is easily seen that the equilibrium solutions of a zero-sum game correspond to solving a min-max optimization problem of the form:

$$\min_{\boldsymbol{x}} \max_{\boldsymbol{y}} f(\boldsymbol{x}, \boldsymbol{y})$$

where $\boldsymbol{x}$ and $\boldsymbol{y}$ are vectors of variables, constrained to be probability distributions (i.e., mixed strategies for the two players) and $f(\boldsymbol{x}, \boldsymbol{y})$ is the payoff function of the first player.

Even further, solving zero-sum games is in fact equivalent to solving linear programs. Despite the fact that a single linear program suffices to find a Nash equilibrium, there has been a surge of interest in recent years for faster algorithms, motivated in part by applications in machine learning. One reason for this is that we may have very large games to solve, corresponding to LPs with thousands of variables and constraints. A second reason could be that, e.g., in learning environments, the players may be using iterative algorithms that can only observe limited information, hence it would be impossible to run a single linear program for the entire game.

Given the above considerations, we have made progress on WP4 along two directions. The first one is an optimization approach, whereas the second one concerns the analysis of a learning algorithm.

An optimization method for finding approximate Nash equilibria.

We have developed a method based on a descent approach for the duality gap of the strategy profiles. The duality gap is a standard metric when measuring convergence and what is interesting with our approach is that especially for zero-sum games, the duality gap function is convex. This provides the intuition that applying descent steps with respect to this function might be more promising than performing the more standard gradient descent variants on the players' utility functions (which are non-convex). Unfortunately it is not so straightforward to compute the gradient under our methodology and for this we have to resort to approximating the directional derivative. Eventually this is done by resorting to linear programming, albeit the linear programs that we end up solving are rather small in size, and hence easily solvable. We have established mathematically that this method does achieve a geometric rate of decrease on the duality gap and thus we reach quite fast an approximate equilibrium. We have also explored certain variants that could speed up further the overall approach. At the same time, we have provided initial experiments that verify the fast convergence of our algorithm and its competitive performance against state of the art methods for solving zero-sum games. We have recently completed the writing of a research article on this, with a detailed exposition of our algorithms and its analysis, which we append at the end of this deliverable.

Analysis of a learning method for converging to approximate Nash equilibria.

In parallel to the above, we have explored a type of learning dynamics that form a variation of the extra gradient and mirror prox methods. In fact these dynamics were proposed in an earlier work from our research team (Fasoulakis et al., AISTATS 2022). However, our previous work established only asymptotic convergence without any concrete convergence rates.

We refer to this method by the name of Forward Looking Best-Response Multiplicative Weights Update method (FLBR-MWU). We provide here a short description of the main idea behind the dynamics. This is an adaptation of the extra gradient method but applied to Multiplicative Weights Updates, and each iteration has an intermediate and a final step. Suppose that starting from some initial profile, we reach the profile $(\boldsymbol{x}^{t-1}, \boldsymbol{y}^{t-1})$ by the end of iteration $t-1$. In the intermediate step of iteration t, we compute a strategy $\hat{\boldsymbol{x}}^t$ for the row player (resp. $\hat{\boldsymbol{y}}^t$ for the column player), which is an approximate best-response strategy to $\boldsymbol{y}^{t-1}$ (resp. to $\boldsymbol{x}^{t-1}$). This serves as a look ahead step of what would be the currently optimal choices. In the final step of iteration t, we compute the new mixed strategy $\boldsymbol{x}^t$ for the row player, by performing MWU updates, but after assuming that the opponent was playing $\hat{\boldsymbol{y}}^t$.

Formally, the first step of the dynamics is defined below, at iteration t, and for all $i, j \in [n]$, given a non-negative parameter $\xi \in \mathbb{R}^+$ (ξ is chosen sufficiently large).

$$\begin{aligned}
\hat{\boldsymbol{x}}_i^t &= \boldsymbol{x}_i^{t-1} \cdot \frac{e^{\xi \boldsymbol{e}_i^\top R \boldsymbol{y}^{t-1}}}{\sum_{j=1}^n \boldsymbol{x}_j^{t-1} e^{\xi \boldsymbol{e}_j^\top R \boldsymbol{y}^{t-1}}}, \\
\hat{\boldsymbol{y}}_j^t &= \boldsymbol{y}_j^{t-1} \cdot \frac{e^{-\xi \boldsymbol{e}_j^\top R^\top \boldsymbol{x}^{t-1}}}{\sum_{i=1}^n \boldsymbol{y}_i^{t-1} e^{-\xi \boldsymbol{e}_i^\top R^\top \boldsymbol{x}^{t-1}}}.
\end{aligned} \tag{1}$$

The second step, which updates the profile $(\boldsymbol{x}^{t-1}, \boldsymbol{y}^{t-1})$ to $(\boldsymbol{x}^t, \boldsymbol{y}^t)$ is below, given the learning rate parameter $\eta \in (0, 1)$. We assume that we use the same fixed constants η and ξ in all iterations.

$$\begin{aligned}
\boldsymbol{x}_i^t &= \boldsymbol{x}_i^{t-1} \cdot \frac{e^{\eta \boldsymbol{e}_i^\top R \hat{\boldsymbol{y}}^t}}{\sum_{j=1}^n \boldsymbol{x}_j^{t-1} e^{\eta \boldsymbol{e}_j^\top R \hat{\boldsymbol{y}}^t}}, \\
\boldsymbol{y}_j^t &= \boldsymbol{y}_j^{t-1} \cdot \frac{e^{-\eta \boldsymbol{e}_j^\top R^\top \hat{\boldsymbol{x}}^t}}{\sum_{i=1}^n \boldsymbol{y}_i^{t-1} e^{-\eta \boldsymbol{e}_i^\top R^\top \hat{\boldsymbol{x}}^t}}.
\end{aligned} \tag{2}$$

Our current goal is to establish that this method achieves a geometric rate of convergence for reaching an approximate Nash equilibrium. We started this effort during the last months of this reporting period, and therefore we have not yet completed our analysis. So far however, the method looks very promising as it outperforms experimentally other standard learning algorithms for zero-sum games. We therefore expect that we will be able to finalize our theoretical analysis in the coming months.

A Appendix

In the remainder of this deliverable we include our article regarding the new optimization method explained earlier. Our article includes all the relevant definitions along with our theoretical and experimental evaluation. We plan to submit this work to an international conference within the coming 2-3 months.

A Descent-based method on the Duality Gap for solving zero-sum games

Michail Fasoulakis[1], Evangelos Markakis[1,2], Giorgos Roussakis[3], and Christodoulos Santorinaios[1,2]

[1]Athens University of Economics and Business, Greece
[2]Archimedes/Athena RC, Greece
[3]Foundation for Research and Technology–Hellas

Abstract

We focus on the design of algorithms for finding equilibria in 2-player zero-sum games. Although it is well known that such problems can be solved by a single linear program, there has been a surge of interest in recent years for simpler algorithms, motivated in part by applications in machine learning. Our work proposes such a method, inspired by the observation that the duality gap (a standard metric for evaluating convergence in min-max optimization problems) is a convex function for bilinear zero-sum games. To this end, we analyze a descent-based approach, variants of which have also been used as a subroutine in a series of algorithms for approximating Nash equilibria in general non-zero-sum games. In particular, we study a steepest descent approach, by finding the direction that minimises the directional derivative of the duality gap function. Our main theoretical result is that the derived algorithms achieve a geometric decrease in the duality gap and improved complexity bounds until we reach an approximate equilibrium. Finally, we complement this with an experimental evaluation, which provides promising findings. Our algorithm is comparable with (and in some cases outperforms) some of the standard approaches for solving 0-sum games, such as OGDA (Optimistic Gradient Descent/Ascent), even with thousands of available strategies per player.

1 Introduction

Our work focuses on the design of algorithms for finding Nash equilibria in 2-player bilinear zero-sum games. Zero-sum games have played a fundamental role both in game theory, being among the first classes of games formally studied, and in optimization, as it is easily seen that their equilibrium solutions correspond to solving a min-max optimization problem. Even further, solving zero-sum games is in fact equivalent to solving linear programs, as properly demonstrated in Adler [2013].

Despite the fact that a single linear program (and its dual) suffices to find a Nash equilibrium, there has been a surge of interest in recent years, for faster algorithms, motivated in part by applications in machine learning. One reason for this is that we may have very large games to solve, corresponding to LPs with thousands of variables and constraints. A second reason could be that e.g., in learning environments, the players may be using iterative algorithms that can only observe limited information, hence it would be impossible to run a single LP for the entire game. As an additional motivation, finding new algorithms for such a fundamental problem can provide insights that could be of further value and interest.

The above considerations have led to a variety of approaches and algorithms, spanning already a few decades of research. Some of the earlier works on this domain have focused purely on an optimization viewpoint. In parallel to this, significant attention has been drawn to learning-oriented algorithms, such as first-order methods. The latter class of algorithms performs gradient descent or ascent on the utility functions of the two players, and some of the proposed variants have been very successful in practice, such as the optimistic gradient and the extra gradient methods Korpelevich [1976], Popov [1980]. Several works have focused on theoretical guarantees for their performance, and a standard metric used in the analysis is the duality gap. This is simply the sum of the regrets of the two players in a given profile, and therefore the goal often amounts to proving appropriate rates of decrease for the duality gap over the iterations of an algorithm.

Our work is motivated by the observation that the duality gap is a convex function for zero-sum games. This naturally gives rise to the suggestion that instead of performing gradient descent on the utility function of a player, which is not a convex function, we could apply a descent procedure directly on the duality gap. It is not straightforward that this can indeed be useful as it is not a priori clear that we can perform a descent step fast (i.e., finding the direction to move to). Nevertheless, it can form the basis for investigating new approaches for zero-sum games.

1.1 Our Contributions

Motivated by the above discussion, we propose and analyze an optimization approach for finding approximate Nash equilibria in zero-sum games. Our algorithm is a descent-based method applied to the duality gap function, and is essentially an adaptation of a subroutine in the algorithms of Tsaknakis and Spirakis [2008], Deligkas et al. [2017, 2023] which are for general games, tailored to zero-sum games and with a different objective function. The method is applying a steepest descent approach, where we find in each step the direction that minimises the directional derivative of the duality gap function and move towards that. In Section 3 we provide the algorithm and our theoretical analysis. Our main result is that the derived algorithm achieves a geometric decrease in the duality gap until we reach an approximate equilibrium. This implies that the algorithm terminates after at most $O\left(\frac{1}{\rho} \cdot \log\left(\frac{1}{\delta}\right)\right)$ iterations with a δ-approximate equilibrium, where ρ is a parameter, related to the computation of the directional derivative. We exhibit that the method can also be further customized and show that a different variant also converges after $O(\frac{1}{\sqrt{\delta}})$ iterations.

In Section 4, we complement our theoretical analysis with an experimental evaluation. Even though the method does need to solve a linear program in each iteration to find the desirable direction, this turns out to be of much smaller size on average (in terms of the number of constraints) than solving the linear program of the entire game. We compare our method against standard LP solvers, but also against state-of-the-art procedures for zero-sum games, such as Optimistic Gradient Descent-Ascent (OGDA). Our findings are promising and reveal that the running time is comparable to (and often outperforms) OGDA, even with thousands of strategies per player. We therefore conclude that the overall approach deserves further exploration, as there are also potential ways of accelerating its running time, discussed in Section 4.

1.2 Related Work

As already mentioned, conceptually, the works most related to ours are Tsaknakis and Spirakis [2008], Deligkas et al. [2017, 2023]. Although these papers do not consider zero-sum games, they do utilize a descent-based part as a starting point. The main differences with our work is that first of all, their descent is performed with respect to the maximum regret among the two players, whereas we use the duality gap function. Furthermore the descent phase is only a subroutine of their algorithms, since it does not suffice to establish guarantees for general games. Hence their focus is less on the decent phase itself and more on utilizing further procedures to produce approximate equilibria.

There is a plethora of algorithms for linear programming and zero-sum games, which is impossible to list here, but we comment on what we feel are most relevant. When focusing on optimization algorithms for zero-sum games, Hoda et al. [2010] use Nesterov's first order smoothing techniques to achieve an ϵ-equilibrium in $O(1/\epsilon)$ iterations, with added benefits of simplicity and rather low computational cost per iteration. Following up on that work, Gilpin et al. [2012] propose an iterated version of Nesterov's smoothing technique, which runs within $O(\frac{||A||}{\delta(A)} \cdot \ln(1/\epsilon))$ iterations. However, while this is a significant improvement, the complexity depends on a condition measure $\delta(A)$, with A being the payoff matrix, not necessarily bounded by a constant. Another optimization approach that is relevant in spirit to ours is via the Nikaido-Isoda function Nikaido and Isoda [1955] and its variants. E.g., in Raghunathan et al. [2019] they run a descent method on the Gradient NI function, which is convex for zero-sum games. We are not aware though of any direct connection to the duality gap function that we use here.

Apart from the optimization viewpoint, there has been great interest in designing faster learning algorithms for zero-sum games. Although this direction started already several decades ago, e.g. with the fictitious play algorithm Brown [1951], Robinson [1951], it has received significant attention more recently given the relevance to formulating GANs in deep learning Goodfellow et al. [2014] and also other applications in machine learning. Some of the earlier and standard results in this area concern convergence *on average*. That is, it has been known that by using no-regret algorithms, such as the Multiplicative Weights Update (MWU) methods Arora et al. [2012] the empirical average of the players' strategies over time converges to a Nash equilibrium in zero-sum games. Similarly, one could also utilize the so-called Gradient Descent/Ascent (GDA) algorithms.

Within the last decade, there has also been a great interest in algorithms attaining the more robust notion of *last-iterate convergence*. This means that the strategy profile (x_t, y_t), reached at iteration t, converges to the actual equilibrium as $t \to \infty$. Negative results in Bailey and Piliouras [2018] and Mertikopoulos et al. [2018] exhibit that several no-regret algorithms such as many MWU as well as GDA variants, do not satisfy last-iterate convergence. Motivated by this, there has been a series of works on obtaining algorithms with provable last iterate convergence. The positive results that have been obtained for zero-sum games is that improved versions of Gradient Descent such as the Extra Gradient method Korpelevich [1976] or the Optimistic Gradient method Popov [1980] attain last iterate convergence. In particular, Daskalakis et al. [2018] and Liang and Stokes [2019] show that the optimistic variant of GDA (referred to as OGDA) converges for zero-sum games. Analogously, OMWU (the optimistic version of MWU) also attains last iterate convergence, shown in Daskalakis and Panageas [2019] and further analyzed in Wei et al. [2021]. The rate of convergence of optimistic gradient methods in terms of the duality gap was studied in Cai et al. [2022], and was later improved to $O(1/t)$ in Cai and Zheng [2023]. Further approaches with convergence guarantees have also been proposed, based on variations of the Mirror-Prox method Fasoulakis et al. [2022] as well as primal-dual hybrid gradient methods Lu and Yang [2023]. Finally, several of these methods have also been studied beyond zero-sum games, including among others Golowich et al. [2020], where Optimistic Gradient is analyzed for more general games and Diakonikolas et al. [2021] where positive results are shown for a class of non-convex and non-concave problems. The picture however is more complex for general games with negative results also established in Daskalakis et al. [2021].

2 Preliminaries

We consider bilinear zero-sum games $(\boldsymbol{R}, -\boldsymbol{R})$, with n pure strategies per player, where $\boldsymbol{R}$ is the payoff matrix of the row player. We assume $\boldsymbol{R} \in [0,1]^{n\times n}$ without loss of generality[1]. We consider mixed strategies $\boldsymbol{x} \in \Delta^{n-1}$ as a probability distribution (column vector) on the pure strategies of a player, with Δ^{n-1} be the $(n-1)$-dimensional simplex. We also denote by $\boldsymbol{e}_i$ the distribution corresponding to a pure strategy i, with 1 in the index i and zero elsewhere. A strategy profile is a pair $(\boldsymbol{x}, \boldsymbol{y})$, where $\boldsymbol{x}$ is the strategy of the row player and $\boldsymbol{y}$ is the strategy of the column player. Under a profile $(\boldsymbol{x}, \boldsymbol{y})$, the expected payoff of the row player is $\boldsymbol{x}^\top \boldsymbol{R}\boldsymbol{y}$ and the expected payoff of the column player is $-\boldsymbol{x}^\top \boldsymbol{R}\boldsymbol{y}$.

A pure strategy i is a ρ-best-response strategy against $\boldsymbol{y}$ for the row player, for $\rho \in [0,1]$, if and only if, $\boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} + \rho \geq \boldsymbol{e}_j^\top \boldsymbol{R}\boldsymbol{y}$, for any j. Similarly, a pure strategy j for the column player is a ρ-best-response strategy against some strategy $\boldsymbol{x}$ of the row player if and only if $\boldsymbol{x}^T \boldsymbol{R}\boldsymbol{e}_j \leq \boldsymbol{x}^T \boldsymbol{R}\boldsymbol{e}_i + \rho$, for any i. Having these, we define as $BR_r^\rho(\boldsymbol{y})$ the set of the ρ-best-response pure strategies of the row player against $\boldsymbol{y}$ and as $BR_c^\rho(\boldsymbol{x})$ the set of the ρ-best-response pure strategies of the column player against $\boldsymbol{x}$. For $\rho = 0$, we will use $BR_r(\boldsymbol{y})$ and $BR_c(\boldsymbol{x})$ for the best response sets.

Definition 1 (Nash equilibrium Nash [1951], Von Neumann [1928])**.** *A strategy profile $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is a Nash equilibrium in the game $(\boldsymbol{R}, -\boldsymbol{R})$, if and only if, for any i, j,*

$$v = \boldsymbol{x}^{*\top} \boldsymbol{R}\boldsymbol{y}^* \geq \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}^*, \text{ and, } v = \boldsymbol{x}^{*\top} \boldsymbol{R}\boldsymbol{y}^* \leq \boldsymbol{x}^{*\top} \boldsymbol{R}\boldsymbol{e}_j,$$

where v is the value of the row player (value of the game).

Definition 2 (δ-Nash equilibrium)**.** *A strategy profile $(\boldsymbol{x}, \boldsymbol{y})$ is a δ-Nash equilibrium (in short, δ-NE) in the game $(\boldsymbol{R}, -\boldsymbol{R})$, with $\delta \in [0,1]$, if and only if, for any i, j,*

$$\boldsymbol{x}^\top \boldsymbol{R}\boldsymbol{y} + \delta \geq \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}, \text{ and, } \boldsymbol{x}^\top \boldsymbol{R}\boldsymbol{y} - \delta \leq \boldsymbol{x}^\top \boldsymbol{R}\boldsymbol{e}_j.$$

[1] We can easily see that we can do scaling for any $\boldsymbol{R} \in \mathbb{R}^{n\times n}$ s.t. $\boldsymbol{R} \in [0,1]^{n\times n}$ keeping exactly the same Nash equilibria.

With these at hand, we can now define the regret functions of the players as follows.

Definition 3 (Regret of a player)**.** *For a game $(\boldsymbol{R}, -\boldsymbol{R})$, the regret function $f_{\boldsymbol{R}} : \Delta^{n-1} \times \Delta^{n-1} \to [0,1]$ of the row player under a strategy profile $(\boldsymbol{x}, \boldsymbol{y})$ is*

$$f_{\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{y}) = \max_i \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} - \boldsymbol{x}^\top \boldsymbol{R}\boldsymbol{y}.$$

Similarly, for the column player the regret function is

$$f_{-\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{y}) = \max_j \boldsymbol{x}^\top(-\boldsymbol{R})\boldsymbol{e}_j + \boldsymbol{x}^\top \boldsymbol{R}\boldsymbol{y} = -\min_j \boldsymbol{x}^\top \boldsymbol{R}\boldsymbol{e}_j + \boldsymbol{x}^\top \boldsymbol{R}\boldsymbol{y}.$$

An important quantity for evaluating the performance or convergence of algorithms is the sum of the regrets, i.e., the function $V(\boldsymbol{x}, \boldsymbol{y}) = f_{\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{y}) + f_{-\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{y}) = \max_i \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} - \min_j \boldsymbol{x}^\top \boldsymbol{R}\boldsymbol{e}_j$. This is referred to in the bibliography as the *duality gap* in the case of zero-sum games.

2.1 Warmup: Duality Gap Properties

Next, we present some known results about the *duality gap* function $V(\boldsymbol{x}, \boldsymbol{y})$ and its connection to Nash equilibria. For the sake of completeness, we provide proofs.

Theorem 4. *The duality gap $V(\boldsymbol{x}, \boldsymbol{y})$ is convex in its domain.*

Proof. Let $(\boldsymbol{x}_1, \boldsymbol{y}_1)$ and $(\boldsymbol{x}_2, \boldsymbol{y}_2)$ be two arbitrary different strategy profiles, $p \in (0,1)$ and $(\boldsymbol{x}, \boldsymbol{y}) = p \cdot (\boldsymbol{x}_1, \boldsymbol{y}_1) + (1-p) \cdot (\boldsymbol{x}_2, \boldsymbol{y}_2) = (p \cdot \boldsymbol{x}_1 + (1-p) \cdot \boldsymbol{x}_2, p \cdot \boldsymbol{y}_1 + (1-p) \cdot \boldsymbol{y}_2)$ be a convex combination of them. Then, we have

$$\begin{aligned}
V(\boldsymbol{x}, \boldsymbol{y}) &= V(p \cdot \boldsymbol{x}_1 + (1-p) \cdot \boldsymbol{x}_2, p \cdot \boldsymbol{y}_1 + (1-p) \cdot \boldsymbol{y}_2) \\
&= \max_i \boldsymbol{e}_i^\top \boldsymbol{R}(p \cdot \boldsymbol{y}_1 + (1-p) \cdot \boldsymbol{y}_2) - \min_j (p \cdot \boldsymbol{x}_1 + (1-p) \cdot \boldsymbol{x}_2)^\top \boldsymbol{R}\boldsymbol{e}_j \\
&\leq p \cdot \max_i \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}_1 + (1-p) \cdot \max_i \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}_2 \\
&\quad - p \cdot \min_j \boldsymbol{x}_1^\top \boldsymbol{R}\boldsymbol{e}_j - (1-p) \cdot \min_j \boldsymbol{x}_2^\top \boldsymbol{R}\boldsymbol{e}_j \\
&= p \cdot \max_i \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}_1 + (1-p) \cdot \max_i \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}_2 - p \cdot \min_j \boldsymbol{x}_1^\top \boldsymbol{R}\boldsymbol{e}_j - (1-p) \cdot \min_j \boldsymbol{x}_2^\top \boldsymbol{R}\boldsymbol{e}_j \\
&= p \cdot V(\boldsymbol{x}_1, \boldsymbol{y}_1) + (1-p) \cdot V(\boldsymbol{x}_2, \boldsymbol{y}_2),
\end{aligned}$$

the first inequality holds by the convexity and concavity of the max and the min function, respectively. □

Theorem 5. *A strategy profile $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is a Nash equilibrium of the game $(\boldsymbol{R}, -\boldsymbol{R})$, if and only if, it is a (global) minimum*[2] *of the function $V(\boldsymbol{x}, \boldsymbol{y})$.*

Proof. Let $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ be a Nash equilibrium, then it holds $V(\boldsymbol{x}^*, \boldsymbol{y}^*) = 0$ by the definition of the NE, but since the values of $V(\boldsymbol{x}, \boldsymbol{y}) \in [0,2]$ this implies that $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is a global minimum of the function in its domain. Let now a strategy profile $(\boldsymbol{x}, \boldsymbol{y})$ such that $V(\boldsymbol{x}, \boldsymbol{y}) = 0 = f_{\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{y}) + f_{-\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{y})$, this trivially implies that $f_{\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{y}) = 0$ and $f_{-\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{y}) = 0$ since $f_{\boldsymbol{R}}, f_{-\boldsymbol{R}} \in [0,1]$, thus we have that $(\boldsymbol{x}, \boldsymbol{y})$ is a NE in the zero-sum game. □

Similarly to the previous theorem, we also have the following.

Theorem 6. *Let $(\boldsymbol{x}, \boldsymbol{y})$ be a strategy profile in a zero-sum game. If $V(\boldsymbol{x}, \boldsymbol{y}) \leq \delta$, then $(\boldsymbol{x}, \boldsymbol{y})$ is a δ-NE.*

[2]Note that the set of Nash equilibria in zero-sum games and the set of optimal solutions, minimizing the duality gap are convex and identical to each other.

3 Descent-based Algorithms on the Duality Gap: Theoretical Analysis

In this section, we present our main algorithm along with some improved variants, based on a gradient-descent approach for the function $V(\boldsymbol{x},\boldsymbol{y})$ in zero-sum games. The algorithm can be seen as an adaptation[3] of a descent procedure that forms the initial phase of algorithms proposed for general non-zero-sum games, in Tsaknakis and Spirakis [2008], Deligkas et al. [2017, 2023]. The main idea behind the algorithm is that since the global minimum of the duality gap function $V(\boldsymbol{x},\boldsymbol{y})$ is a Nash equilibrium and the duality gap is a convex function for zero-sum bilinear games, we use a descent method based on the directional derivative of $V(\boldsymbol{x},\boldsymbol{y})$. This differs substantially from applying the more common idea of gradient descent/ascent (GDA) on the utility functions of the players, which are not convex functions. To identify the direction that minimizes the directional derivative at every step we use linear programming (albeit solving much smaller linear programs on average than the program describing the zero-sum game itself).

To begin with, we define first the directional derivative.

Definition 7. *The directional derivative of the duality gap at a point $\boldsymbol{z}=(\boldsymbol{x},\boldsymbol{y})$, with respect to a direction $\boldsymbol{z'}=(\boldsymbol{x'},\boldsymbol{y'})\in\Delta^{n-1}\times\Delta^{n-1}$ is the limit, if it exists,*

$$\nabla_{\boldsymbol{z'}}V(\boldsymbol{z})=\lim_{\varepsilon\to 0}\frac{V\Big((1-\varepsilon)\cdot\boldsymbol{z}+\varepsilon\cdot\boldsymbol{z'}\Big)-V(\boldsymbol{z})}{\varepsilon}$$

We provide below a much more convenient form for the directional derivative that facilitates the remaining analysis.

Theorem 8. *The directional derivative of the duality gap V at a point $\boldsymbol{z}=(\boldsymbol{x},\boldsymbol{y})$ with respect to a direction $\boldsymbol{z'}=(\boldsymbol{x'},\boldsymbol{y'})\in\Delta^{n-1}\times\Delta^{n-1}$, is given by*

$$\nabla_{\boldsymbol{z'}}V(\boldsymbol{z})=\max_{i\in BR_r(\boldsymbol{y})}\boldsymbol{e}_i^\top\boldsymbol{R}\boldsymbol{y'}-\min_{j\in BR_c(\boldsymbol{x})}(\boldsymbol{x'})^\top\boldsymbol{R}\boldsymbol{e}_j-V(\boldsymbol{z})$$

Furthermore, by the definition of directional derivative we have the following consequence.

Lemma 9. *Given $\delta\in[0,1]$, let $\boldsymbol{z}=(\boldsymbol{x},\boldsymbol{y})$ be a strategy profile that is not a δ-Nash equilibrium. Then*

$$\nabla_{\boldsymbol{z'}}V(\boldsymbol{z})<-\delta,$$

where $\boldsymbol{z'}=(\boldsymbol{x'},\boldsymbol{y'})\in\Delta^{n-1}\times\Delta^{n-1}$ is a direction that minimizes the directional derivative.

The proof of Lemma 9 follows by a more general result presented in Lemma 12 below (using also Lemma 11). In a similar manner to Definition 7, we define now an approximate version of the directional derivative. The reason we do that will become clear later on, in order to show that the duality gap decreases from one iteration of the algorithm to the next. The main idea in the definition below is to include approximate best responses in the maximization and minimization terms involved in the statement of Theorem 8. Namely, for $\rho>0$, recall the definition of $BR_r^\rho(\boldsymbol{y})$ as the set of ρ-best response strategies of the row player against strategy $\boldsymbol{y}$ of the column player (and similarly for $BR_c^\rho(\boldsymbol{x})$).

Definition 10 (ρ-directional derivative)**.** *The ρ-directional derivative of the duality gap V at a point $\boldsymbol{z}=(\boldsymbol{x},\boldsymbol{y})$ with respect to a direction $\boldsymbol{z'}=(\boldsymbol{x'},\boldsymbol{y'})\in\Delta^{n-1}\times\Delta^{n-1}$ is*

$$\nabla_{\rho,\boldsymbol{z'}}V(\boldsymbol{z})=\max_{i\in BR_r^\rho(\boldsymbol{y})}\boldsymbol{e}_i^\top\boldsymbol{R}\boldsymbol{y'}-\min_{j\in BR_c^\rho(\boldsymbol{x})}(\boldsymbol{x'})^\top\boldsymbol{R}\boldsymbol{e}_j-V(\boldsymbol{z}).$$

Lemma 11. *It holds that for any direction $\boldsymbol{z'}=(\boldsymbol{x'},\boldsymbol{y'})\in\Delta^{n-1}\times\Delta^{n-1}$, and for any $\rho>0$,*

$$\nabla_{\boldsymbol{z'}}V(\boldsymbol{z})\le\nabla_{\rho,\boldsymbol{z'}}V(\boldsymbol{z}).$$

Lemma 12. *Given $\delta\in[0,1]$, let $\boldsymbol{z}=(\boldsymbol{x},\boldsymbol{y})$ be a strategy profile that is not a δ-Nash equilibrium. Then*

$$\nabla_{\rho,\boldsymbol{z'}}V(\boldsymbol{z})<-\delta,$$

where $\boldsymbol{z'}=(\boldsymbol{x'},\boldsymbol{y'})\in\Delta^{n-1}\times\Delta^{n-1}$ is a direction that minimizes the ρ-directional derivative.

The proofs of these lemmas and any other missing proof from this section are given in Appendix A.

[3] Here as the objective function we use the sum of the regrets instead of the maximum of the two regrets.

3.1 The Main Algorithm

We now present our algorithm. Algorithm 1 takes as input a game and 3 parameters, namely $\delta \in (0,1]$, which refers to the approximation guarantee that is desired, $\rho \in (0,1]$ which involves the approximation to the directional derivative, and ϵ, which refers to the size of the step taken in each iteration. Our theoretical analysis will require ρ and ϵ to be correlated.

Algorithm 1: The gradient descent-based algorithm.

INPUT: A game $(\boldsymbol{R}, -\boldsymbol{R})$, an approximation parameter $\delta \in (0,1]$ and constants $\rho, \varepsilon \in (0,1]$.
OUTPUT: A δ-NE strategy profile

Pick an arbitrary strategy profile $(\boldsymbol{x}, \boldsymbol{y})$
While $V(\boldsymbol{x}, \boldsymbol{y}) > \delta$
  $(\boldsymbol{x}', \boldsymbol{y}') = \text{FindDirection}(\boldsymbol{x}, \boldsymbol{y}, \rho)$
  $(\boldsymbol{x}, \boldsymbol{y}) = (1-\varepsilon) \cdot (\boldsymbol{x}, \boldsymbol{y}) + \varepsilon \cdot (\boldsymbol{x}', \boldsymbol{y}')$
Return $(\boldsymbol{x}, \boldsymbol{y})$

Algorithm 2: FindDirection$(\boldsymbol{x}, \boldsymbol{y}, \rho)$.

INPUT: A strategy profile $(\boldsymbol{x}, \boldsymbol{y})$ and parameter $\rho \in (0,1]$.
OUTPUT: The direction $(\boldsymbol{x}', \boldsymbol{y}')$ that minimizes the ρ-directional derivative.

Solve the linear program (w.r.t. $(\boldsymbol{x}', \boldsymbol{y}')$ and γ):
  minimize γ
    s.t. $\gamma \geq (\boldsymbol{e}_i)^\top \boldsymbol{R}\boldsymbol{y}' - (\boldsymbol{x}')^\top \boldsymbol{R}\boldsymbol{e}_j$,
      $i \in BR_r^\rho(\boldsymbol{y}), j \in BR_c^\rho(\boldsymbol{x})$ and $\boldsymbol{x}', \boldsymbol{y}' \in \Delta^{n-1}$
Return $(\boldsymbol{x}', \boldsymbol{y}')$

Observation 13. *If $\rho = 1$, then Algorithm 2 returns an exact Nash equilibrium of the game $(\boldsymbol{R}, -\boldsymbol{R})$.*

The proof of this simple observation is referred to Appendix B. We conclude the presentation of our main algorithm with the following remark.

Remark 14. *The choice of ρ demonstrates the trade off between global optimization (Linear Programming) and the descent-based approach. In the extreme case where $\rho = 1$, Observation 13 shows one iteration would suffice, solving the (large) linear program of the entire zero-sum game. On the other hand, when ρ is small, close to 0, then the method solves in each iteration rather small linear programs in Algorithm 2 (dependent on the sets $BR_c^\rho(\boldsymbol{x}), BR_r^\rho(\boldsymbol{y})$).*

3.2 Proof of Correctness and Rate of Convergence

Our main result is the following theorem.

Theorem 15. *For any constants $\delta, \rho \in (0,1]$, and with $\epsilon = \rho/2$, Algorithm 1 returns a δ-Nash equilibrium in bilinear zero-sum games after at most $O(\frac{1}{\rho \cdot \delta} \log \frac{1}{\delta})$ iterations, and with a geometric rate of convergence for the duality gap.*

In order to prove this theorem, we will start first with the following auxiliary lemma.

Lemma 16. *If $\varepsilon \leq \frac{\rho}{2}$, then it holds that*

$$\max\Big\{0, \max_{\overline{i \in BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon \cdot \boldsymbol{y}'\Big) - \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon \cdot \boldsymbol{y}'\Big)\Big\} = 0.$$

Similarly, for the column player, it holds that

$$\max\Big\{0, -\min_{\overline{j \in BR_c^\rho(\boldsymbol{x})}} \Big((1-\varepsilon)\cdot \boldsymbol{x} + \varepsilon \cdot \boldsymbol{x}'\Big)^\top \boldsymbol{R}\boldsymbol{e}_j + \min_{j \in BR_c^\rho(\boldsymbol{x})} \Big((1-\varepsilon)\cdot \boldsymbol{x} + \varepsilon \cdot \boldsymbol{x}'\Big)^\top \boldsymbol{R}\boldsymbol{e}_j\Big\} = 0.$$

We can now establish that the duality gap decreases geometrically, as long as we have not yet found a δ-approximate equilibrium. We first show an additive decrease.

Lemma 17. *Let $\epsilon \le \frac{\rho}{2}$ and suppose that after t iterations we are at a profile $(\boldsymbol{x}^t, \boldsymbol{y}^t)$, which is not a δ-Nash equilibrium. Then,*

$$V(\boldsymbol{x}^{t+1}, \boldsymbol{y}^{t+1}) \le V(\boldsymbol{x}^t, \boldsymbol{y}^t) - \epsilon \cdot \delta$$

where $(\boldsymbol{x}^{t+1}, \boldsymbol{y}^{t+1})$ is the strategy profile at iteration $t+1$.

Proof. To shorten notation, let $\boldsymbol{x}^t = \boldsymbol{x}, \boldsymbol{y}^t = \boldsymbol{y}, \boldsymbol{x}'^t = \boldsymbol{x}'$, $\boldsymbol{y}'^t = \boldsymbol{y}', \boldsymbol{z}^t = (\boldsymbol{x}, \boldsymbol{y}), \boldsymbol{z}^{t+1} = (\boldsymbol{x}^{t+1}, \boldsymbol{y}^{t+1})$. Then we have

$$(\boldsymbol{x}^{t+1}, \boldsymbol{y}^{t+1}) = ((1-\varepsilon) \cdot \boldsymbol{x} + \varepsilon \cdot \boldsymbol{x}', (1-\varepsilon) \cdot \boldsymbol{y} + \varepsilon \cdot \boldsymbol{y}').$$

Similar to Equation (1) in Appendix A.1, we have that

$$\max_i \boldsymbol{e}_i^\top \boldsymbol{R} \boldsymbol{y}^{t+1} = \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R} \boldsymbol{y}^{t+1} + \max\Big\{0, \max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R} \boldsymbol{y}^{t+1} - \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R} \boldsymbol{y}^{t+1}\Big\}.$$

Note that since $\varepsilon \le \frac{\rho}{2}$, Lemma 16 applies and zeroes out the last term. Respectively, we obtain that

$$\min_j (\boldsymbol{x}^{t+1})^\top \boldsymbol{R} \boldsymbol{e}_j = \min_{j \in BR_c^\rho(\boldsymbol{x})} (\boldsymbol{x}^{t+1})^\top \boldsymbol{R} \boldsymbol{e}_j$$

Hence,

$$\begin{aligned}
V(\boldsymbol{z}^{t+1}) &= \max_i \boldsymbol{e}_i^\top \boldsymbol{R} \boldsymbol{y}^{t+1} - \min_j (\boldsymbol{x}^{t+1})^\top \boldsymbol{R} \boldsymbol{e}_j \\
&= \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R} \boldsymbol{y}^{t+1} - \min_{j \in BR_c^\rho(\boldsymbol{x})} (\boldsymbol{x}^{t+1})^\top \boldsymbol{R} \boldsymbol{e}_j \\
&= \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon) \cdot \boldsymbol{y} + \varepsilon \cdot \boldsymbol{y}'\Big) - \min_{j \in BR_c^\rho(\boldsymbol{x})} \Big((1-\varepsilon) \cdot \boldsymbol{x} + \varepsilon \cdot \boldsymbol{x}'\Big)^\top \boldsymbol{R} \boldsymbol{e}_j \\
&\le (1-\varepsilon) \max_i \boldsymbol{e}_i^\top \boldsymbol{R} \boldsymbol{y} + \varepsilon \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R} \boldsymbol{y}' - (1-\varepsilon) \min_j \boldsymbol{x}^\top \boldsymbol{R} \boldsymbol{e}_j - \varepsilon \min_{j \in BR_c^\rho(\boldsymbol{x})} (\boldsymbol{x}')^\top \boldsymbol{R} \boldsymbol{e}_j \\
&= \max_i \boldsymbol{e}_i^\top \boldsymbol{R} \boldsymbol{y} - \min_j \boldsymbol{x}^\top \boldsymbol{R} \boldsymbol{e}_j + \varepsilon \Big(\max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R} \boldsymbol{y}' - \min_{j \in BR_c^\rho(\boldsymbol{x})} (\boldsymbol{x}')^\top \boldsymbol{R} \boldsymbol{e}_j - \max_i \boldsymbol{e}_i^\top \boldsymbol{R} \boldsymbol{y} + \min_j \boldsymbol{x}^\top \boldsymbol{R} \boldsymbol{e}_j \Big) \\
&= V(\boldsymbol{z}^t) + \varepsilon \cdot \nabla_{\rho, \boldsymbol{z}'} V(\boldsymbol{z}^t) < V(\boldsymbol{z}^t) - \varepsilon \cdot \delta,
\end{aligned}$$

where the last inequality follows from Lemma 12. ∎

The next step is to turn the additive decrease of Lemma 17 into a multiplicative decrease.

Corollary 18. *For $\epsilon = \rho/2$, we have that*

$$V(\boldsymbol{x}^{t+1}, \boldsymbol{y}^{t+1}) \le \left(1 - \frac{\rho \cdot \delta}{4}\right) \cdot V(\boldsymbol{x}^t, \boldsymbol{y}^t)$$

Proof. Using Lemma 17, we get that $V(\boldsymbol{z}^{t+1}) \le (1-c) \cdot V(\boldsymbol{z}^t)$ with $c = \frac{\varepsilon \cdot \delta}{V(\boldsymbol{z}^t)} \ge \frac{\rho \cdot \delta}{4}$, since $V(\boldsymbol{x}, \boldsymbol{y}) \le 2$ for any profile, and $\varepsilon = \frac{\rho}{2}$. ∎

Finally, we can complete the proof of our main theorem.

Proof of Theorem 15. We have already proved the geometric decrease of the duality gap, for constant ρ and δ. Hence, the algorithm eventually will satisfy that the duality gap is at most δ and will terminate with a δ-NE. It remains to bound the number of iterations that are needed. Suppose that the algorithm terminates after t iterations, with profile $(\boldsymbol{x}^t, \boldsymbol{y}^t)$. By repeatedly applying Corollary 18, we have that

$$V(\boldsymbol{x}^t, \boldsymbol{y}^t) \le (1-c)^t \cdot V(\boldsymbol{x}^0, \boldsymbol{y}^0)$$

with $c = \frac{\rho \cdot \delta}{4}$. In order to ensure that $V(\boldsymbol{x}^t, \boldsymbol{y}^t) \le \delta$, it suffices to have that $2 \cdot (1-c)^t \le \delta$, since $V(\boldsymbol{x}^0, \boldsymbol{y}^0) \le 2$.

$$2(1-c)^t \le \delta \implies t \ge \frac{\log \frac{2}{\delta}}{\log \frac{1}{1-c}} \implies t \ge \frac{1-c}{c} \log \frac{2}{\delta}$$

where the last inequality holds due to $\log x \le x - 1$, for $x \ge 1$. Since $\frac{1-c}{c} = O(\frac{1}{c})$, the proof is completed by substituting the value of c. ∎

3.3 Decaying Schedule Speedups

In this section, we present a different implementation of our main approach, which results in an improved analysis. The idea is to gradually decay δ and use it to bound c, instead of the more coarse approximation of $V(\boldsymbol{x}, \boldsymbol{y}) \leq 2$, that we used in the proof of Theorem 15. This is presented as Algorithm 3.

Algorithm 3: Decaying Delta Speedup.

INPUT: A game $(\boldsymbol{R}, -\boldsymbol{R})$, an approximation parameter $\delta \in (0, 1]$ and a constant $\rho \in (0, 1]$.
OUTPUT: A δ-NE strategy profile.

Pick an arbitrary strategy profile $(\boldsymbol{x}, \boldsymbol{y})$
Set $i = 0, \delta_0 = 1$ and $\varepsilon = \frac{\rho}{2}$
While TRUE
  Set $i = i + 1$ and $\delta_i = \delta_{i-1}/2$
  Update $(\boldsymbol{x}, \boldsymbol{y})$ via Algorithm 1 $\Big((\boldsymbol{R}, -\boldsymbol{R}), \delta_i, \rho, \varepsilon\Big)$
  If $\delta_i \leq \delta$: break
Return $(\boldsymbol{x}, \boldsymbol{y})$

Theorem 19. *Algorithm 3 maintains a geometric decrease rate in the duality gap and reaches a δ-NE after at most $O\Big(\frac{1}{\rho} \cdot \log\Big(\frac{1}{\delta}\Big)\Big)$ iterations.*

Proof. We think of the iterations of the entire algorithm as divided into epochs, where each epoch corresponds to a new value for δ. Fix an epoch i, with $i > 0$. Within this epoch, Algorithm 1 is run with approximation parameter δ_i. Consider an arbitrary iteration of Algorithm 1 during this epoch, say at time $t + 1$, starting with the profile $\boldsymbol{z}^t = (\boldsymbol{x}^t, \boldsymbol{y}^t)$ and ending at the profile $\boldsymbol{z}^{t+1} = (\boldsymbol{x}^{t+1}, \boldsymbol{y}^{t+1})$. By Lemma 17, we have that $V(\boldsymbol{z}^{t+1}) \leq V(\boldsymbol{z}^t) - \epsilon \cdot \delta_i = (1 - c_i) \cdot V(\boldsymbol{z}^t)$, where $c_i = \frac{\epsilon \cdot \delta_i}{V(\boldsymbol{z}^t)} = \frac{\rho \cdot \delta_i}{2 \cdot V(\boldsymbol{z}^t)}$. Since we are at epoch i, we know that $V(\boldsymbol{z}^t) \leq \delta_{i-1} = 2 \cdot \delta_i$, because the duality gap was at most δ_{i-1} at the beginning of epoch i and within the epoch it only decreases further due to Lemma 17 (for epoch 1, it is even better, since $V(\boldsymbol{z}^t) \leq V(\boldsymbol{z}^0) \leq 2 = 2\delta_0 \leq 4\delta_1$, where $\boldsymbol{z}^0$ is the initial profile). Therefore, $c_i \geq \frac{\rho \cdot \delta_i}{2 \cdot \delta_{i-1}} = \frac{\rho}{4}$. Hence, we have established that in any iteration, regardless of the epoch:

$$V(\boldsymbol{z}^{t+1}) \leq \left(1 - \frac{\rho}{4}\right) \cdot V(\boldsymbol{z}^t) \leq \left(1 - \frac{\rho}{4}\right)^t \cdot V(\boldsymbol{z}^0).$$

Since ρ is constant, we have a geometric decrease, and this proves the first part of the theorem.

To bound the total number of iterations, let t_i be the number of iterations of Algorithm 1 within epoch i, after which, the algorithm achieves a δ_i-NE. Then, similar to the proof of Theorem 15, and since in the beginning of epoch i, the duality gap is at most δ_{i-1}, we have that t_i should satisfy

$$(1 - c_i)^{t_i} \cdot \delta_{i-1} \leq \delta_i \implies t_i \geq \frac{1}{\log \frac{1}{1 - c_i}} \implies t_i \geq \frac{1 - c_i}{c_i}$$

Thus, at epoch i, we need $t_i = O(\frac{1}{\rho})$ to reach a δ_i-NE. Next, note that if k is the total number of epochs required to achieve a δ-NE, when starting with δ_0, it holds that $\frac{\delta_0}{2^k} \leq \delta \implies k \geq \log \frac{\delta_0}{\delta}$. Since $\delta_0 = 1$, the number of required epochs is $O(\log \frac{1}{\delta})$. Therefore, the total number of iterations for the entire algorithm is $O\Big(\frac{1}{\rho} \cdot \log \frac{1}{\delta}\Big)$. ∎

To demonstrate the flexibility of our approach, we conclude the theoretical exploration with yet another variation, where we additionally use a decreasing schedule for the value of ρ. Specifically, this gives rise to the following scheme which we refer to as Algorithm 4.

- Use the same schedule for δ_i as Algorithm 3.
- At iteration i set $\rho_i = \sqrt{\delta_i}$, for Algorithm 1 (with $\varepsilon_i = \frac{\rho_i}{2}$).

Note that we have now eliminated the dependence on ρ but at the expense of making more expensive the dependence on δ.

Algorithm 4: Decaying Delta and Rho Speedup.

INPUT: A game $(\boldsymbol{R}, -\boldsymbol{R})$ and an approximation parameter $\delta \in (0, 1]$.
OUTPUT: A δ-NE strategy profile.

Pick an arbitrary strategy profile $(\boldsymbol{x}, \boldsymbol{y})$
Set $i = 0$ and $\delta_0 = 1$
While TRUE
  Set $i = i + 1$ and $\delta_i = \delta_{i-1}/2$
  Set $\rho_i = \sqrt{\delta_i}$ and $\varepsilon = \frac{\rho_i}{2}$
  Update $(\boldsymbol{x}, \boldsymbol{y})$ via Algorithm 1 $\Big((\boldsymbol{R}, -\boldsymbol{R}), \delta_i, \rho_i, \varepsilon\Big)$
  If $\delta_i \leq \delta$: break
Return $(\boldsymbol{x}, \boldsymbol{y})$

Theorem 20. *Algorithm 4 reaches a δ-Nash equilibrium after at most $O\Big(\frac{1}{\sqrt{\delta}}\Big)$ iterations, for any constant δ.*

Proof. Following the same analysis as in the proof of Theorem 19, we obtain that for any 2 consecutive iterations within epoch i, we have that $V(\boldsymbol{z}^{t+1}) \leq (1 - c_i) \cdot V(\boldsymbol{z}^t)$, where $c_i \geq \frac{\rho_i}{4}$. This implies that if t_i is the number of iterations needed within epoch i, it holds that $t_i \leq \lceil \frac{4}{\rho_i} \rceil \leq \frac{4}{\sqrt{\delta_i}} + 1$.

We also have again that the total number of epochs is $k = O(\log(1/\delta))$. Putting everything together, and since $\delta_i = \frac{1}{2^i}$, the total number of iterations is

$$t = \sum_{i=1}^{k} t_i \leq \sum_{i=1}^{k} (\frac{4}{\sqrt{\delta_i}} + 1) = k + 4 \cdot \sum_{i=1}^{k} \frac{1}{\sqrt{\frac{1}{2^i}}} = k + O\Big(2^{k/2}\Big) = O\Big(\frac{1}{\sqrt{\delta}}\Big)$$

□

4 Experimental Evaluation

All our algorithms were implemented in Python, and the exact specifications can be found in Appendix D. Before proceeding to our main findings, we exhibit first that the geometric decrease in the duality gap can indeed be observed experimentally. Figure 1 shows a typical behavior of our algorithms, in terms of the duality gap. The figure here is for a random game of size $n = 1000$.

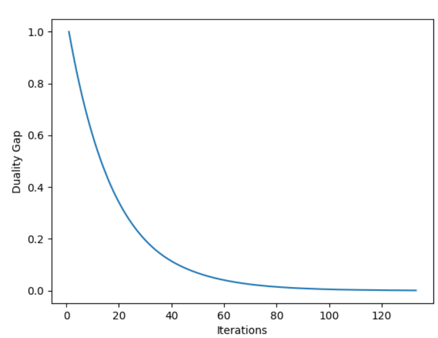

Figure 1: The decrease in the duality gap for a random game.

4.1 From Theory to Implementation

We deem useful to discuss first how to approach the selection of the parameters that the algorithms depend on. We have seen in Algorithm 1 and its variants two families of parameters: ρ_i and δ_i. A third parameter is the learning rate ε, which is the step size that we take in each iteration.

Choice of $\boldsymbol{\varepsilon}$ We have established that as long as $\varepsilon \leq \rho/2$, the points along the line $(1-\varepsilon)\cdot(\boldsymbol{x},\boldsymbol{y})+\varepsilon\cdot(\boldsymbol{x}',\boldsymbol{y}')$ decrease the duality gap (Lemma 17). Note, though, that the problem of minimizing V along this set is a convex optimization problem. Hence, we can try to find the optimal ε_i at each iteration i, and there are a few possible approaches for this: line search, ternary search or even solving it exactly using dynamic programming. We decided to use the following heuristic: for large values of the duality gap, namely $V > 0.1$, we employ ternary search and as the duality gap decreases we use line search but only on a small part of the line. More specifically, once $V \leq 0.1$ we start with $\varepsilon = 0.2$ and decrease it by 10% across iterations. We decided upon this method since we noticed that experiments conform to theory for smaller values of V and ρ. Finally, a more ML-like approach would be to set a constant ε, similarly to a constant step size η in gradient methods. While this approach has merit, it did not show improved performance (see more in Appendix D.1).

Choice of $\boldsymbol{\rho}$ (and a new algorithm) The most critical parameter regarding the running time of our algorithms is ρ, since it controls the size of the LPs in Algorithm 2, i.e., the number of constraints, via the sets of ρ-approximate best responses, $BR_r^\rho(\boldsymbol{y})$ and $BR_c^\rho(\boldsymbol{x})$. We need ρ to be large enough to avoid having only a single best response, in which case our algorithms reduces to Best Response Dynamics, while at the same time it should be small enough so that the LPs have small size and we can solve them fast. Our experimentation did not reveal any particular range of ρ with a consistently better performance. As a result, in addition to our existing algorithms, we developed one more approach, independent of ρ: we fix a number k (much smaller than n), and in every iteration, we include in the approximate best response set of each player its top k better responses. We refer to this approach as the *Fixed Support Variant* in the sequel. We used $k=100$ for our experiments and point to Appendix D.2 for justification.

Optimizing FindDirection For this algorithm we use two implementation tricks. The first one is quite simple: it is easy to observe that the LP of Algorithm 2 is equivalent to solving two smaller LPs (see Appendix B); it turns out that solving it this way is faster. The second trick revolves around ρ. Recall that the direction we find is itself an approximation. Hence, solving the LP approximately is meaningful, in the sense that it provides an even coarser approximate direction. It turns out that even a 0.1 approximate solution (which is achievable by setting an appropriate parameter in the LP solver that we used) works for most cases, and results in significantly less running time (see also the discussion in Appendix D.4).

4.2 Comparisons between Our Variants

We report first on our comparisons between Algorithm 3 with $\rho = 0.001$, henceforth called the *Constant ρ Variant*, Algorithm 4 with $\rho_i = 0.01\sqrt{\delta_i}$, which we refer to as the *Adaptive ρ Variant* and our Fixed Support Variant discussed in Section 4.1. We note that for the variant with the adaptive value of ρ, we did not follow precisely the values presented by our theoretical analysis, of $\rho_i = \sqrt{\delta_i}$. Although theoretically equivalent, this change was only to avoid a blowup in the number of best response strategies used in Algorithm 2 during the first iterations, i.e. for δ_1 and δ_2 we would have $\rho > 0.7$, which is quite large and undesirable.

To test our algorithms we generated random games of size $n \times n$, where each entry is picked uniformly at random from $[0, 1]$. The size of the games range from 500 to 5000 pure strategies with a step of 500. For each size we generate 30 games and solve them to an accuracy of $\delta = 0.01$. We used two types of initialization in all methods, the fully uniform strategy profile and the profile $(\boldsymbol{e}_1, \boldsymbol{e}_1)$, i.e., first row, first column. The latter has the advantage of not being too close to a Nash equilibrium from the start, in almost all games, and reveals more clearly the exploration that the method performs.

The averaged results are presented in Figure 2, where we show both the actual time and the number of iterations. In terms of actual time, our Fixed Support variant is the clear winner. Although Figure 2 reveals that as n grows, the Fixed ρ variant attains a lower number of iterations, this does not translate into improved running time. The intuition for this is that as n grows and ρ remains constant, we expect a larger

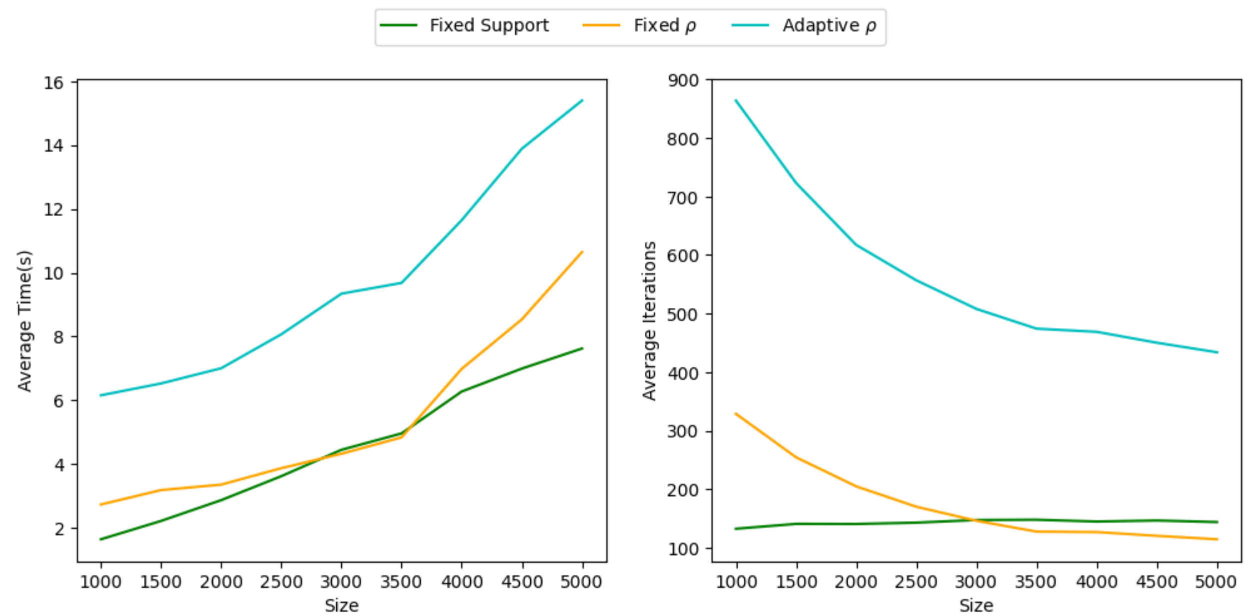

Figure 2: Average time and number of iterations for our variants

number of strategies to be ρ-best responses. Consequently, the LP in Algorithm 2 is closer to the full LP and thus more informative, but at the same time more expensive to solve.

As a result of these comparisons, we select our Fixed Support variant as the variant to compare against other methods from the literature in the next subsection.

4.3 Comparisons with LP and Gradient Methods

We compared our Fixed Support variant against solving directly the full LP with a standard LP solver, and against a prominent first order method. Regarding the LP solver, we used the standard method of SciPy. We note that we used the same method for the smaller LPs that we solve in Algorithm 2 of our methods. To maintain an equal comparison with our algorithms, we used a tolerance of 0.01. As for first order methods, we used Optimistic Gradient Descent Ascent (OGDA), which is among the fastest gradient based methods, with step size $\eta = 0.01$. We refer to Appendix C for its definition. Another popular method is Optimistic Multiplicative Weights Update (OMWU), which however does not behave as well in practice, as also explained in Cai et al. [2024].

For each value of n that we used, we generated 50 uniformly random games and 50 games using the Gaussian distribution. We also generated more structured but still random games, such as games with low rank. We present here the comparisons for the uniformly random games and we refer to Appendices D.3 and D.4 for the other classes of games. As in Section 4.2, we used two different initializations: starting from $(\boldsymbol{e}_1, \boldsymbol{e}_1)$ and starting from the uniform strategy profile: $(\frac{1}{n}, \ldots, \frac{1}{n})$. The average running time can be seen in Figure 3. We summarize our findings as follows:

- The LP solver was far slower, even for lower values of n, as shown in the left subplot, and we dropped it from the experiments with larger games.
- When the initialization is (e_1, e_1) (or any pure strategy profile), the advantage of our method is more clear (see left subplot of Figure 3). When we start with the uniform profile, we observe that our method is slower for smaller games but becomes faster in very large games (right subplot).
- Another observation is that our method seems smoother with less sharp jumps than OGDA when starting from $(\boldsymbol{e}_1, \boldsymbol{e}_1)$ while the opposite holds for the uniform profile.

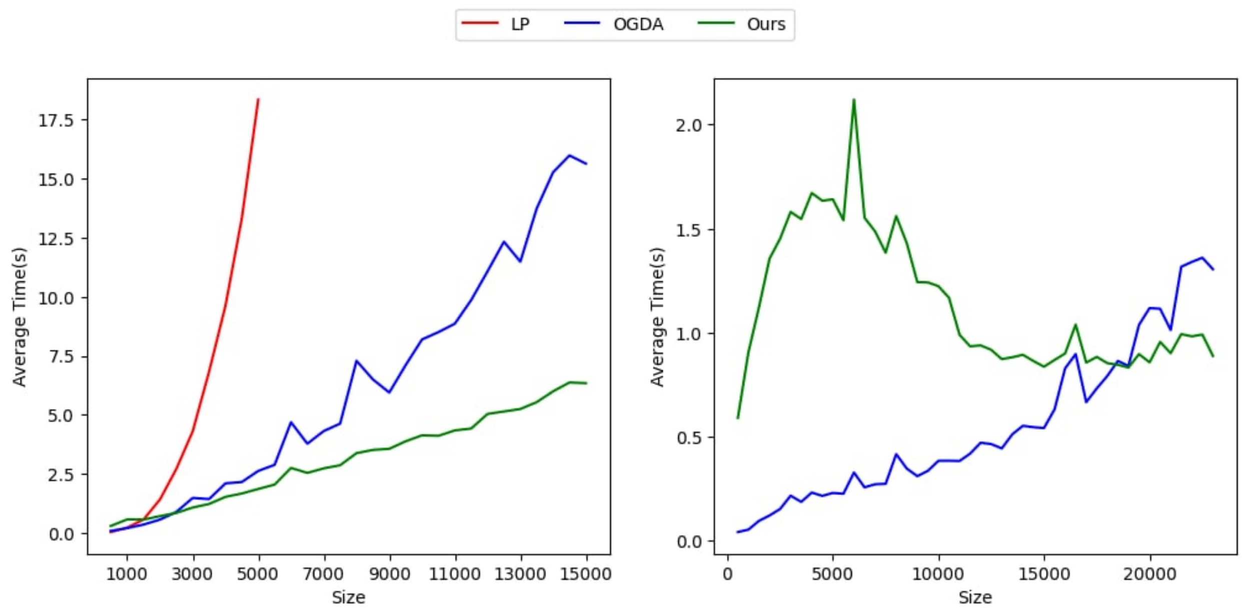

Figure 3: Time comparison between our Fixed Support Variant, LP solver and Optimistic Gradient Descent-Ascent

We view as the main takeaway of our experiments that our method is comparable to OGDA and in several cases even outperforms OGDA. One limitation of our current implementation is the choice of $\delta = 0.01$. For much lower accuracies, our methods occasionally get stuck. We therefore feel that the overall approach deserves further exploration, especially on potential ways of accelerating its execution.

5 Conclusions

We have analyzed a descent-based method for the duality gap in zero-sum games. Our goal has been to demonstrate the potential of such algorithms as a proof of concept. We expect that our method can be further optimized in practice and find this a promising direction for future work. In particular, one idea to explore is whether we can reuse the LP solutions we get in Algorithm 2 from one iteration to the next (since we only change the current solution slightly by a step of size ϵ). Exploring such *warm start* strategies (see e.g. Yildirim and Wright [2002]) could provide significant speedups.

Acknowledgments

This work has been implemented within the framework of the H.F.R.I call "Basic research Financing (Horizontal support of all Sciences)" under the National Recovery and Resilience Plan "Greece 2.0" funded by the European Union – NextGenerationEU (H.F.R.I. Project Number: 15877).

References

I. Adler. The equivalence of linear programs and zero-sum games. *Int. J. Game Theory*, 42(1):165–177, 2013.

S. Arora, E. Hazan, and S. Kale. The multiplicative weights update method: a meta-algorithm and applications. *Theory Comput.*, 8(1):121–164, 2012.

J. P. Bailey and G. Piliouras. Multiplicative weights update in zero-sum games. In *Proceedings of the Conference on Economics and Computation (EC'18)*, pages 321–338, 2018.

G. W. Brown. Iterative solution of games by fictitious play. In *Activity Analysis of Production and Allocation*. 1951.

Y. Cai and W. Zheng. Doubly optimal no-regret learning in monotone games. In *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pages 3507–3524. PMLR, 2023.

Y. Cai, A. Oikonomou, and W. Zheng. Finite-time last-iterate convergence for learning in multi-player games. In *Proceedings of the Annual Conference on Neural Information Processing Systems (NeurIPS'22)*, 2022.

Y. Cai, G. Farina, J. Grand-Clément, C. Kroer, C. Lee, H. Luo, and W. Zheng. Fast last-iterate convergence of learning in games requires forgetful algorithms. *CoRR*, abs/2406.10631, 2024.

C. Daskalakis and I. Panageas. Last-iterate convergence: Zero-sum games and constrained min-max optimization. In *Proceedings of the ITCS'19*, 2019.

C. Daskalakis, A. Ilyas, V. Syrgkanis, and H. Zeng. Training GANs with optimism. In *Proceedings of the International Conference on Learning Representations (ICLR'18)*, 2018.

C. Daskalakis, S. Skoulakis, and M. Zampetakis. The complexity of constrained min-max optimization. In *STOC '21: 53rd Annual ACM SIGACT Symposium on Theory of Computing, Virtual Event, Italy, June 21-25, 2021*, pages 1466–1478. ACM, 2021.

A. Deligkas, J. Fearnley, R. Savani, and P. G. Spirakis. Computing approximate Nash equilibria in polymatrix games. *Algorithmica*, 77(2):487–514, 2017.

A. Deligkas, M. Fasoulakis, and E. Markakis. A polynomial-time algorithm for 1/3-approximate Nash equilibria in bimatrix games. *ACM Trans. Algorithms*, 19(4):31:1–31:17, 2023.

J. Diakonikolas, C. Daskalakis, and M. I. Jordan. Efficient methods for structured nonconvex-nonconcave min-max optimization. In A. Banerjee and K. Fukumizu, editors, *The 24th International Conference on Artificial Intelligence and Statistics, AISTATS 2021, April 13-15, 2021, Virtual Event*, volume 130 of *Proceedings of Machine Learning Research*, pages 2746–2754. PMLR, 2021.

M. Fasoulakis, E. Markakis, Y. Pantazis, and C. Varsos. Forward looking best-response multiplicative weights update methods for bilinear zero-sum games. In *Proceedings of the International Conference on Artificial Intelligence and Statistics (AISTATS'22)*, pages 11096–11117, 2022.

A. Gilpin, J. Pena, and T. Sandholm. First-order algorithm with convergence for-equilibrium in two-person zero-sum games. *Mathematical programming*, 133(1):279–298, 2012.

N. Golowich, S. Pattathil, and C. Daskalakis. Tight last-iterate convergence rates for no-regret learning in multi-player games. In *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020.

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative Adversarial Nets. In *Proceedings of Annual Conference on Neural Information Processing Systems (NIPS '14)*, pages 2672–2680, 2014.

S. Hoda, A. Gilpin, J. Pena, and T. Sandholm. Smoothing techniques for computing Nash equilibria of sequential games. *Mathematics of Operations Research*, 35(2):494–512, 2010.

G. Korpelevich. The extragradient method for finding saddle points and other problems. *Matecon*, 12:747–756, 1976.

T. Liang and J. Stokes. Interaction matters: A note on non-asymptotic local convergence of generative adversarial networks. In *Proceedings of The 22nd International Conference on Artificial Intelligence and Statistics, AISTATS'19*, pages 907–915, 2019.

H. Lu and J. Yang. On the infimal sub-differential size of primal-dual hybrid gradient method and beyond. *CoRR*, abs/2206.12061, 2023.

P. Mertikopoulos, C. H. Papadimitriou, and G. Piliouras. Cycles in adversarial regularized learning. In *Proceedings of the Twenty-Ninth Annual ACM-SIAM Symposium on Discrete Algorithms, SODA 2018*, pages 2703–2717. SIAM, 2018.

J. Nash. Non-cooperative games. *Annals of Mathematics*, 54 (2), 1951.

H. Nikaido and K. Isoda. Note on noncooperative convex games. *Pacific Journal of Mathematics*, 5(1):807815, 1955.

L. Popov. A modification of the Arrow-Hurwicz method for search of saddle points. *Mathematical notes of the Academy of Sciences of the USSR*, 28:845–848, 1980.

A. U. Raghunathan, A. Cherian, and D. K. Jha. Game theoretic optimization via gradient-based Nikaido-Isoda function. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019*, volume 97 of *Proceedings of Machine Learning Research*, pages 5291–5300. PMLR, 2019.

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

H. Tsaknakis and P. G. Spirakis. An optimization approach for approximate Nash equilibria. *Internet Math.*, 5(4):365–382, 2008.

J. Von Neumann. Zur theorie der gesellschaftsspiele. *Math. Ann.*, 100:295–320, 1928.

C.-Y. Wei, C.-W. Lee, M. Zhang, and H. Luo. Linear last-iterate convergence in constrained saddle-point optimization. In *Proceedings of the 9th International Conference on Learning Representations ICLR '21*, 2021.

E. A. Yildirim and S. J. Wright. Warm-start strategies in interior-point methods for linear programming. *SIAM J. Optim.*, 12(3):782–810, 2002.

A Missing Proofs from Section 3

A.1 Proof of Theorem 8

Using the definition of the duality gap, the directional derivative is equal to

$$\nabla_{(\boldsymbol{x}',\boldsymbol{y}')}V(\boldsymbol{x},\boldsymbol{y}) = \lim_{\varepsilon\to 0} \frac{\max_i \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon\cdot \boldsymbol{y}'\Big) - \min_j \Big((1-\varepsilon)\cdot \boldsymbol{x} + \varepsilon\cdot \boldsymbol{x}'\Big)^\top \boldsymbol{R}\boldsymbol{e}_j - V(\boldsymbol{x},\boldsymbol{y})}{\varepsilon}.$$

However, we can write up the term $\max_i \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon\cdot \boldsymbol{y}'\Big)$, similarly to Deligkas et al. [2017], Tsaknakis and Spirakis [2008], as

$$\begin{aligned} &\max_i \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon\cdot \boldsymbol{y}'\Big) = \max_{i\in BR_r(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon\cdot \boldsymbol{y}'\Big) \\ &+ \max\Big\{0, \max_{i\in \overline{BR_r(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon\cdot \boldsymbol{y}'\Big) - \max_{i\in BR_r(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon\cdot \boldsymbol{y}'\Big)\Big\}, \end{aligned} \tag{1}$$

where $\overline{BR_r(\boldsymbol{y})}$ is the complement set of $BR_r(\boldsymbol{y})$. Similarly,

$$\begin{aligned} &\min_j \Big((1-\varepsilon)\cdot \boldsymbol{x} + \varepsilon\cdot \boldsymbol{x}'\Big)^\top R\boldsymbol{e}_j = \min_{j\in BR_c(\boldsymbol{x})} \Big((1-\varepsilon)\cdot \boldsymbol{x} + \varepsilon\cdot \boldsymbol{x}'\Big)^\top R\boldsymbol{e}_j \\ &- \max\Big\{0, -\min_{j\in \overline{BR_c(\boldsymbol{x})}} \Big((1-\varepsilon)\cdot \boldsymbol{x} + \varepsilon\cdot \boldsymbol{x}'\Big)^\top R\boldsymbol{e}_j + \min_{j\in BR_c(\boldsymbol{x})} \Big((1-\varepsilon)\cdot \boldsymbol{x} + \varepsilon\cdot \boldsymbol{x}'\Big)^\top R\boldsymbol{e}_j\Big\}. \end{aligned} \tag{2}$$

Arguing in a similar fashion as in Deligkas et al. [2017], there exists $\epsilon^* > 0$ such that for $\epsilon \le \epsilon^*$, the term

$$\max\Big\{0, \max_{i\in \overline{BR_r(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon\cdot \boldsymbol{y}'\Big) - \max_{i\in BR_r(\boldsymbol{y})} \boldsymbol{e}_i^\top R\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon\cdot \boldsymbol{y}'\Big)\Big\}$$

is zero, and hence it can be ignored when we take the limit of $\epsilon \to 0$. In the same manner, the corresponding term for the column player also becomes 0.

Note also that for any $i, j \in BR_r(\boldsymbol{y})$, we have that $\boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} = \boldsymbol{e}_j^\top \boldsymbol{R}\boldsymbol{y}$, and hence the term $\max_{i\in BR_r(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}$ is independent of the row we choose, thus $\max_{i\in BR_r(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon\cdot \boldsymbol{y}'\Big) = (1-\varepsilon)\cdot \max_{i\in BR_r(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} + \varepsilon\cdot \max_{i\in BR_r(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}'$, similar for the min part. By using this below, we conclude that the directional derivative equals to

$$\begin{aligned} &\lim_{\varepsilon\to 0} \frac{\max_{i\in BR_r(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon\cdot \boldsymbol{y}'\Big) - \min_{j\in BR_c(\boldsymbol{x})} \Big((1-\varepsilon)\cdot \boldsymbol{x} + \varepsilon\cdot \boldsymbol{x}'\Big)^\top \boldsymbol{R}\boldsymbol{e}_j - V(\boldsymbol{x},\boldsymbol{y})}{\varepsilon} \\ &= \lim_{\varepsilon\to 0} \frac{(1-\varepsilon)\cdot \max_{i\in BR_r(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} + \varepsilon\cdot \max_{i\in BR_r(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}'}{\varepsilon} \\ &\quad - \frac{(1-\varepsilon)\cdot \min_{j\in BR_c(\boldsymbol{x})} x^\top \boldsymbol{R}\boldsymbol{e}_j + \varepsilon\cdot \min_{j\in BR_c(\boldsymbol{x})} (\boldsymbol{x}')^\top \boldsymbol{R}\boldsymbol{e}_j + V(\boldsymbol{x},\boldsymbol{y})}{\varepsilon} \\ &= \lim_{\varepsilon\to 0} \frac{(1-\varepsilon)\cdot V(\boldsymbol{x},\boldsymbol{y}) + \varepsilon\cdot \Big(\max_{i\in BR_r(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}' - \min_{j\in BR_c(\boldsymbol{x})} (\boldsymbol{x}')^\top \boldsymbol{R}\boldsymbol{e}_j\Big) - V(\boldsymbol{x},\boldsymbol{y})}{\varepsilon} \\ &= \max_{i\in BR_r(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}' - \min_{j\in BR_c(\boldsymbol{x})} (\boldsymbol{x}')^\top \boldsymbol{R}\boldsymbol{e}_j - V(\boldsymbol{x},\boldsymbol{y}). \end{aligned}$$

as claimed.

A.2 Proof of Lemma 11

By definition, we have that

$$
\begin{aligned}
\nabla_{\boldsymbol{z}'} V(\boldsymbol{z}) &= \max_{i \in BR_r(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}' - \min_{j \in BR_c(\boldsymbol{x})} (\boldsymbol{x}')^\top \boldsymbol{R}\boldsymbol{e}_j - V(\boldsymbol{z}) \\
&\leq \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}' - \min_{j \in BR_c^\rho(\boldsymbol{x})} (\boldsymbol{x}')^\top \boldsymbol{R}\boldsymbol{e}_j - V(\boldsymbol{z}) \\
&= \nabla_{\rho,\boldsymbol{z}'} V(\boldsymbol{z}),
\end{aligned}
$$

where the first inequality holds since, by definition, $BR_r(\boldsymbol{y}) \subseteq BR_r^\rho(\boldsymbol{y})$ and $BR_c(\boldsymbol{x}) \subseteq BR_c^\rho(\boldsymbol{x})$.

A.3 Proof of Lemma 12

Let $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ be a Nash equilibrium, then it holds $V(\boldsymbol{x}^*, \boldsymbol{y}^*) = 0$ by the definition of the NE, but since the values of $V(\boldsymbol{x}, \boldsymbol{y}) \in [0, 2]$ this implies that $(\boldsymbol{x}^*, \boldsymbol{y}^*)$ is a global minimum of the function in its domain. Let now a strategy profile $(\boldsymbol{x}, \boldsymbol{y})$ such that $V(\boldsymbol{x}, \boldsymbol{y}) = 0 = f_{\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{y}) + f_{-\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{y})$, this trivially implies that $f_{\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{y}) = 0$ and $f_{-\boldsymbol{R}}(\boldsymbol{x}, \boldsymbol{y}) = 0$ since $f_{\boldsymbol{R}}, f_{-\boldsymbol{R}} \in [0, 1]$, thus we have that $(\boldsymbol{x}, \boldsymbol{y})$ is a NE in the zero-sum game.

A.4 Proof of Lemma 16

Firstly, we have that

$$
\max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon \cdot \boldsymbol{y}'\Big) \geq \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y}\Big) = \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} - \varepsilon \cdot \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}.
$$

By the definition of the max function, we have

$$
\max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon \cdot \boldsymbol{y}'\Big) \leq (1-\varepsilon)\cdot \max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} + \varepsilon \cdot \max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}'.
$$

These two bounds give

$$
\begin{aligned}
&\max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon \cdot \boldsymbol{y}'\Big) - \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\Big((1-\varepsilon)\cdot \boldsymbol{y} + \varepsilon \cdot \boldsymbol{y}'\Big) \\
&\leq \max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} + \varepsilon \cdot \Big(\max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}' - \max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}\Big) - \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} + \varepsilon \cdot \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} \\
&= \max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} - \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} + \varepsilon \cdot \Big(\max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}' - \max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} + \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}\Big).
\end{aligned}
$$

By the definition of ρ-best-response, we have that

$$
\max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} - \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} < -\rho.
$$

Furthermore, we have that $\max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}' - \max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} + \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} \leq 2$, since $R_{ij} \leq 1$. Thus, we have that

$$
\max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} - \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} + \varepsilon \cdot \Big(\max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}' - \max_{i \in \overline{BR_r^\rho(\boldsymbol{y})}} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y} + \max_{i \in BR_r^\rho(\boldsymbol{y})} \boldsymbol{e}_i^\top \boldsymbol{R}\boldsymbol{y}\Big) < -\rho + 2\cdot\varepsilon.
$$

So, we want to find a value of ε such that $-\rho + 2\cdot\varepsilon \leq 0$, which holds for $\varepsilon \leq \frac{\rho}{2}$. In a very similar fashion, for the second part of the Lemma, we have

$$
\min_{j \in \overline{BR_c^\rho(\boldsymbol{x})}} \Big((1-\varepsilon)\cdot \boldsymbol{x} + \varepsilon \cdot \boldsymbol{x}'\Big)^\top R\boldsymbol{e}_j \geq (1-\varepsilon)\cdot \min_{j \in \overline{BR_c^\rho(\boldsymbol{x})}} \boldsymbol{x}^\top R\boldsymbol{e}_j + \varepsilon \cdot \min_{j \in \overline{BR_c^\rho(\boldsymbol{x})}} (\boldsymbol{x}')^\top R\boldsymbol{e}_j.
$$

Furthermore,

$$\min_{j\in BR_c^\rho(\boldsymbol{x})}\Big((1-\varepsilon)\cdot\boldsymbol{x}+\varepsilon\cdot\boldsymbol{x}'\Big)^\top R\boldsymbol{e}_j = -\max_{j\in BR_c^\rho(\boldsymbol{x})}\Big((1-\varepsilon)\cdot\boldsymbol{x}+\varepsilon\cdot\boldsymbol{x}'\Big)^\top(-R)\boldsymbol{e}_j$$
$$\le -(1-\varepsilon)\cdot\max_{j\in BR_c^\rho(\boldsymbol{x})}\boldsymbol{x}^\top(-R)\boldsymbol{e}_j = (1-\varepsilon)\cdot\min_{j\in BR_c^\rho(\boldsymbol{x})}\boldsymbol{x}^\top R\boldsymbol{e}_j.$$

The above inequality holds since $\max_j\Big((1-\varepsilon)\cdot\boldsymbol{x}+\varepsilon\cdot\boldsymbol{x}'\Big)^\top(-R)\boldsymbol{e}_j \ge \max_j\Big((1-\varepsilon)\cdot\boldsymbol{x}\Big)^\top(-R)\boldsymbol{e}_j = (1-\varepsilon)\cdot\max_{j\in BR_c^\rho(\boldsymbol{x})}\boldsymbol{x}^\top(-R)\boldsymbol{e}_j$. Thus, these two bounds give

$$-\min_{j\in \overline{BR_c^\rho(\boldsymbol{x})}}\Big((1-\varepsilon)\cdot\boldsymbol{x}+\varepsilon\cdot\boldsymbol{x}'\Big)^\top R\boldsymbol{e}_j + \min_{j\in BR_c^\rho(\boldsymbol{x})}\Big((1-\varepsilon)\cdot\boldsymbol{x}+\varepsilon\cdot\boldsymbol{x}'\Big)^\top R\boldsymbol{e}_j$$
$$\le -(1-\varepsilon)\cdot\min_{j\in \overline{BR_c^\rho(\boldsymbol{x})}}\boldsymbol{x}^\top R\boldsymbol{e}_j - \varepsilon\cdot\min_{j\in \overline{BR_c^\rho(\boldsymbol{x})}}(\boldsymbol{x}')^\top R\boldsymbol{e}_j + (1-\varepsilon)\cdot\min_{j\in BR_c^\rho(\boldsymbol{x})}\boldsymbol{x}^\top R\boldsymbol{e}_j$$
$$= -\min_{j\in \overline{BR_c^\rho(\boldsymbol{x})}}\boldsymbol{x}^\top R\boldsymbol{e}_j + \min_{j\in BR_c^\rho(\boldsymbol{x})}\boldsymbol{x}^\top R\boldsymbol{e}_j + \varepsilon\cdot\Big(\min_{j\in \overline{BR_c^\rho(\boldsymbol{x})}}\boldsymbol{x}^\top R\boldsymbol{e}_j - \min_{j\in \overline{BR_c^\rho(\boldsymbol{x})}}(\boldsymbol{x}')^\top R\boldsymbol{e}_j - \min_{j\in BR_c^\rho(\boldsymbol{x})}\boldsymbol{x}^\top R\boldsymbol{e}_j\Big).$$

But, by the definition of ρ-best-response, we have that

$$-\min_{j\in \overline{BR_c^\rho(\boldsymbol{x})}}\boldsymbol{x}^\top R\boldsymbol{e}_j + \min_{j\in BR_c^\rho(\boldsymbol{x})}\boldsymbol{x}^\top R\boldsymbol{e}_j < -\rho,$$

and

$$\min_{j\in \overline{BR_c^\rho(\boldsymbol{x})}}\boldsymbol{x}^\top R\boldsymbol{e}_j - \min_{j\in \overline{BR_c^\rho(\boldsymbol{x})}}(\boldsymbol{x}')^\top R\boldsymbol{e}_j - \min_{j\in BR_c^\rho(\boldsymbol{x})}\boldsymbol{x}^\top R\boldsymbol{e}_j \le 1,$$

since $R_{ij} \le 1$. In total, we have that

$$\min_{j\in BR_c^\rho(\boldsymbol{x})}\boldsymbol{x}^\top R\boldsymbol{e}_j - \min_{j\in \overline{BR_c^\rho(\boldsymbol{x})}}\boldsymbol{x}^\top R\boldsymbol{e}_j + \varepsilon\Big(\min_{j\in \overline{BR_c^\rho(\boldsymbol{x})}}\boldsymbol{x}^\top R\boldsymbol{e}_j - \min_{j\in \overline{BR_c^\rho(\boldsymbol{x})}}(\boldsymbol{x}')^\top R\boldsymbol{e}_j - \min_{j\in BR_c^\rho(\boldsymbol{x})}\boldsymbol{x}^\top R\boldsymbol{e}_j\Big) < -\rho+\varepsilon.$$

Thus, we only need to ensure that $-\rho+\varepsilon \le 0$, which is true when $\varepsilon \le \frac{\rho}{2}$.

B Equivalent Formulation of Algorithm 2

Recall that the linear program involved in Find Direction is the following, given a profile $(\boldsymbol{x},\boldsymbol{y})$:

$$\begin{aligned}
\min\quad & \gamma\\
\text{s.t.}\quad & \gamma \ge (\boldsymbol{e}_i)^\top \boldsymbol{R}\boldsymbol{y}' - (\boldsymbol{x}')^\top \boldsymbol{R}\boldsymbol{e}_j\\
& i\in BR_r^\rho(\boldsymbol{y}), j\in BR_c^\rho(\boldsymbol{x}), \boldsymbol{x}'\in\Delta^{n-1}, \boldsymbol{y}'\in\Delta^{n-1}
\end{aligned}$$

It is easy to see that this is equivalent to:

$$\begin{aligned}
\min\quad & \gamma_1+\gamma_2\\
\text{s.t.}\quad & \gamma_1 \ge (\boldsymbol{e}_i)^\top \boldsymbol{R}\boldsymbol{y}'\\
& \gamma_2 \ge -(\boldsymbol{x}')^\top \boldsymbol{R}\boldsymbol{e}_j\\
& i\in BR_r^\rho(\boldsymbol{y}), j\in BR_c^\rho(\boldsymbol{x}), \boldsymbol{x}'\in\Delta^{n-1}, \boldsymbol{y}'\in\Delta^{n-1}
\end{aligned}$$

Now, we see that the variables of $\boldsymbol{x}'$ appear in separate constraints from the variables of $\boldsymbol{y}'$. Hence, it suffices to solve separately the problems:

$$\begin{aligned}
\min\quad & \gamma_1\\
\text{s.t.}\quad & \gamma_1 \ge (\boldsymbol{e}_i)^\top \boldsymbol{R}\boldsymbol{y}'\\
& i\in BR_r^\rho(\boldsymbol{y}), \boldsymbol{y}'\in\Delta^{n-1}
\end{aligned}
\qquad\qquad
\begin{aligned}
\min\quad & \gamma_2\\
\text{s.t.}\quad & \gamma_2 \ge -(\boldsymbol{x}')^\top \boldsymbol{R}\boldsymbol{e}_j\\
& j\in BR_c^\rho(\boldsymbol{x}), \boldsymbol{x}'\in\Delta^{n-1}
\end{aligned}$$

With this formulation at hand we can now prove Observation 13.

Proof of Observation 13. In the case where $\rho = 1$ we have that $BR_c^\rho(\boldsymbol{x}) = [n]$ and $BR_r^\rho(\boldsymbol{y}) = [n]$. This means that there is a constraint in each LP for every pure strategy, and this means that the LPs reduce to the well known primal and dual LP formulation of computing the equilibrium strategies in zero-sum games. Note that the second LP can be written as a maximization problem (for $-\gamma_2$), which makes the primal-dual connection more apparent. $\square$

C OGDA and Other First Order Methods

The equations for the Optimistic Gradient Descent/Ascent are as follows:

$$\boldsymbol{x}_{t+1} = \boldsymbol{x}_t - 2\alpha \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_t, \boldsymbol{y}_t) + \alpha \nabla_{\boldsymbol{x}} f(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1})$$
$$\boldsymbol{y}_{t+1} = \boldsymbol{y}_t - 2\alpha \nabla_{\boldsymbol{y}} f(\boldsymbol{x}_t, \boldsymbol{y}_t) + \alpha \nabla_{\boldsymbol{y}} f(\boldsymbol{x}_{t-1}, \boldsymbol{y}_{t-1})$$

where for bilinear objective functions, such as zero-sum games we have $f(\boldsymbol{x}, \boldsymbol{y}) = x^\top \boldsymbol{R} y$.

To serve as a means of comparison to our method, we implemented the OGDA algorithm, following the aforementioned equations. To verify the validity of our implementation, we present a comparison with the corresponding PyOL(Python library for Online Learning) version of the same algorithm. In the figure that follows, the algorithms are set with $\eta = 0.01$ and are tested on uniformly random games.

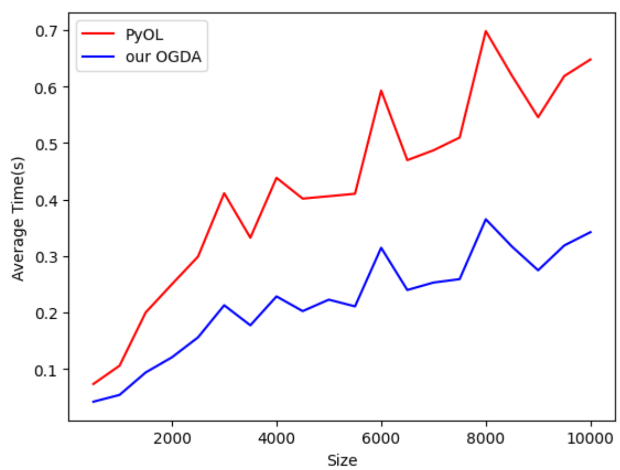

Figure 4: Time comparison between PyOL and our OGDA implementations

There exist also other first order methods, which have good theoretical guarantees. A popular such method is Optimistic Multiplicative Weights Update (OMWU), an optimistic variant of MWU. We realized however that MWU is less stable than OGDA and exhibits cyclic behavior in certain cases. In fact this has been also observed by other works and the recent paper by Cai et al. [2024] provides further justification on why some first order methods may fail to be a good practical solution.

As a conclusion, we decided to present comparisons of our method only against OGDA in Section 4.

D Additional Experiments

Here we present additional experiments regarding the choice of parameters and more comparisons between our variants and other methods. All the experiments here and in Section 4 were run on a Macbook M1 Pro(10 core) with 16GB RAM. We developed our code in Python 3.10.9, using the packages NumPy 2.0.2 and SciPy 1.14.1.

D.1 Comparison with Constant Step Size ϵ

We begin our experiments with the comparison between our Fixed Support Variant (with optimized ε) and the Fixed Support Variant with a constant ε, along the spirit of standard first-order methods using a constant learning rate. We present two comparisons in Figure 5 for a random game of a size $n = 5000$. In the left subplot we see the result of starting from $(\boldsymbol{e}_1, \boldsymbol{e}_1)$. As expected, the optimized version here is much faster, since when the duality gap is large there is a room for big steps. In addition, we also see (in the right subplot) that even for the uniform initialization, i.e. when we are close to an approximate Nash equilibrium, that the optimized version terminates faster. Given that the iterations are more expensive, we should compare the times as well. Indeed, the constant step size requires 1.8 seconds vs 1.15 for the optimized version.

Therefore, we selected to use our optimized version for the choice of ϵ as described in Section 4.1.

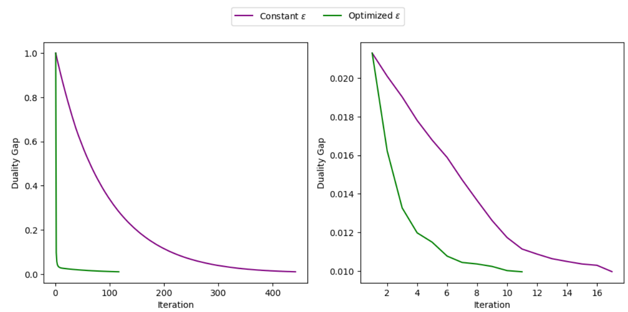

Figure 5: Iteration comparison between for the initializations $(\boldsymbol{e}_1, \boldsymbol{e}_1)$ (left) and the uniform point (right)

D.2 Tuning our Fixed Support Variant

For the use of our Fixed Support variant, recall that we used a parameter k to denote the number of the top better responses that we select to include when constructing the LP constraints in Algorithm 2. We tried to experiment with the appropriate size for k (also referred to as the support size). We present in Figure 6 the averaged results for 20 games of size $n = 10000$. As the figure reveals, the range of k between $[60, 130]$ seemed to behave quite well, and as a result, we selected $k = 100$. We would like to point out that the image is similar for other game sizes.

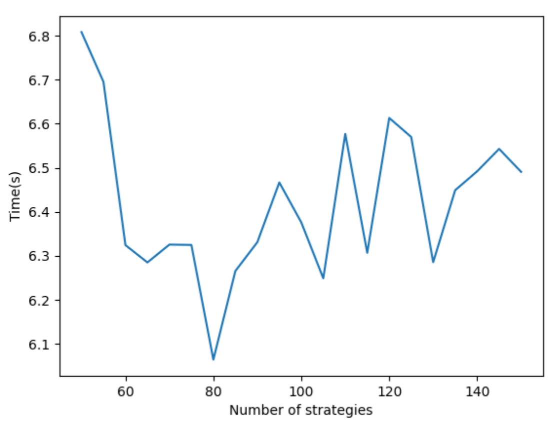

Figure 6: Running time comparison for different values of k

We propose, as an interesting direction for future work, some possible modifications. One is to set k as a function of n, i.e. $k = \log n$. Having the support size growing with n is natural and we cannot rule out the possibility that for even larger games sizes, with size in the millions, $k = 100$ would be too small to achieve high performance. Another approach, quite similar to one approach with ρ, would be to use an adaptive support size. In this case, the adaptation could be over the step size ε; whenever the step becomes too small, we can increase k and try for larger steps in the next iterations.

D.3 Gaussian Random Games

Our next experiment concerns a different class of randomly generated games: instead of sampling from the uniform distribution we sample from the standard Gaussian, and then rescale to have the entries back in $[0, 1]$. Qualitatively, we observed no difference between our methods; the Fixed Support variant, with the same parameters as in the uniform game generation, was the winner among all our variants, as before. Therefore, here we only show a comparison between our Fixed Support variant and OGDA, in Figure 7. Our findings are consistent with what we presented in Section 4. Additionally here we notice that both methods are faster, compared to the times with the uniform distribution. Our method matches and outperforms OGDA at an earlier range of game sizes (at around $n = 5000$ as opposed to $n = 18000$ for the uniformly random games).

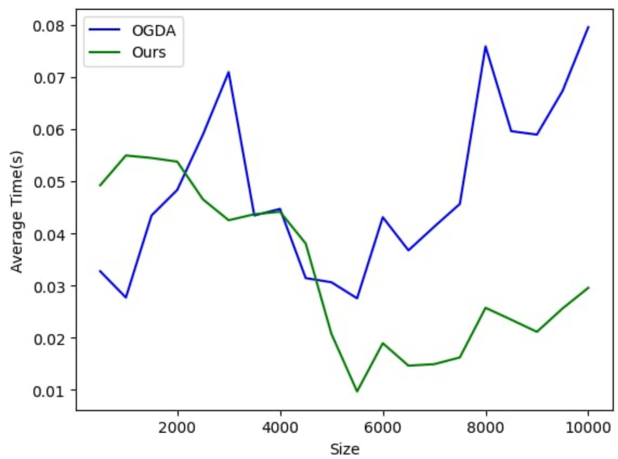

Figure 7: Comparison with Gaussian Random Games

D.4 Low Rank Zero-sum Games

We also experimented with a class of games where the payoff entries are not all drawn independently from each other, which translates to a lower than full rank payoff matrix. To generate a matrix $\boldsymbol{R}$ of size n and fixed rank r, we uniformly sample matrices $\boldsymbol{U}, \boldsymbol{V}$ of dimensions $n \times r$. Obtaining $\boldsymbol{R}$ via $\boldsymbol{R} = \boldsymbol{U} \cdot \boldsymbol{V}^\top$ gives us a matrix of desired rank r.

This class of games provided new insights for our method. First, we noticed that both our Fixed Support Variant and OGDA were getting stuck and/or significantly slowed down to lower accuracies. To amend this issue for OGDA, we experimented with different values of constant learning rates but this was to no avail. Then, we turned to the square rooted learning rate option of PyOL library which helped accelerate OGDA to some degree. On the other hand, for our method there was a parameter we had not paid much attention to earlier, that needed tweaking: the tolerance parameter of the LP solver in Algorithm 2. Once using a lower tolerance, i.e. asking for higher accuracy solutions to our small LPs the speed of our method improves drastically! The result for matrices of rank 10 is depicted in Figure 8 with the picture being similar for higher ranks as well. In particular, we observe that we outperform OGDA as soon as the game size exceed 1000 strategies.

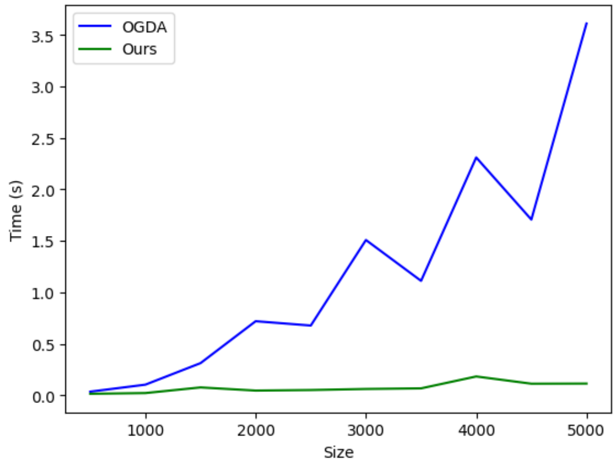

Figure 8: Comparison for fixed rank $r = 10$