ApproxED: Approximate Exploitability Descent via Learned Best Responses

Carlos Martin
Carnegie Mellon University
Pittsburgh, United States
cgmartin@cs.cmu.edu

Tuomas Sandholm
Carnegie Mellon University
Strategy Robot, Inc.
Optimized Markets, Inc.
Strategic Machine, Inc.
Pittsburgh, United States
sandholm@cs.cmu.edu

ABSTRACT

There has been substantial progress on finding game-theoretic equilibria. Most of that work has focused on games with finite, discrete action spaces. However, many games involving space, time, money, and other fine-grained quantities have continuous action spaces (or are best modeled as having such). We study the problem of finding an approximate Nash equilibrium of games with continuous action sets. The standard measure of closeness to Nash equilibrium is exploitability, which measures how much players can benefit from unilaterally changing their strategy. We propose two new methods that minimize an approximation of exploitability with respect to the strategy profile. The first method uses a learned best-response function, which takes the current strategy profile as input and outputs candidate best responses for each player. The strategy profile and best-response functions are trained simultaneously, with the former trying to minimize exploitability while the latter tries to maximize it. The second method maintains an ensemble of candidate best responses for each player. In each iteration, the best-performing elements of each ensemble are used to update the current strategy profile. The strategy profile and ensembles are simultaneously trained to minimize and maximize the approximate exploitability, respectively. We evaluate our methods on various continuous games and GAN training, showing that they outperform prior methods.

KEYWORDS

Game theory; equilibrium finding; continuous games; game solving

ACM Reference Format:
Carlos Martin and Tuomas Sandholm. 2025. ApproxED: Approximate Exploitability Descent via Learned Best Responses. In *Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Detroit, Michigan, USA, May 19 – 23, 2025*, IFAAMAS, 10 pages.

1 INTRODUCTION

Most work concerning equilibrium computation has focused on games with finite, discrete action spaces. However, many games involving space, time, money, *etc.* have continuous action spaces. Examples include continuous resource allocation games [25], security games in continuous spaces [46–48], network games [28], military simulations and wargames [64], and video games [5, 85]. Moreover, even if the action space is discrete, it can be fine-grained enough to treat as continuous for the purpose of computational efficiency [7, 11, 26]. Although continuous action spaces can be discretized[1], this entails a loss in solution quality. Discretization is especially problematic in high-dimensional spaces. For example, in GANs [32], the strategy space is a probability distribution over images generated by a neural network.

As the goal, we use the standard solution concept of a *Nash equilibrium (NE)*, that is, a strategy profile for which each strategy is a best response to the other players' strategies. The main measure of closeness to NE is *exploitability*, which measures how much players can benefit from unilaterally changing their strategy. Typically, we seek an NE, that is, a strategy profile for which exploitability is zero. We can try to search for NE by performing gradient descent on exploitability, since it is non-negative and zero precisely at NE. However, computing exploitability requires computing best responses to the current strategy profile, which is itself a nontrivial problem in many games. We present two new methods that minimize an approximation of the exploitability with respect to the strategy profile. Our experiments on various continuous games show that our techniques outperform prior approaches.

2 PROBLEM FORMULATION

We use the following notation. Hereafter, $\triangle\mathcal{X}$ is the set of probability distributions on a space $\mathcal{X}$, $[n] = \{0, \dots, n-1\}$ is the set of natural numbers less than a natural number $n \in \mathbb{N}$, $|\mathcal{X}|$ is the cardinality of a set $\mathcal{X}$, and $\mathbb{S}_n$ is the unit n-sphere. For any logical formula ϕ, $[\![\phi]\!]$ is an Iverson bracket, which is 1 if ϕ is true and 0 otherwise.

A *game* is a tuple $(\mathcal{I}, \mathcal{X}, u)$ where $\mathcal{I}$ is a set of players, $\mathcal{X}_i$ is a strategy set for player i, and $u_i : \prod_i \mathcal{X}_i \to \mathbb{R}$ is a utility function for player i. A strategy profile $x \in \prod_i \mathcal{X}_i$ is an assignment of a strategy to each player. A game is zero-sum if $\sum_{i \in \mathcal{I}} u_i = 0$. Given a strategy profile x, Player i's regret is $R_i(x) = \sup_{y_i \in \mathcal{X}_i} u_i(y_i, x_{-i}) - u_i(x)$, where x_{-i} denotes the other players' strategies. It is the highest utility Player i could gain from unilaterally changing its strategy. A strategy profile x is an ε-equilibrium if $\sup_{i \in \mathcal{I}} R_i(x) \leq \varepsilon$. A 0-equilibrium is called a *Nash equilibrium (NE)*. In an NE, each player's strategy is a best response to the other players' strategies, that is, $u_i(x) \geq u_i(y_i, x_{-i})$ for all $i \in \mathcal{I}$ and $y_i \in \mathcal{X}_i$.

[1] Kroer and Sandholm [54] provide bounds on solution quality for discretization of continuous action spaces in extensive-form games.

Proc. of the 24th International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2025), Y. Vorobeychik, S. Das, A. Nowé (eds.), May 19 – 23, 2025, Detroit, Michigan, USA.

Table 1: ODE corresponding to each method.

Method	$\dot{x}$
SG	v
EG	$v\|_{x+\gamma v}$
OP	$(I+\gamma \frac{\mathrm{d}}{\mathrm{d}t})v = v + \gamma \dot{v}$
CO	$(I-\gamma J^\top)v = v - \gamma \nabla \frac{1}{2}\|v\|^2$
SGA	$(I-\gamma J_a^\top)v$
SLA	$(I+\gamma J)v$
LA	$(I+\gamma J_o)v$
LOLA	$(I+\gamma J_o)v - \gamma \operatorname{diag} J_o^\top \nabla u$
LSS	$(I+J^\top J^{-1})v$
PCGD	$(I-\gamma J_o)^{-1}v$
ED	$-\nabla_x \sup_y \phi(x,y) = -\nabla_x \Phi(x)$
GNI	$-\nabla_x \phi(x, x+\gamma v)$

The standard measure of closeness to NE is exploitability, also known as NashConv [56, 60, 82, 87]. It is defined as $\Phi = \sum_{i\in\mathcal{I}} R_i$. (In a two-player zero-sum game, Φ reduces to the so-called *duality gap* [33].) It is non-negative everywhere and zero precisely at NE. Thus finding an NE is equivalent to minimizing exploitability [60]. Let $\phi(x,y) = \sum_{i\in\mathcal{I}}(u_i(y_i, x_{-i}) - u_i(x))$ be the *Nikaido-Isoda (NI)* function [22, 23, 42, 69]. Since $\Phi(x) = \sup_y \phi(x,y)$, finding an NE is equivalent to solving the min-max problem $\inf_x \sup_y \phi(x,y)$. Some prior work has used this function to search for NE [6, 23, 34, 38, 39, 42, 52, 53, 72, 73, 83, 84].

3 RELATED RESEARCH

In this section we review the prior methods for solving continuous games, which we use as baselines in our experiments. They can be characterized by the *ordinary differential equations (ODEs)* shown in Table 1. Here, $v = \operatorname{diag} \nabla u$ is the *simultaneous gradient*. Each component $v_i = \nabla_i u_i$ is the gradient of a player's utility with respect to their strategy. Additionally, $J = \nabla v$ is the Jacobian of the vector field v, $J^\top$ is its transpose, $J_a = \frac{1}{2}(J - J^\top)$ is its antisymmetric part, and J_o is its off-diagonal part (replacing its diagonal with zeroes). Dots indicate derivatives with respect to time, $\gamma > 0$ is a hyperparameter, and $v|_y$ denotes v evaluated at y rather than x.

The simultnaeous gradient yields a vector field on the space of strategy profiles. The fact that this vector field may not be conservative (that is, the gradient of some potential), like it is in gradient descent, is the main source of difficulties for standard gradient-based optimization methods, since trajectories can cycle around fixed points rather than converging to them. The actual optimization is done by discretizing each ODE in time. For example, SG is discretized as $x_{i+1} = x_i + \eta v_i$ and OP is discretized as $x_{i+1} = x_i + \eta v_i + \gamma(v_i - v_{i-1})$, where $\eta > 0$ is a stepsize and $v_j = v(x_j)$.

Simultaneous gradients (SG) maximizes each player's utility independently, as if the other players are fixed. *Extragradient (EG)* [50] takes a step in the direction of the simultaneous gradient and uses the simultaneous gradient at that new point to take a step from the original point. Golowich et al. [31] proved a tight last-iterate convergence guarantee for EG. *Optimistic gradient (OP)* [15, 43, 71] uses past gradients to predict future gradients and update according to the latter.

Consensus optimization (CO) [66] penalizes the magnitude of the simultaneous gradient, encouraging "consensus" between players that attracts them to fixed points. *Symplectic gradient adjustment (SGA)* [2] (also known as Crossing-the-Curl [27]) reduces the rotational component of game dynamics by using the antisymmetric part of the Jacobian. *Lookahead (LA)* [89] excludes the diagonal components of the Jacobian. Each player predicts the behaviour of other players after a step of naive learning, but assumes this step will occur independently of the current optimisation. In *symmetric lookahead (SLA)* [59], instead of best-responding to opponents' learning, each player responds to *all* players learning, including themselves. It is a linearized version of EG [19, Lemma 1.35].

In *learning with opponent-learning awareness (LOLA)* [24], a learner optimises its utility under one step look-ahead of opponent learning. Instead of optimizing utility under the current parameters, it optimises utility after the opponent updates its policy with one naive learning step.

Mazumdar et al. [65] proposed *local symplectic surgery (LSS)* to find local NE in two-player zero-sum games. It solves a linear system on each timestep, which is prohibitive for high-dimensional parameter spaces. Hence, its authors propose a two-timescale approximation that updates the strategy profile while simultaneously improving an approximate solution to the linear system.

Competitive gradient descent (CGD) [76] naturally generalizes gradient descent to the two-player setting. On each iteration, it jumps to the NE of a quadratically-regularized bilinear local approximation of the game. Its convergence and stability properties are robust to strong interactions between the players without adapting the stepsize. *Polymatrix competitive gradient descent (PCGD)* [62] generalizes CGD to more than two players. It jumps to the NE of a quadratically-regularized local polymatrix approximation of the game. CGD and PGCD require solving a linear system of equations on each iteration, which is prohibitive for high-dimensional parameter spaces [62, p. 10].

Exploitability descent (ED) [60] directly minimizes exploitability, and converges to approximate equilibria in two-player zero-sum extensive-form games. However, it requires computing best responses y on each iteration, which is inefficient and/or intractable in general games. *Gradient-based Nikaido-Isoda (GNI)* [73] minimizes a local approximation of exploitability that uses local best responses $y = x + \gamma v$.

Goktas and Greenwald [30] recast the exploitability-minimization problem as a min-max optimization problem and obtain polynomial-time first-order methods for computing variational equilibria in convex-concave cumulative regret pseudo-games with jointly convex constraints. They present two algorithms called *extragradient descent ascent (EDA)* and *augmented descent ascent (ADA)*. We benchmark against EDA but not ADA because, unlike the other baselines, it requires *multiple* substeps of gradient ascent *per timestep* to approximate a best response. (As the authors note, "ADA's accuracy depends on the accuracy of the best-response found".)

Fiez et al. [21] study learning in Stackelberg games, establishing connections between Nash and Stackelberg equilibria along with the limit points of simultaneous gradient descent. They gradient-based learning dynamics emulating the natural structure of a Stackelberg game using the implicit function theorem, and provide convergence analysis for deterministic and stochastic updates for zero-sum and general-sum games. However, in this paper, we are interested in finding *Nash* equilibria, not Stackelberg equilibria.

Wellman et al. [88] survey *empirical game-theoretic analysis (EGTA)*, which uses simulation to generate data from which one can induce a game model, called the empirical game. As described in §3.2.2, the machine learning approach to empirical game modeling is a form of regression where the input is a set of (profile, payoff-vector) pairs, and the output is the vector of empirical utility functions. These techniques can be used to infer a complete empirical game model from an incomplete one.

4 PROPOSED METHODS

We are given a utility function u[2] and our goal is to find an NE. Since the exploitability $\Phi(x) = \sup_y \phi(x, y)$ is non-negative everywhere, and zero exactly at NE, we reformulate the problem of finding an NE as *finding a global minimum of the exploitability function.* That is, we wish to solve the min-max optimization problem $\inf_x \sup_y \phi(x, y)$. This is equivalent to finding a minimally-exploitable strategy for a two-player zero-sum meta-game with utility function ϕ. To find a minimum, we could try performing gradient descent on Φ, like ED does. However, ED requires best-response oracles. In general games, exact best responses can be difficult or intractable to compute. Thus we need alternative solutions.

To solve this problem, we can try to perform gradient descent on x and ascent on y simultaneously: $\dot{x} = -\nabla_x \phi(x, y)$, $\dot{y} = \nabla_y \phi(x, y)$. Unfortunately, this approach can fail even in simple games. For example, consider the simple bilinear game with $u(x, y) = xy$. The unique Nash equilibrium is at the origin. However, simultaneous gradient descent fails to converge to it, and instead cycles around it indefinitely. The essence of this cycling problem is that Player 2 has to “relearn” a good response to Player 1 every time the Player 1’s strategy switches sign. This is a general problem for games. “Small” changes in other players’ strategies can cause “large” (discontinuous) changes in a player’s best response. When such changes occur, players have to “relearn” how to respond to the other players’ strategies. We propose two methods to tackle this problem. These are described in the next two subsections, respectively.

4.1 Best-response functions

For this method, we conceptually reformulate the problem as

$$\operatorname*{argmin}_{x \in \mathcal{X}} \sup_{y \in \mathcal{Y}} \phi(x, y) = \operatorname*{argmin}_{x \in \mathcal{X}} \phi(x, b(x))$$

where $b : \mathcal{X} \to \mathcal{Y}$ is a function that satisfies $b(x) \in \operatorname{argmax}_{y \in \mathcal{Y}} \phi(x, y)$. Since b is a *function*, it can map different strategies for Player 1 to different strategies for Player 2. Thus it can, at least in principle, *immediately* adapt to Player 1’s strategy x, without forgetting prior solutions, and thus avoid the cycling problem.

More precisely, suppose $\mathcal{Y}$ is compact and $\phi : \mathcal{X} \times \mathcal{Y} \to \mathbb{R}$ is continuous in its second argument. Let $x \in \mathcal{X}$. By the extreme value theorem, a continuous real-valued function on a non-empty compact set attains its extrema. Therefore, there exists $y \in \mathcal{Y}$ such that $\phi(x, y) = \sup_{y \in \mathcal{Y}} \phi(x, y)$. Since this is true for every $x \in \mathcal{X}$, there exists a function $b : \mathcal{X} \to \mathcal{Y}$ such that, for every $x \in \mathcal{X}$, $\phi(x, b(x)) = \sup_{y \in \mathcal{Y}} \phi(x, y)$. In other words, b is a *best-response function.* Even when $\mathcal{Y}$ is not compact and ϕ does not attain its extrema, one can define a best-response *value* for any $x \in \mathcal{X}$ as $\sup_{y \in \mathcal{X}} \phi(x, y)$, provided the latter exists. In that case, we have the following. Let $\varepsilon > 0$ and $x \in \mathcal{X}$. Any function gets arbitrarily close to its supremum (continuity is not required). Therefore, there exists a $y \in \mathcal{Y}$ such that $\phi(x, y) + \varepsilon \geq \sup_{y \in \mathcal{Y}} \phi(x, y)$. Therefore, there exists a function $b_\varepsilon : \mathcal{X} \to \mathcal{Y}$ such that, for every $x \in \mathcal{X}$, $\phi(x, b_\varepsilon(x)) + \varepsilon \geq \sup_{y \in \mathcal{Y}} \phi(x, y)$. That is b_ε is an *ε-approximate* best-response function.

To find x and b simultaneously, we can perform simultaneous gradient ascent: $\dot{x} = -\nabla_x \phi(x, b(x))$, $\dot{b} = +\nabla_b \phi(x, b(x))$, where $\nabla_x \phi(x, b(x))$ is a total (not partial) derivative. That is, the best response function tries to *increase* the exploitability while the strategy profile tries to *decrease* it. Since b is a function, Player 1’s changing behavior poses no fundamental hindrance to it learning good responses and “saving” them for later use if Player 1’s behavior changes. It could even learn a good approximation to the true best-response function, leaving Player 1 to face a simple standard optimization problem.

If $\mathcal{X}$ is infinite and $\mathcal{Y}$ is nontrivial, $\mathcal{X} \to \mathcal{Y}$ has infinite dimension. To represent and optimize b in practice, we need a finite-dimensional parameterization of (a subset of) this function space. In particular, if b is parameterized by w and is (approximately) surjective onto $\mathcal{Y}$, then $\Phi(x) = \sup_{y \in \mathcal{Y}} \phi(x, y) \approx \sup_{w \in \mathcal{W}} \phi(x, b(w, x))$. Inspired by this idea, we propose jointly optimizing x and w according to the following ODE system.

$$\dot{x} = -\nabla_x \phi(x, b(w, x)) \qquad \dot{w} = +\nabla_w \phi(x, b(w, x)) \tag{1}$$

We call our method *approximate exploitability descent with learned best-response functions (ApproxED-BRF).* Its structure is depicted in Figure 1. As the figure illustrates, it maps a *strategy profile* to a profile of *best responses* for each player to the other players. It is an all-to-all function. We emphasize that this cross-player propagation of information occurs only for the purpose of *finding approximate best responses* in order to train *the strategy parameters*, and does not mean, for example, that players now get to observe each other’s actions within the real game itself.

The best-response function can take on many possible forms. One possibility is to use a neural network. Neural networks are universal function approximators and have a powerful ability to generalize well across inputs [14, 40, 41, 57, 70]. Neural networks

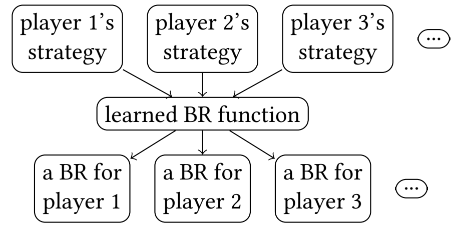

Figure 1: Structure of the BRF, in the case of 3 players.

[2]For exposition, we assume utility functions are differentiable. If they are not, we can replace gradients with *pseudo*gradients [3, 18, 67, 68, 75, 78].

are also, by far, the most popular function approximators used in game solving. Therefore, we use this approach in our experiments.

On the other hand, we also experiment with settings where each player's strategy is *itself* is represented by a neural network. In this case, the best-response functions take those networks' parameters as input. In other words, we adapt the concept of *hypernetworks* [1, 35, 61, 63, 77] to the game-theoretic context. We note that, to obtain good performance, the learned best response function does not need to represent the true best response function (which may be discontinuous) *exactly*, but only *approximately*. Our experimental results indicate that the approximation yielded by the neural network performs well across a wide class of games.

4.2 Best-response ensembles

For this method, we conceptually reformulate the problem as

$$\operatorname*{argmin}_{x \in \mathcal{X}} \sup_{y \in \mathcal{Y}} \phi(x, y) \approx \operatorname*{argmin}_{x \in \mathcal{X}} \max_{j \in \mathcal{J}} \phi(x, y_j)$$

where $\mathcal{J}$ is a finite set of indices, and $x \in \mathcal{X}$ and $y : \mathcal{J} \to \mathcal{Y}$ are trainable parameters. In other words, we use an *ensemble* of $|\mathcal{J}|$ responses to x, where the *best* response is selected automatically by evaluating x against each y_j and taking the one that attains the best value. Each individual y_j is a *strategy* for the original game. Since there are multiple responses in the ensemble, each one can "focus on" tackling a particular "type" of behavior from x without having to change drastically when the latter changes. We can then train x and y simultaneously: $\dot{x} = -\nabla_x \max_{j \in \mathcal{J}} \phi(x, y_j)$, $\dot{y} = \nabla_y \max_{j \in \mathcal{J}} \phi(x, y_j)$. That is, x improves against the best y_j, while the best y_j improves against x. Ties are broken in indexical lower (lower indices first). This allows for symmetry breaking if the ensemble elements are initially equal and the game is deterministic.

There is an issue with the aforementioned scheme, however. If one of the ensemble elements y_j strictly dominates the others for all encountered x, then the other elements will never be selected under the maximum operator. Thus they will never have a chance to change, improve performance, and thus contribute. In that case, the scheme degenerates to ordinary simultaneous gradient ascent. We observed this degeneracy in some games.

To solve this issue, we introduced the following approach. To give *all* y_j some chance to improve, while incentivizing them to "focus" on particular types of x rather than cover all cases, we use a *rank-based weighting* approach. Specifically, we let $\dot{y} = \nabla_y \operatorname{mix}_{j \in \mathcal{J}} \phi(x, y_j)$ where $\operatorname{mix}_{j \in \mathcal{J}} a_j = \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} r_j a_j$ and $r_j \in \{1, \ldots, |\mathcal{J}|\}$ is the ordinal rank of element j. This makes better elements receive a higher weight. Thus the best y_j has the most incentive to adapt against the current x, while others have less incentive, but still some nonetheless.[3]

Our method is defined by the following ODE system.

$$\dot{x} = -\nabla_x \sum_{i \in \mathcal{I}} \left(\max_{j \in \mathcal{J}} u_i(y_{ij}, x_{-i}) - u_i(x) \right) \tag{2}$$

$$\dot{y} = +\nabla_y \sum_{i \in \mathcal{I}} \left(\operatorname*{mix}_{j \in \mathcal{J}} u_i(y_{ij}, x_{-i}) - u_i(x) \right) \tag{3}$$

[3]Furthermore, since the weight of each ensemble element depends only on the rank or order of values, it is invariant under monotone transformations of the utility function.

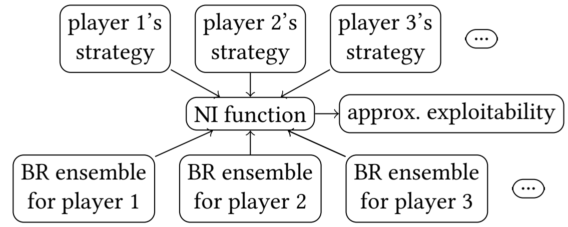

Figure 2: Structure of the BRE, in the case of 3 players.

Here, $y = \{y_{ij}\}_{i \in \mathcal{I}, j \in \mathcal{J}}$ is an ensemble of $|\mathcal{J}|$ responses for each individual player $i \in \mathcal{I}$. We call our method *approximate exploitability descent with learned best-response ensembles (ApproxED-BRE)*. Its structure is depicted in Figure 2.

One can compute the $u_i(y_{ij}, x_{-i})$ and their gradients in parallel for $i \in \mathcal{I}$, $j \in \mathcal{J}$. Hence, with access to $O(|\mathcal{I}||\mathcal{J}|)$ cores, the method can run approximately as fast as standard simultaneous gradient ascent. The ensemble size can be as big as the amount of memory and number of cores or workers available allows. If a practitioner has access to parallel computing infrastructure, they can scale up the ensemble size as much as possible, until all of the available parallelism is used up, thus reaping the benefits of more coverage of the response space while incurring little to no penalty in wall time. An interesting direction for future research would be to analyze the effect of the ensemble size.[4]

5 EXPERIMENTS

In this section, we present our experiments. We use a learning rate of $\eta = 10^{-3}$ and $\gamma = 10^{-1}$. For BRF's best-response function, we use a fully-connected network with a hidden layer of size 32, the tanh activation function, and He weight initialization [36]. We do not try to find the best neural architecture, because this problem comprises an entire field, may be task-specific, and is not the focus of our paper. Thus our experiments are conservative, in the sense that our technique could perform even better compared to the baselines if engineering effort were spent tuning the neural network. For BRE, we use ensembles of size 10 for each player. For each experiment, we ran 64 trials. In our plots, solid lines show the mean across trials, and bands show its standard error. For games with stochastic utility functions, we used a batch size of 64. Each trial was run on one NVIDIA A100 SXM4 40GB GPU. Our code uses Python 3.12.2, jax 0.4.28 [8], flax 0.8.3 [37], optax 0.2.2 [16], and matplotlib 3.8.4 [44].

Some of the benchmarks are based on normal-form or extensive-form games with finite action sets, and thus finite-dimensional continuous mixed strategies. While there are algorithms for such games that might have better performance (such as counterfactual regret minimization [90] and its fastest new variants [10, 20]), these do not readily generalize to general continuous-action games. Thus we are interested in comparing only to those algorithms which, like ours, *do* generalize to continuous-action games, namely those described in §3.

[4]For clarity, we emphasize that each individual strategy (including each individual element of an ensemble) can, in general, represent a *mixed* strategy of the original game. Thus the support size of the NE (in terms of *pure strategies* of the original game) is not necessarily directly related to the ensemble size.

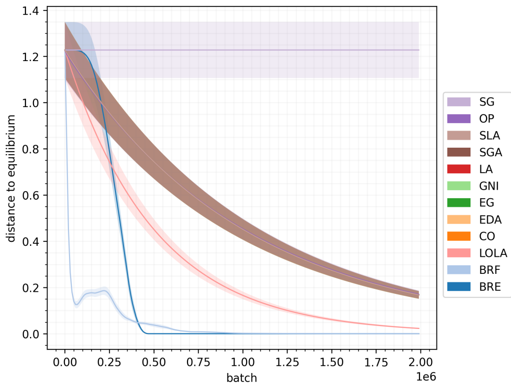

Figure 3: Saddle-point game distances.

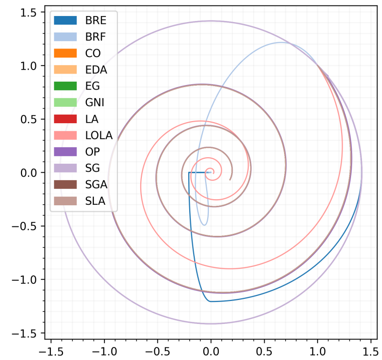

Figure 4: Saddle-point game trajectories.

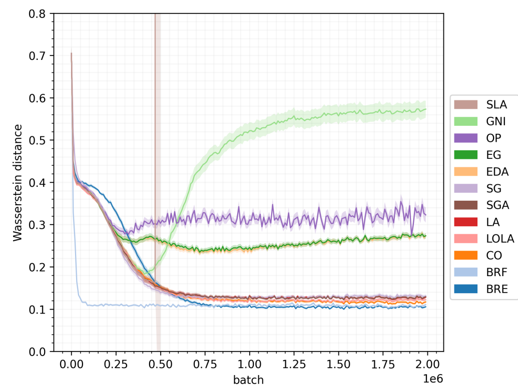

Figure 5: GAN on ring dataset.

We parameterize mixed strategies on finite action sets (*e.g.*, for normal-form games, or at a particular information set inside an extensive-form game) using logits. The action probabilities are obtained by applying the softmax function, which ensures they are non-negative and sum to 1. The utilities are the expected utilities under the resulting mixed strategies. Therefore, our games games are continuous in the *nonlinear* and *high-dimensional* space of mixed behavioral strategies parameterized by logits, which is the strategy space we optimize over.

In games where a player must randomize over a *continuous* action set, we use an implicit density model, also called a deep generative model [74]. It samples noise from some base distribution, such as a multivariate standard normal distribution, and feeds it to a neural network, inducing a transformed probability distribution on the output space. Unlike an explicit parametric distribution on the output space, it can flexibly model a wide class of distributions. A *generative adversarial network (GAN)* [32] is one example of an implicit density model.

We emphasize that it is not our goal to match or exceed the state of the art on a particular game. Rather, our goal is to present a method that can tackle *in full generality* the problem described in §2. We now describe each game and our results for it.

Saddle-point game. For illustration, we start with a simple game. This is a two-player zero-sum game with actions that are real numbers and utility function $u_1(x, y) = -u_2(x, y) = xy$. It has a unique NE at the origin. Results are shown in Figure 3. Our methods converge fastest. Strategy trajectories are shown in Figure 4. Our methods take a more direct path to the equilibrium.

GAN training. A *generative adversarial network (GAN)* [32] is a generative model that consists of two neural networks: a generator and discriminator. The generator maps latent noise to a data sample. The discriminator maps a data sample to a probability. The generator learns to generate fake data, while the discriminator learns to distinguish it from real data. GAN training is a very high-dimensional problem with a highly nonlinear utility function, since the strategies are parameters for the generator and discriminator.

We test the equilibrium-finding methods on the following datasets. The *ring* dataset consists of a mixture of 8 Gaussians with a standard deviation of 0.1 whose means are equally spaced around a circle of radius 1. The *grid* dataset consists of a mixture of 9 Gaussians with a standard deviation of 0.1 whose means are laid out in a regular square grid spanning from −1 to +1 in each coordinate. The *spiral* dataset consists of a noisy Archimedean spiral, where $t \sim \mathcal{U}(0, 1)$, $r = \sqrt{t}$, $\theta = 2\pi rn$, $x = \mathcal{N}(r\cos\theta, \sigma)$, and $y = \mathcal{N}(r\sin\theta, \sigma)$. Here, n is the number of turns (we use 2) and σ is the standard deviation of the noise (we use 0.05). Finally, the *cube* dataset consists of points sampled uniformly from the edges of a cube and perturbed with Gaussian noise of scale 0.05.

In all cases, the generator's latent noise distribution is a standard Gaussian matching the dimension of the dataset. The generator and discriminator have hidden layers of size 32. We evaluate our method using *Wasserstein distance (WD)* between the real data distribution and the generator's fake data distribution. It is the minimum transportation cost needed to turn one distribution into another. We estimate the WD between the real distribution and generator distribution by taking 1000 samples of each, computing the Euclidean distance matrix, and solving the resulting linear assignment problem.[5] Results are shown in Figures 5, 6, 7, and 8. Our methods outperform the rest. In all cases, SLA diverged to infinity.

[5]For the linear assignment problem, we use the implementation found in the Python scientific computing library SciPy [86], which uses a modified Jonker-Volgenant algorithm with no initialization [13].

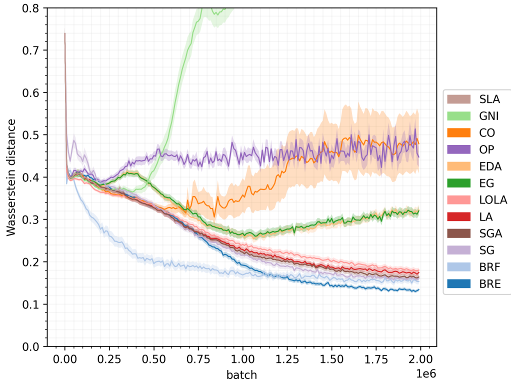

Figure 6: GAN on grid dataset.

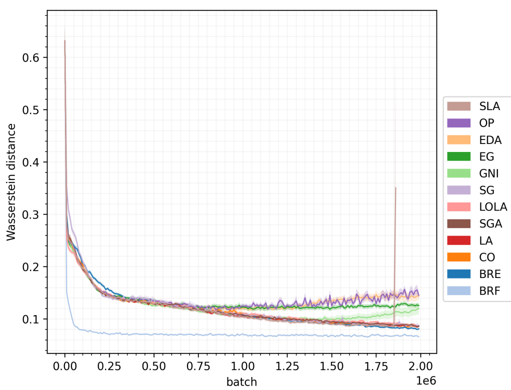

Figure 7: GAN on spiral dataset.

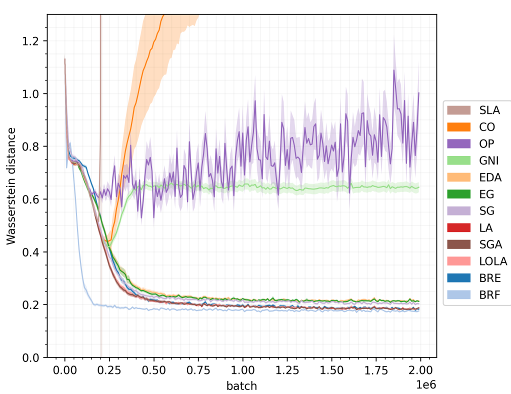

Figure 8: GAN on cube dataset.

We also test on MNIST [17], a dataset of 70,000 28×28 grayscale images of handwritten digits from 0 to 9. The generator and discriminator networks are the same as before, and fully connected, but with the hidden layer size increased to 256 and the noise dimension increased to 32. Due to the larger network size, we use a smaller learning rate of 10^{-4}. Samples are shown in Figure 9.

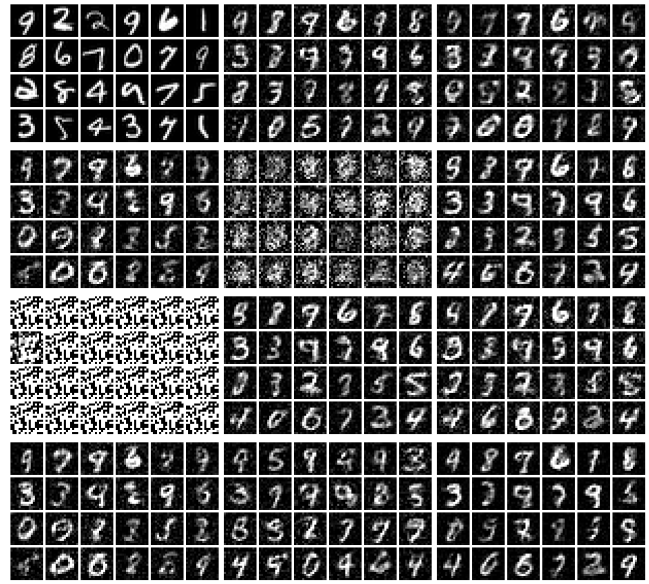

Figure 9: GAN (MNIST dataset). Left to right, top to bottom: Ground truth, SG, OP, EG, CO, SGA, GNI, LA, LOLA, EDA, BRF, BRE. SLA is excluded due to divergence.

Continuous security game. Security games are used to model defender-adversary interactions in many domains, such as the protection of infrastructure like airports, ports, and flights [47], as well as wildlife, fisheries, and forests [49, 80]. Security games are often modeled with Stackelberg equilibrium as the solution concept, which coincides with NE in zero-sum security games and certain structured general-sum games [51]. Many security games have continuous action spaces. These have been studied by Kamra et al. [46], Kamra et al. [47], and Kamra et al. [48]. Consider the following game. Let $\mathcal{S} = [0, 1]^2$. The attacker chooses a point $x \in \mathcal{S}$. Simultaneously, the defender chooses n points $y_i \in \mathcal{S}$. Let $d = \inf_{i \in [n]} \|x - y_i\|$ be the distance between the attacker's point and the defender's closest point. The defender receives a utility of $\exp(-d^2)$, and the attacker receives $-\exp(-d^2)$. Thus the defender seeks to be close to the attacker, while the opposite is true for the attacker. Results are shown in Figures 10 and 11. Our methods perform best.

Glicksberg–Gross game. This is a two-player zero-sum normal-form game with continuous action sets $\mathcal{A}_i = [0, 1]$ and utility function $u_1(x, y) = -u_2(x, y) = \frac{(1+x)(1+y)(1-xy)}{(1+xy)^2}$. Glicksberg and Gross [29] analyzed this game and proved that it has a unique mixed-strategy NE where each player's strategy has a cumulative distribution function of $F(t) = \frac{4}{\pi} \arctan \sqrt{t}$. To model mixed strategies, we use the following implicit density model. We feed a sample from a 1-dimensional standard normal distribution into a fully-connected network with one hidden layer of size 32 and output layer of size 1. The output is squeezed to the unit interval using the logistic sigmoid function. Results are shown in figure 12. Our methods converge the fastest.

Shapley game. This is a two-player normal-form game with 3 actions per player. It was introduced by Shapley [79, p. 26], and is

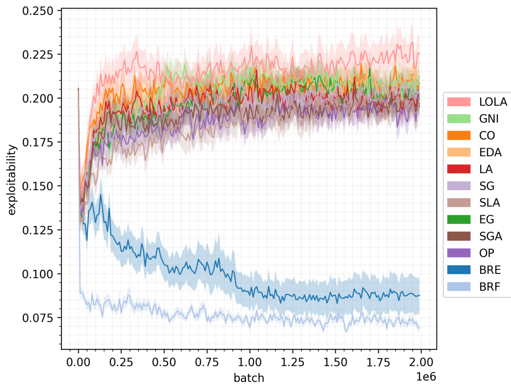

Figure 10: Security game with 1 points.

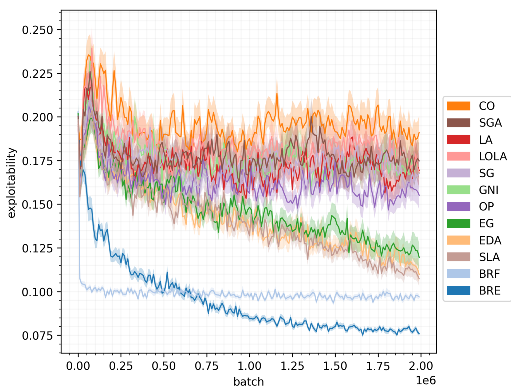

Figure 11: Security game with 2 points.

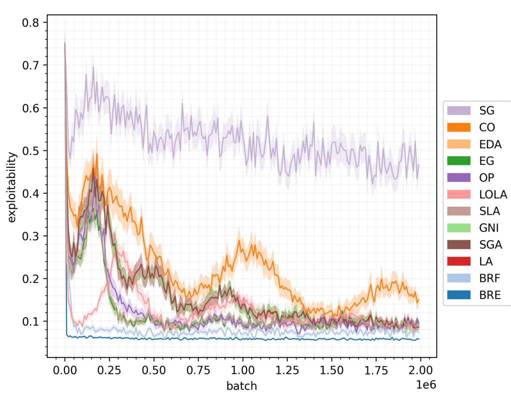

Figure 12: Glicksberg–Gross game.

a classic example of a game for which fictitious play [4, 9] diverges. Results are shown in Figure 13. Our methods converge, while the rest diverge and perform, almost identically to each other.

Poker games. Kuhn poker is a variant of poker introduced by Kuhn [55]. It is a two-player zero-sum imperfect-information game. A 3-player variant was introduced by Szafron et al. [81], and was one

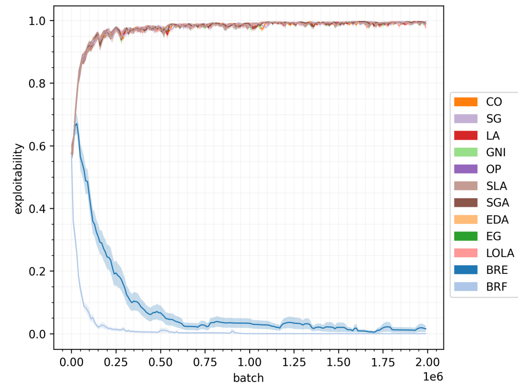

Figure 13: Shapley game.

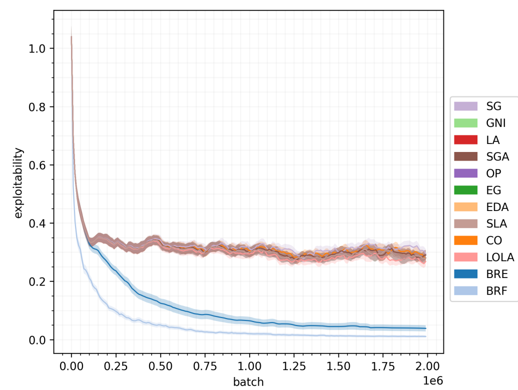

Figure 14: Kuhn poker with 2 players.

of the largest three-player games to be solved analytically to date. 2-player Kuhn poker has a 12-dimensional strategy space per player (24 in total). 3-player Kuhn poker has a 32-dimensional strategy space per player (96 in total). Thus the utility function for these games is high-dimensional and nonlinear, making them a good benchmark. We use the poker implementation of OpenSpiel [87]. Results are shown in Figures 14 and 15. Our methods converge fastest. The rest perform almost identically to each other.

Generalized rock paper scissors. Rock paper scissors (RPS) is a classic two-player zero-sum normal-form game with 3 actions per player. It has a unique mixed-strategy NE where each player mixes uniformly over its actions. Cloud et al. [12, p. 7] generalize RPS to n actions by letting $u_1(a) = -u_2(a) = [\![a_2 - a_1 = 1 \mod n]\!] - [\![a_1 - a_2 = 1 \mod n]\!]$. Results are shown in Figures 16 and 17. Our methods converge the fastest.

Generalized matching pennies. Matching pennies (MP) is a classic two-player zero-sum normal-form game with 2 actions per player. It can be generalized to n players [45, 58] by letting $u_i : [2]^n \to \mathbb{R}$ where $u_i(a) = [\![a_i = a_{i+1 \bmod n} \oplus i = n - 1]\!]$. That is, each player seeks to match the next, but the last player seeks to *un*match the first. This game has a unique mixed-strategy NE where each player mixes uniformly over its actions. Results are shown in Figures 18

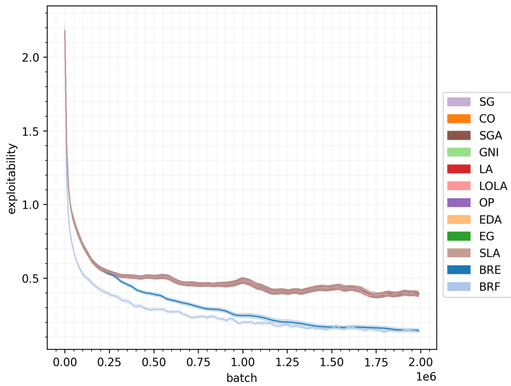

Figure 15: Kuhn poker with 3 players.

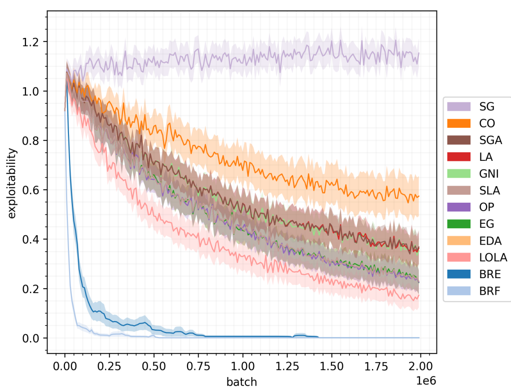

Figure 16: Rock paper scissors with 3 actions.

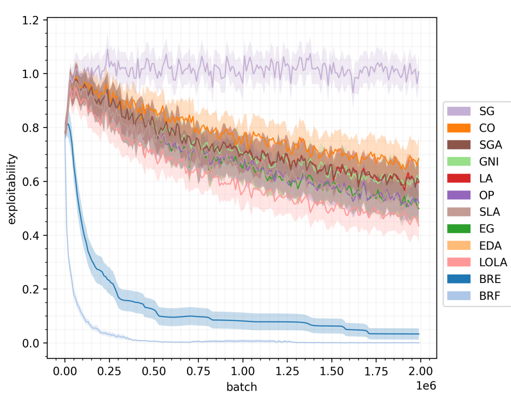

Figure 17: Rock paper scissors with 4 actions.

and 19. In the 2-player game, our methods converge fastest. In the 3-player game, our methods converge, while the rest diverge.

6 CONCLUSION

In this paper, we studied the problem of finding an approximate NE of continuous games. The main measure of closeness to NE is

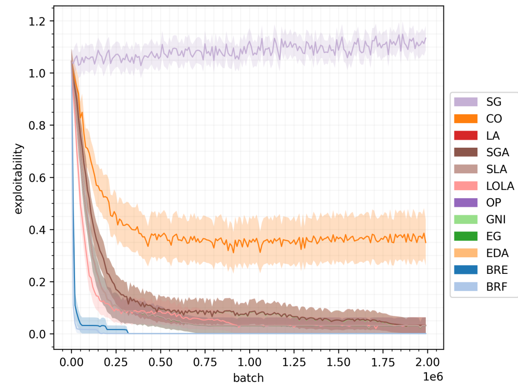

Figure 18: Matching pennies with 2 players.

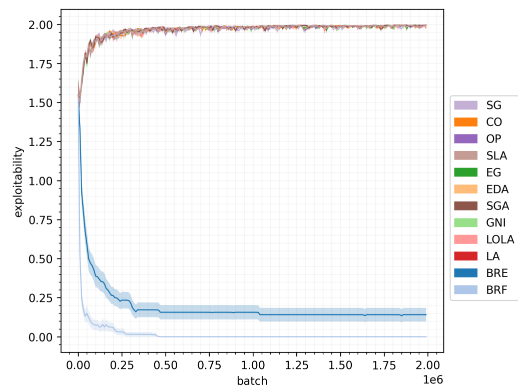

Figure 19: Matching pennies with 3 players.

exploitability, which measures how much players can benefit from unilaterally changing their strategy. We proposed two new methods that minimize an approximation of the exploitability with respect to the strategy profile. We evaluated these methods in various continuous games, showing that they outperform prior methods.

Prior equilibrium-finding techniques usually suffer from cycling or divergent behavior. By considering the equilibrium-finding problem from first principles, and formulating a novel solution to it, our paper opens up new possibilities for tackling games where such problematic behavior appears, reducing the amount of time and resources needed to obtain good approximate equilibria.

Supplementary material—including additional related research, theoretical analysis, code, additional tables and figures, and suggestions for future research—can be found in the pre-print version of this paper at https://arxiv.org/abs/2301.08830.

ACKNOWLEDGMENTS

This material is based on work supported by the Vannevar Bush Faculty Fellowship ONR N00014-23-1-2876; NSF grants CCF-1733556, RI-2312342, and RI-1901403; ARO award W911NF2210266; and NIH award A240108S001.

REFERENCES

[1] Juhan Bae and Roger B. Grosse. 2020. Delta-STN: Efficient bilevel optimization for neural networks using structured response Jacobians. *Conference on Neural Information Processing Systems (NeurIPS)* 33 (2020).

[2] David Balduzzi, Sebastien Racaniere, James Martens, Jakob Foerster, Karl Tuyls, and Thore Graepel. 2018. The mechanics of n-player differentiable games. In *International Conference on Machine Learning (ICML)*.

[3] Albert S. Berahas, Liyuan Cao, Krzysztof Choromanski, and Katya Scheinberg. 2022. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *Foundations of Computational Mathematics* 22, 2 (2022).

[4] Ulrich Berger. 2007. Brown's original fictitious play. *Journal of Economic Theory* 135, 1 (2007).

[5] Christopher Berner, Greg Brockman, Brooke Chan, Vicki Cheung, Przemysław Debiak, Christy Dennison, David Farhi, Quirin Fischer, Shariq Hashme, Chris Hesse, et al. 2019. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv preprint arXiv:1912.06680* 0 (2019).

[6] Steffan Berridge and Jacek B. Krawczyk. 1970. *Relaxation algorithms in finding Nash equilibria.* Victoria University of Wellington.

[7] Émile Borel. 1938. *Traité du calcul des probabilités et ses applications.* Applications aux jeux des hazard, Vol. IV. Gauthier-Villars.

[8] James Bradbury, Roy Frostig, Peter Hawkins, Matthew James Johnson, Chris Leary, Dougal Maclaurin, George Necula, Adam Paszke, Jake VanderPlas, Skye Wanderman-Milne, et al. 2018. JAX: Composable transformations of Python+NumPy programs.

[9] George W. Brown. 1951. Iterative Solutions of Games by Fictitious Play. In *Activity Analysis of Production and Allocation*, Tjalling C. Koopmans (Ed.). John Wiley & Sons, 374–376.

[10] Noam Brown and Tuomas Sandholm. 2019. Solving imperfect-information games via discounted regret minimization. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[11] Bill Chen and Jerrod Ankenman. 2006. *The Mathematics of Poker.* ConJelCo.

[12] Alex Cloud, Albert Wang, and Wesley Kerr. 2022. Anticipatory fictitious play. *arXiv:2212.09941* 0, 0 (2022).

[13] David F. Crouse. 2016. On implementing 2D rectangular assignment algorithms. *IEEE Trans. Aerospace Electron. Systems* 52, 4 (2016).

[14] George Cybenko. 1989. Approximation by superpositions of a sigmoidal function. *Mathematics of control, signals and systems* 2, 4 (1989).

[15] Constantinos Daskalakis, Andrew Ilyas, Vasilis Syrgkanis, and Haoyang Zeng. 2018. Training GANs with optimism. In *International Conference on Learning Representations (ICLR)*.

[16] DeepMind, Igor Babuschkin, Kate Baumli, Alison Bell, Surya Bhupatiraju, Jake Bruce, Peter Buchlovsky, David Budden, Trevor Cai, Aidan Clark, Ivo Danihelka, Antoine Dedieu, Claudio Fantacci, Jonathan Godwin, Chris Jones, Ross Hemsley, Tom Hennigan, Matteo Hessel, Shaobo Hou, Steven Kapturowski, Thomas Keck, Iurii Kemaev, Michael King, Markus Kunesch, Lena Martens, Hamza Merzic, Vladimir Mikulik, Tamara Norman, George Papamakarios, John Quan, Roman Ring, Francisco Ruiz, Alvaro Sanchez, Laurent Sartran, Rosalia Schneider, Eren Sezener, Stephen Spencer, Srivatsan Srinivasan, Miloš Stanojević, Wojciech Stokowiec, Luyu Wang, Guangyao Zhou, and Fabio Viola. 2020. The DeepMind JAX Ecosystem. http://github.com/google-deepmind

[17] Li Deng. 2012. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine* 29, 6 (2012).

[18] John C. Duchi, Michael I. Jordan, Martin J. Wainwright, and Andre Wibisono. 2015. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory* 61, 5 (2015).

[19] Carles Domingo Enrich. 2019. *Games in machine learning: Differentiable n-player games and structured planning.* Master's thesis. Universitat Politècnica de Catalunya.

[20] Gabriele Farina, Christian Kroer, and Tuomas Sandholm. 2021. Faster Game Solving via Predictive Blackwell Approachability: Connecting Regret Matching and Mirror Descent. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[21] Tanner Fiez, Benjamin Chasnov, and Lillian Ratliff. 2020. Implicit learning dynamics in stackelberg games: Equilibria characterization, convergence analysis, and empirical study. In *International Conference on Machine Learning (ICML)*. Proceedings of Machine Learning Research (PMLR).

[22] Sjur Didrik Flåm and Anatoly S. Antipin. 1996. Equilibrium programming using proximal-like algorithms. *Mathematical Programming* 78, 1 (1996).

[23] Sjur Didrik Flåm and Andrzej Ruszczyński. 2008. Finding normalized equilibrium in convex-concave games. *International Game Theory Review* 10, 1 (2008).

[24] Jakob N. Foerster, Richard Y. Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. 2018. Learning with opponent-learning awareness. In *Autonomous Agents and Multi-Agent Systems*.

[25] Sam Ganzfried. 2021. Algorithm for computing approximate Nash equilibrium in continuous games with application to continuous Blotto. *Games* 12, 2 (2021).

[26] Sam Ganzfried and Tuomas Sandholm. 2010. Computing equilibria by incorporating qualitative models. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

[27] Ian Gemp and Sridhar Mahadevan. 2018. Global convergence to the equilibrium of GANs using variational inequalities. *arXiv:1808.01531* 0, 0 (2018).

[28] Papiya Ghosh and Rajendra P. Kundu. 2019. Best-shot network games with continuous action space. *Research in Economics* 73, 3 (2019).

[29] Irving Leonard Glicksberg and Oliver Alfred Gross. 1953. Notes on games over the square. *Contributions to the theory of games* 2 (1953).

[30] Denizalp Goktas and Amy Greenwald. 2022. Exploitability minimization in games and beyond. *Conference on Neural Information Processing Systems (NeurIPS)* 35 (2022).

[31] Noah Golowich, Sarath Pattathil, Constantinos Daskalakis, and Asuman Ozdaglar. 2020. Last iterate is slower than averaged iterate in smooth convex-concave saddle point problems. In *Conference on Learning Theory (COLT)*.

[32] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2020. Generative adversarial networks. *Commun. ACM* 63, 11 (2020).

[33] Paulina Grnarova, Yannic Kilcher, Kfir Y. Levy, Aurelien Lucchi, and Thomas Hofmann. 2021. Generative minimization networks: Training GANs without competition. *arXiv:2103.12685* 0 (2021).

[34] Gül Gürkan and Jong-Shi Pang. 2009. Approximations of Nash equilibria. *Mathematical Programming* 117 (2009).

[35] David Ha, Andrew Dai, and Quoc V. Le. 2016. Hypernetworks. *arXiv:1609.09106* 0, 0 (2016).

[36] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2015. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *International Conference on Computer Vision (ICCV)*.

[37] Jonathan Heek, Anselm Levskaya, Avital Oliver, Marvin Ritter, Bertrand Rondepierre, Andreas Steiner, and Marc van Zee. 2023. Flax: A neural network library and ecosystem for JAX. http://github.com/google/flax

[38] Anna Von Heusinger and Christian Kanzow. 2009. Optimization reformulations of the generalized Nash equilibrium problem using Nikaido-Isoda-type functions. *Computational Optimization and Applications* 43 (2009).

[39] Anna Von Heusinger and Christian Kanzow. 2009. Relaxation methods for generalized Nash equilibrium problems with inexact line search. *Journal of Optimization Theory and Applications (JOTA)* 143 (2009).

[40] Kurt Hornik. 1991. Approximation capabilities of multilayer feedforward networks. *Neural networks* 4, 2 (1991).

[41] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. 1989. Multilayer feedforward networks are universal approximators. *Neural networks* 2, 5 (1989).

[42] Jian Hou, Zong-Chuan Wen, and Qing Chang. 2018. An unconstrained optimization reformulation for the Nash game. *Journal of Interdisciplinary Mathematics (JIM)* 21, 5 (2018).

[43] Yu-Guan Hsieh, Franck Iutzeler, Jérôme Malick, and Panayotis Mertikopoulos. 2019. On the convergence of single-call stochastic extra-gradient methods. In *Conference on Neural Information Processing Systems (NeurIPS)*.

[44] John D. Hunter. 2007. Matplotlib: A 2D graphics environment. *Computing in Science & Engineering* 9, 3 (2007).

[45] James S. Jordan. 1993. Three problems in learning mixed-strategy Nash equilibria. *Games and Economic Behavior* 5, 3 (1993).

[46] Nitin Kamra, Fei Fang, Debarun Kar, Yan Liu, and Milind Tambe. 2017. Handling continuous space security games with neural networks. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

[47] Nitin Kamra, Umang Gupta, Fei Fang, Yan Liu, and Milind Tambe. 2018. Policy learning for continuous space security games using neural networks. *AAAI Conference on Artificial Intelligence (AAAI)* 32, 1 (2018).

[48] Nitin Kamra, Umang Gupta, Kai Wang, Fei Fang, Yan Liu, and Milind Tambe. 2019. DeepFP for finding Nash equilibrium in continuous action spaces. In *Decision and Game Theory for Security*.

[49] Debarun Kar, Thanh H Nguyen, Fei Fang, Matthew Brown, Arunesh Sinha, Milind Tambe, and Albert Xin Jiang. 2017. Trends and applications in Stackelberg security games. In *Handbook of dynamic game theory*. Springer International Publishing.

[50] G. M. Korpelevich. 1976. The extragradient method for finding saddle points and other problems. *Ekonomika i Matematicheskie Metody* 12, 4 (1976).

[51] Dmytro Korzhyk, Zhengyu Yin, Christopher Kiekintveld, Vincent Conitzer, and Milind Tambe. 2011. Stackelberg vs. Nash in security games. *Journal of Artificial Intelligence Research* 41 (2011).

[52] Jacek B. Krawczyk. 2005. Coupled constraint Nash equilibria in environmental games. *Resource and Energy Economics* 27, 2 (2005).

[53] Jacek B. Krawczyk and Stanislav Uryasev. 2000. Relaxation algorithms to find Nash equilibria with economic applications. *Environmental Modeling & Assessment* 5 (2000).

[54] Christian Kroer and Tuomas Sandholm. 2015. Discretization of Continuous Action Spaces in Extensive-Form Games. In *International Conference on Autonomous Agents and Multi-Agent Systems (AAMAS)*.

[55] Harold William Kuhn. 1950. A Simplified Two-Person Poker. In *Contributions to the Theory of Games*, Harold William Kuhn and Albert William Tucker (Eds.). Annals of Mathematics Studies, 24, Vol. 1. Princeton University Press, 97–103.

[56] Marc Lanctot, Vinicius Zambaldi, Audrunas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. 2017. A unified game-theoretic approach to multiagent reinforcement learning. In *Conference on Neural Information Processing Systems (NeurIPS)*.

[57] Moshe Leshno, Vladimir Ya Lin, Allan Pinkus, and Shimon Schocken. 1993. Multilayer feedforward networks with a nonpolynomial activation function can approximate any function. *Neural networks* 6, 6 (1993).

[58] David S. Leslie and Edmund J. Collins. 2003. Convergent multiple-timescales reinforcement learning algorithms in normal form games. *The Annals of Applied Probability* 13, 4 (2003).

[59] Alistair Letcher. 2018. *Stability and exploitation in differentiable games.* Master's thesis. University of Oxford.

[60] Edward Lockhart, Marc Lanctot, Julien Pérolat, Jean-Baptiste Lespiau, Dustin Morrill, Finbarr Timbers, and Karl Tuyls. 2019. Computing approximate equilibria in sequential adversarial games by exploitability descent. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

[61] Jonathan Lorraine and David Duvenaud. 2018. Stochastic hyperparameter optimization through hypernetworks. *arXiv:1802.09419* 0 (2018).

[62] Jeffrey Ma, Alistair Letcher, Florian Schäfer, Yuanyuan Shi, and Anima Anandkumar. 2021. Polymatrix Competitive Gradient Descent. *arXiv:2111.08565* 0, 0 (2021).

[63] Matthew MacKay, Paul Vicol, Jon Lorraine, David Duvenaud, and Roger Grosse. 2019. Self-tuning networks: Bilevel optimization of hyperparameters using structured best-response functions. *arXiv:1903.03088* 0, 0 (2019).

[64] Alberto Marchesi, Francesco Trovò, and Nicola Gatti. 2020. Learning Probably Approximately Correct Maximin Strategies in Simulation-Based Games with Infinite Strategy Spaces. In *Autonomous Agents and Multi-Agent Systems*.

[65] Eric V. Mazumdar, Michael I. Jordan, and S. Shankar Sastry. 2019. On finding local Nash equilibria (and only local Nash equilibria) in zero-sum games. *arXiv:1901.00838* 0 (2019).

[66] Lars Mescheder, Sebastian Nowozin, and Andreas Geiger. 2017. The numerics of GANs. In *Conference on Neural Information Processing Systems (NeurIPS)*.

[67] Luke Metz, C. Daniel Freeman, Samuel S. Schoenholz, and Tal Kachman. 2021. Gradients are not all you need. *arXiv:2111.05803* 0 (2021).

[68] Yurii Nesterov and Vladimir Spokoiny. 2017. Random gradient-free minimization of convex functions. *Foundations of Computational Mathematics* 17, 2 (2017).

[69] Hukukane Nikaidô and Kazuo Isoda. 1955. Note on non-cooperative convex games. *Pacific J. Math.* 5 (1955).

[70] Allan Pinkus. 1999. Approximation theory of the MLP model in neural networks. *Acta numerica* 8 (1999).

[71] Leonid Denisovich Popov. 1980. A modification of the Arrow-Hurwicz method for search of saddle points. *Mathematical notes of the Academy of Sciences of the USSR* 28 (1980).

[72] Biao Qu and Jing Zhao. 2013. Methods for solving generalized Nash equilibrium. *Journal of Applied Mathematics* 2013, 1 (2013).

[73] Arvind Raghunathan, Anoop Cherian, and Devesh Jha. 2019. Game theoretic optimization via gradient-based Nikaido-Isoda function. In *International Conference on Machine Learning (ICML)*.

[74] Lars Ruthotto and Eldad Haber. 2021. An introduction to deep generative modeling. *GAMM-Mitteilungen* 44, 2 (2021).

[75] Tim Salimans, Jonathan Ho, Xi Chen, Szymon Sidor, and Ilya Sutskever. 2017. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv:1703.03864* 0, 0 (2017).

[76] Florian Schäfer and Anima Anandkumar. 2019. Competitive gradient descent. In *Conference on Neural Information Processing Systems (NeurIPS)*.

[77] Jürgen Schmidhuber. 1992. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation* 4, 1 (1992).

[78] Ohad Shamir. 2017. An optimal algorithm for bandit and zero-order convex optimization with two-point feedback. *Journal of Machine Learning Research* 18, 52 (2017).

[79] Lloyd .S Shapley. 1964. Some topics in two-person games. In *Advances in Game Theory*, M. Drescher, L. S. Shapley, and Albert William Tucker (Eds.). Princeton University Press.

[80] Arunesh Sinha, Fei Fang, Bo An, Christopher Kiekintveld, and Milind Tambe. 2018. Stackelberg security games: Looking beyond a decade of success. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

[81] Duane Szafron, Richard G. Gibson, and Nathan R. Sturtevant. 2013. A parameterized family of equilibrium profiles for three-player Kuhn poker. In *Autonomous Agents and Multi-Agent Systems*.

[82] Finbarr Timbers, Nolan Bard, Edward Lockhart, Marc Lanctot, Martin Schmid, Neil Burch, Julian Schrittwieser, Thomas Hubert, and Michael Bowling. 2022. Approximate exploitability: Learning a best response. In *Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI)*.

[83] Ioannis Tsaknakis and Mingyi Hong. 2021. Finding first-order Nash equilibria of zero-sum games with the regularized Nikaido-Isoda function. In *International Conference on Artificial Intelligence and Statistics (AISTATS)*.

[84] Stanislav Uryasev and Reuven Y. Rubinstein. 1994. On relaxation algorithms in computation of noncooperative equilibria. *IEEE Transactions on Automatic Control (TACON)* 39, 6 (1994).

[85] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. 2019. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* 575, 7782 (2019).

[86] Pauli Virtanen, Ralf Gommers, Travis E. Oliphant, Matt Haberland, Tyler Reddy, David Cournapeau, Evgeni Burovski, Pearu Peterson, Warren Weckesser, Jonathan Bright, et al. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods* 17, 3 (2020).

[87] Michael Walton and Viliam Lisy. 2021. Multi-agent Reinforcement Learning in OpenSpiel: A Reproduction Report. *arXiv preprint arXiv:2103.00187* 0 (2021).

[88] Michael P. Wellman, Karl Tuyls, and Amy Greenwald. 2024. Empirical game-theoretic analysis: A survey. *arXiv:2403.04018* 0 (2024).

[89] Chongjie Zhang and Victor Lesser. 2010. Multi-agent learning with policy prediction. In *AAAI Conference on Artificial Intelligence (AAAI)*.

[90] Martin Zinkevich, Michael Bowling, Michael Johanson, and Carmelo Piccione. 2007. Regret Minimization in Games with Incomplete Information. In *Conference on Neural Information Processing Systems (NeurIPS)*.