Joint-Perturbation Simultaneous Pseudo-Gradient*

Carlos Martin[1] and **Tuomas Sandholm**[1,2,3,4]
[1]Carnegie Mellon University
[2]Strategy Robot, Inc.
[3]Optimized Markets, Inc.
[4]Strategic Machine, Inc.
{cgmartin, sandholm}@cs.cmu.edu

Abstract

We study the problem of computing an approximate Nash equilibrium of a game whose strategy space is continuous without access to gradients of the utility function. Lack of access to gradients is common in reinforcement learning settings, where the environment is treated as a black box, as well as equilibrium finding in mechanisms such as auctions, where the mechanism's payoffs are discontinuous in the players' actions. To tackle this problem, we turn to zeroth-order optimization techniques that combine pseudo-gradients with equilibrium-finding dynamics. Specifically, we introduce a new technique that requires a number of utility function evaluations per iteration that is constant rather than linear in the number of players. It achieves this by performing a single joint perturbation on all players' strategies, rather than perturbing each one individually. This is very important for many-player games, especially when the utility function is expensive to compute in terms of wall time, memory, money, or other resources. We evaluate our approach on various games, including auctions, which have important real-world applications. Our approach yields a dramatic improvement in performance in terms of the wall time required to reach an approximate Nash equilibrium.

1 Introduction

We tackle the problem of computing an approximate Nash equilibrium of a game with a black-box utility function, for which we lack access to gradients. A standard way to learn player strategies for a game is to use simultaneous gradient ascent, in which, at each iteration, each player myopically adjusts its parameters to increase its own utility, treating the other players as fixed. Computing the simultaneous gradient requires taking gradients of the utility function, which are unavailable in the black-box setting. To address this obstruction, one can employ evolution strategies, a family of methods that perturbs parameters and evaluates the function at those perturbed points in order to optimize some objective. This yields an unbiased estimator of the gradient of a smoothed version of the original objective function, also called a *pseudo-gradient*. Computing the simultaneous pseudo-gradient through the standard approach requires a number of utility function evaluations that is *linear* in the number of players. Our contribution is to introduce a new method which requires a number of function evaluations that is only *constant* in the number of players. It performs a joint perturbation on *all* players' strategies at once, rather than perturbing each one individually. When utility function evaluation is expensive (in terms of wall time, memory, money, *etc.*) and the number of players is large, this can yield dramatic benefits for training. We benchmark our approach on several games, including various auctions, showing a significant reduction in training time. The rest of this paper is structured as follows. In §2, we describe related research. In §3, we introduce relevant notation and present a mathematical formulation of the problem we are solving. In §4, we present our method. In §5, we present our experimental settings, results, and discussion. In §6, we present our conclusions.

2 Related Research

Black-box zeroth-order optimization uses only function evaluations to optimize a black-box function with respect to a set of inputs. In particular, it does not require gradients. There is a class of black-box optimization algorithms called *evolution strategies (ES)* [Rechenberg and Eigen, 1973; Schwefel, 1977; Rechenberg, 1978; Bäck, 1996; Bäck *et al.*, 1997; Eiben and Smith, 2003]. These maintain and evolve a population of parameter vectors. *Natural evolution strategies (NES)* [Wierstra and others, 2008; Yi and others, 2009; Wierstra and others, 2014] represent the population as a *distribution over parameters* and maximize its average objective value using the score function estimator. For many parameter distributions, such as Gaussian smoothing, this is equivalent to evaluating the function at randomly-sampled points and estimating the gradient as a sum of estimates of directional derivatives along random directions [Fu and others, 2015; Duchi and others, 2015; Nesterov and Spokoiny, 2017; Shamir, 2017; Berahas and others, 2022]. Li and Wellman [2021] tackle the problem of solving symmetric one-shot Bayesian games with no given analytic structure, high-dimensional type and action spaces, many players, and general-sum payoffs. They represent agent strategies in parametric form as neural networks, and apply NES to optimize them. For pure equilibrium computation,

*An extended version of this paper can be found at https://arxiv.org/abs/2408.09306.

they formulate the problem as bi-level optimization and use NES to implement both inner-loop best response optimization and outer-loop regret minimization. For mixed equilibrium computation, they adopt an incremental strategy generation framework in which NES produces a finite sequence of approximate best-response strategies. They then calculate equilibria over this finite strategy set via a model-based optimization process. Both methods use NES to search for strategies over the functional space of policies, given only black-box simulation access to noisy payoff samples. To tackle symmetric auctions, Bichler *et al.* [2021] present a learning method called *neural pseudogradient ascent (NPGA)* that represents strategies as neural networks and applies policy iteration on the basis of gradient dynamics in self-play to provably learn local equilibria. The method follows the simultaneous gradient of the game and uses a smoothing technique to circumvent discontinuities in the *ex post* utility functions of auction games. (In auctions, discontinuities arise at the bid value where an arbitrarily small change makes the difference between winning and not winning.) The method converges to a Bayesian Nash equilibrium for a wide variety of symmetric auction games. Bichler *et al.* [2023b] analyze a wide variety of asymmetric auction models. Their results show that they closely approximate Bayesian Nash equilibria in all models in which the analytical Bayes–Nash equilibrium is known. Additionally, they analyze new and larger environments for which no analytical solution is known and verify that the solution found approximates equilibrium closely. Bichler *et al.* [2023a] introduce an algorithmic framework for Bayesian games with continuous type and action spaces. It discretizes the type and action spaces and then learns distributional strategies [Milgrom and Weber, 1985] (a form of mixed strategies for Bayesian games) via online convex optimization, specifically *simultaneous online dual averaging (SODA)*. They show that the equilibrium of the discretized game approximates an equilibrium in the continuous game. Discretization can work well for small games, but does not scale to high-dimensional observation and action spaces. Martin and Sandholm [2023] study the problem of computing an approximate Nash equilibrium of continuous-action game without access to gradients. They model players' strategies as artificial neural networks. In particular, they use randomized policy networks to model mixed strategies. These take noise in addition to an observation as input and can flexibly represent arbitrary observation-dependent, continuous-action distributions. Being able to model such mixed strategies is crucial for tackling continuous-action games that lack pure-strategy equilibria. They apply this method to continuous Colonel Blotto games, single-item and multi-item auctions, and a visibility game, showing that it can quickly find a high-quality approximate equilibrium.

3 Problem Formulation

Throughout the paper, we use the following notation. The operator $\otimes$ denotes the tensor product. The operator $\odot$ denotes the elementwise a.k.a. Hadamard product. If $\mathcal{S}$ is a set, $\triangle\mathcal{S}$ is the set of all Borel probability measures on $\mathcal{S}$. If $\mathcal{F}$ is a family indexed by $\mathcal{I}$, $\mathcal{F}_\times = \prod_{i\in I} \mathcal{F}_i$. If $\mu_i \in \triangle\mathcal{S}_i$ is a probability measure for $i \in \mathcal{I}$, $\bigotimes_{i\in\mathcal{I}} \mu_i \in \triangle\mathcal{S}_\times$ is the product measure.

We now introduce some key game-theoretic concepts that are needed to formally understand our problem and subsequent solution.

Strategic-form game. A strategic-form game is a tuple $(\mathcal{I}, \mathcal{S}, u)$ where $\mathcal{I}$ is a set of players, $\mathcal{S}_i$ is a set of strategies for $i \in \mathcal{I}$, and $u : \mathcal{S}_\times \to \mathbb{R}^{\mathcal{I}}$ is a utility function. A strategy profile is an element of $\mathcal{S}_\times$, that is, an assignment of a strategy to each player. The notation s_{-i} denotes s excluding $i \in \mathcal{I}$. Given a strategy profile, a *best response (BR)* for a player is a strategy that maximizes its utility given the other players' strategies. That is, given $s \in \mathcal{S}_\times$, a BR for $i \in \mathcal{I}$ is an element of $\mathcal{B}_i(s_{-i}) = \operatorname{argmax}_{r_i\in\mathcal{S}_i} u(r_i, s_{-i})_i$. A *Nash equilibrium (NE)* is a strategy profile for which each player's strategy is a BR to the other players' strategies. That is, it is an $s \in \mathcal{S}_\times$ such that $s_i \in \mathcal{B}_i(s_{-i})$ for all $i \in \mathcal{I}$.

Exploitability. Given $s \in \mathcal{S}_\times$, the BR utility of $i \in \mathcal{I}$ is $b_i(s_{-i}) = \sup_{r_i\in\mathcal{S}_i} u_i(r_i, s_{-i})$, *i.e.*, the highest utility it could attain by unilaterally changing its strategy. Player i's regret is $R_i(s) = b_i(s_{-i}) - u_i(s)$, which is the highest utility it could *gain* from unilaterally changing its strategy. The *exploitability* is defined as $\Phi = \sum_{i\in\mathcal{I}} R_i$. It is non-negative everywhere and zero precisely at NE. Consequently, it is used as a standard measure of "closeness" to NE in the literature [Lanctot and others, 2017; Lockhart and others, 2019; Walton and Lisy, 2021; Timbers and others, 2022]. If NE exist, computing them is equivalent to globally minimizing exploitability.

Mixed strategies. For any game $(\mathcal{I}, \mathcal{S}, u)$, there is a game $(\mathcal{I}, \Sigma, \bar{u})$ where $\Sigma_i = \triangle\mathcal{S}_i$ and $\bar{u}_i(\sigma) = \mathrm{E}_{s\sim\bigotimes_{j\in\mathcal{I}}\sigma_j} u_i(s)$. That is, each player's strategy is a probability measure over its original strategy set (*i.e.*, a mixed strategy), and its utility is the resulting expected utility in the original game. A mixed-strategy NE of the original game is an NE of this new game.[1]

Continuous game. A continuous-action game is a game whose strategy sets are subsets of Euclidean space, *e.g.*, $\mathcal{S}_i \subseteq \mathbb{R}^d$. The question of the existence and uniqueness of Nash equilibria for such games has been studied extensively in the literature. We briefly review some of these results to show that the continuous games we are interested in do, in fact, possess Nash equilibria, and thus that it makes sense to try to find them with our method. Nash [1950] showed that if each $\mathcal{S}_i$ is nonempty and finite, a mixed-strategy NE exists. Glicksberg [1952] showed that if each $\mathcal{S}_i$ is nonempty and compact, and each u_i is continuous, a mixed-strategy NE exists. Glicksberg [1952], Fan [1952], and Debreu [1952] showed that if each $\mathcal{S}_i$ is nonempty, compact, and convex, and each u_i is continuous and quasiconcave in s_i, a pure strategy NE exists. Dasgupta and Maskin [1986] showed that if each $\mathcal{S}_i$ is nonempty, compact, convex, and u_i is upper

[1] More generally, one can consider settings where randomness is a limited resource (*e.g.*, only a limited number of random bits are available to the agent) or where the agent can only mix between a limited number of pure strategies (*i.e.*, its mixed strategy must be *sparse*). Alternatively, its mixed strategy may be restricted to some class of *representable* distributions, such as an explicit parametric model or implicit density model like a Generative Adversarial Network [Goodfellow and others, 2014; Goodfellow and others, 2020].

semicontinuous and graph continuous and quasiconcave in s_i, a pure strategy NE exists. They also showed that if each $\mathcal{S}_i$ is nonempty, compact, and convex, and u_i is bounded and continuous except on a subset (defined by technical conditions) and weakly lower semicontinuous in s_i, and $\sum_{i\in\mathcal{I}} u_i$ is upper semicontinuous, a mixed-strategy NE exists. Rosen [1965] proved the uniqueness of a pure NE for continuous-action games under diagonal strict concavity assumptions.

4 Proposed Method

In order to build up to our method, we first introduce some key concepts in a gradual fashion.

Pseudo-gradient. Let $d \in \mathbb{N}$ be a natural number, $f : \mathbb{R}^d \to \mathbb{R}$ be a function, $\mu = \mathcal{N}(\mathbf{0}_d, \mathbf{I}_d)$ be the d-dimensional standard normal distribution, $\sigma \in \mathbb{R}$ be a scalar, and $f_\sigma(\mathbf{x}) = \mathrm{E}_{\mathbf{z}\sim\mu} f(\mathbf{x} + \sigma\mathbf{z})$. Assume $\sigma > 0$. Then $f_\sigma = f * G_\sigma$, where G_σ is a Gaussian kernel of width σ. It is a property of convolutions that $\nabla(f * G_\sigma) = f * \nabla G_\sigma$. Therefore, since G_σ is smooth (*i.e.*, infinitely differentiable), f_σ is also smooth. This holds even if f itself is not even continuous. In the literature, ∇f_σ is called the *pseudo-gradient* of f. It satisfies the identity $\nabla f_\sigma(\mathbf{x}) = \mathrm{E}_{\mathbf{z}\sim\mu} \frac{1}{\sigma} f(\mathbf{x} + \sigma\mathbf{z})\mathbf{z}$. This yields an unbiased estimator that forms the basis of OpenAI ES [Salimans and others, 2017], an NES method, as described in §2. The pseudo-gradient can be used to perform zeroth-order, *i.e.* gradient-free, optimization of f. This can be useful in cases where straightforward stochastic gradient ascent is not possible or desirable. For example, f might not even be continuous in the first place, let alone differentiable. In other situations, f *is* differentiable, but obtaining an unbiased estimator of its gradient is difficult or intractable. This can happen if, for example, the utility function is an expectation of a non-differentiable payoff function. An example of such a situation is an auction, wherein a player wins an item as soon as its bid exceeds a threshold, yielding a payoff discontinuity. In such a situation, it might be the case that the *ex ante* utilities are differentiable but the *ex post* utilities are not [Bichler and others, 2021]. The use of pseudo-gradients for optimization has been studied by Duchi *et al.* [2015], Nesterov and Spokoiny [2017], Shamir [2017], Salimans *et al.* [2017], Berahas *et al.* [2022], and Metz *et al.* [2021], among others. It has been shown to be a scalable alternative to classical methods in reinforcement learning [Salimans and others, 2017].

Variance reduction. We define

$$\mathbf{g}_{\text{SP}} = \tfrac{1}{\sigma} f(\mathbf{x} + \sigma\mathbf{z})\mathbf{z} \tag{1}$$

$$\mathbf{g}_{\text{FD}} = \tfrac{1}{\sigma}(f(\mathbf{x} + \sigma\mathbf{z}) - f(\mathbf{x}))\mathbf{z} \tag{2}$$

$$\mathbf{g}_{\text{CD}} = \tfrac{1}{2\sigma}(f(\mathbf{x} + \sigma\mathbf{z}) - f(\mathbf{x} - \sigma\mathbf{z}))\mathbf{z} \tag{3}$$

as the *single-point (SP)*, *forward-difference (FD)*, and *centered-difference (CD)* (or antithetic) estimators, respectively. These are all unbiased estimators of the pseudo-gradient. However, the first has a large variance, so the latter two are typically used in practice [Berahas and others, 2022]. The variance of the pseudo-gradient estimator can be further reduced by taking a mean over many perturbations, as in $\frac{1}{2n\sigma}\sum_{i=1}^{n}(f(\mathbf{x}+\sigma\mathbf{z}_i) - f(\mathbf{x}-\sigma\mathbf{z}_i))\mathbf{z}_i$, where $\mathbf{z}_i \sim \mu$ are independent. Other variance reduction techniques are surveyed in Mohamed *et al.* [2020], among other works.

Simultaneous gradient. One common approach to game-solving in the literature is *simultaneous gradient ascent (SGA)*, which is defined as follows. Let $\mathbf{u} : \mathbb{R}^{n\times d} \to \mathbb{R}^n$ be a utility function, where n is the number of players and d is the dimensionality of each player's strategy parameters. The *simultaneous gradient* of $\mathbf{u}$ is the function $\mathbf{v} : \mathbb{R}^{n\times d} \to \mathbb{R}^{n\times d}$ where $\mathbf{v}_i = \nabla_i \mathbf{u}_i$. That is, for each player, it is the derivative that player's utility with respect to that player's parameters. Equivalently, $\mathbf{v} = \operatorname{diag} \nabla\mathbf{u}$, where $\nabla\mathbf{u}$ is the Jacobian of $\mathbf{u}$. SGA consists of discretizing the ordinary differential equation $\frac{\mathrm{d}}{\mathrm{d}t}\mathbf{x} = \mathbf{v}(\mathbf{x})$ in time. That is, each player tries to greedily increase their own utility, acting as if the other players are fixed. Explicitly, it uses the iteration scheme $\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha_t \mathbf{v}(\mathbf{x}_t)$ for $t \in \mathbb{N}$, where $\alpha_t > 0$ is some stepsize. Mertikopoulos and Zhou [2019] analyze the convergence of SGA to NE. They prove that, if the game admits a pseudoconcave potential or if it is monotone, the players' actions converge to NE, no matter the level of uncertainty affecting the players' feedback. Bichler *et al.* [2021] write that most auctions in the literature assume symmetric bidders and symmetric equilibrium bid functions [Krishna, 2002]. This symmetry creates a potential game, and SGA provably converges to a pure local NE in finite-dimensional continuous potential games [Mazumdar *et al.*, 2020]. Thus in any symmetric and smooth auction game, symmetric gradient ascent with appropriate (square-summable but not summable) step sizes almost surely converges to a local ex-ante approximate Bayes–NE [Bichler and others, 2021, Proposition 1]. We emphasize that the aforementioned results are only *sufficiency* results, not *necessity* results. That is, they *prove that* SGA converges under certain conditions. This does not mean that SGA does *not* converge under more general conditions. Indeed, in practice, SGA has been observed to converge under more general conditions for a wider class of games; hence the popularity of SGA in multiagent reinforcement learning.

Optimistic gradient. Another approach to game-solving in the literature is *optimistic gradient ascent (OGA)*, which iterates $\mathbf{x}_{t+1} = \mathbf{x}_t + \alpha_t \mathbf{v}(\mathbf{x}_t) + \beta_t(\mathbf{v}(\mathbf{x}_t) - \mathbf{v}(\mathbf{x}_{t-1}))$ for $t \in \mathbb{N}$, where $\alpha_t, \beta_t \in \mathbb{R}$. In the standard version of OGA, $\alpha_t = \beta_t$. OGA uses the past simultaneous gradient $\mathbf{v}(\mathbf{x}_{t-1})$ to create an extrapolation or prediction of the future simultaneous gradient $\mathbf{v}(\mathbf{x}_{t+1})$, and updates according to this prediction. OGA converges in some games where SGA fails to converge or diverges. OGA has been analyzed by Popov [1980], Daskalakis *et al.* [2018], and Hsieh *et al.* [2019], among others.[2]

Simultaneous pseudo-gradient. Both SGA and OGA, as well as other game-solving approaches in the literature, require computing the simultaneous gradient $\mathbf{v}$. However, in some situations, $\mathbf{v}$ does not exist because $\mathbf{u}$ is not differentiable. In other situations, $\mathbf{u}$ *is* differentiable, but obtaining

[2]There are also other learning dynamics in the literature, which are surveyed and analyzed by Balduzzi *et al.* [2018], Letcher *et al.* [2019b], Letcher *et al.* [2019a], Mertikopoulos and Zhou [2019], Mazumdar *et al.* [2019], Hsieh *et al.* [2021], and Willi *et al.* [2022], among others, but these are beyond the scope of this paper.

an unbiased estimator of its gradient is difficult or intractable. This can happen if, for example, $\mathbf{u}$ is an expectation (with respect to a distribution parameterized by $\mathbf{x}$) of some non-differentiable function. An example of such a situation is an auction, which we will describe in §5. To resolve this problem, we replace the gradient of $\mathbf{u}_i$ in the definition of $\mathbf{v}$ with a pseudo-gradient. Explicitly, $\mathbf{g}_i = \frac{1}{\sigma}\mathbf{u}(\mathbf{x}_i + \sigma\mathbf{z}_i, \mathbf{x}_{-i})_i\mathbf{z}_i$ for each Player i, where $\mathbf{z}_i \sim \mu_i$ and μ_i is a multivariate standard normal distribution of the same dimension as $\mathbf{x}_i$. That is, we estimate the pseudo-gradient of $\mathbf{u}_i$ (which is a scalar-valued function, since it outputs only the utility of Player i) with respect to the parameters of Player i. This is the approach taken by Bichler *et al.* [2021]. It requires one perturbation for each player and subsequent evaluation of $\mathbf{u}$. Thus, the number of utility function evaluations per iteration is linear in the number of players.

Pseudo-Jacobian. We extend the preceding concept of the pseudo-gradient from a scalar-valued function to a vector-valued function. Let $n \in \mathbb{N}$, $\mathbf{f} : \mathbb{R}^d \to \mathbb{R}^n$, and $\mathbf{f}_\sigma(\mathbf{x}) = \mathrm{E}_{\mathbf{z}\sim\mu}\,\mathbf{f}(\mathbf{x} + \sigma\mathbf{z})$. By analogy with the pseudo-gradient, we call $\nabla\mathbf{f}_\sigma$ the *pseudo-Jacobian* of $\mathbf{f}$. Furthermore, it satisfies the identity $\nabla\mathbf{f}_\sigma(\mathbf{x}) = \mathrm{E}_{\mathbf{z}\sim\mu}\,\frac{1}{\sigma}\mathbf{f}(\mathbf{x} + \sigma\mathbf{z}) \otimes \mathbf{z}$. Therefore, we have an unbiased estimator for it.

In the remainder of this subsection, we show that this estimator is not "too noisy". (For example, it is possible in principle for an estimator to be unbiased, but have very large or even infinite variance.) To do this, we give quantitative upper bounds on the moments of the magnitude of this estimator. Suppose that we use the lower-variance central difference estimator from Equation 3. This estimator is $\mathbf{J} = \frac{\mathbf{f}(\mathbf{x}+\sigma\mathbf{z})-\mathbf{f}(\mathbf{x}-\sigma\mathbf{z})}{2\sigma} \otimes \mathbf{z}$ where $\mathbf{z}$ is a sample from the standard d-dimensional multivariate normal distribution. Suppose $\mathbf{f}$ is α-Hölder continuous with constant C. That is, for all $\mathbf{x}, \mathbf{y} \in \operatorname{dom}\mathbf{f}$, $\|\mathbf{f}(\mathbf{x}) - \mathbf{f}(\mathbf{y})\| \le C\|\mathbf{x} - \mathbf{y}\|^\alpha$. Then

$$\|\mathbf{J}\|_{\mathrm{F}} = \frac{\|\mathbf{f}(\mathbf{x} + \sigma\mathbf{z}) - \mathbf{f}(\mathbf{x} - \sigma\mathbf{z})\|}{2\sigma}\|\mathbf{z}\| \tag{4}$$

$$\le \frac{C\|2\sigma\mathbf{z}\|^\alpha}{2\sigma}\|\mathbf{z}\| \tag{5}$$

$$= (2\sigma)^{\alpha-1}C\|\mathbf{z}\|^{\alpha+1} \tag{6}$$

where $\|\cdot\|_{\mathrm{F}}$ is the matrix Frobenius norm. Now, the nth raw moment of the chi distribution is $\mathbb{E}\,\|\mathbf{z}\|^n = 2^{n/2}\Gamma(\frac{d+n}{2})/\Gamma(\frac{d}{2})$, where $d \in \mathbb{N}$ is the dimensionality of $\mathbf{z}$. Thus

$$\mathbb{E}\,\|\mathbf{J}\|_{\mathrm{F}}^n \le (2\sigma)^{n(\alpha-1)}C^n\,\mathbb{E}\,\|\mathbf{z}\|^{n(\alpha+1)} \tag{7}$$

$$= (2\sigma)^{n(\alpha-1)}C^n 2^{n(\alpha+1)/2}\Gamma(\tfrac{d+n(\alpha+1)}{2})/\Gamma(\tfrac{d}{2}) \tag{8}$$

$$= 2^{n(3\alpha-1)/2}\sigma^{n(\alpha-1)}C^n\Gamma(\tfrac{d+n(\alpha+1)}{2})/\Gamma(\tfrac{d}{2}) \tag{9}$$

This yields an upper bound on any moment of the norm of the estimator. It can be applied to C-Lipschitz functions with $\alpha = 1$, and to C-bounded-range functions with $\alpha = 0$.

Joint perturbation. We now combine all of the preceding concepts that have been discussed so far. That is, we combine (1) the identity $\mathbf{v} = \operatorname{diag}\nabla\mathbf{u}$, (2) the concept of the pseudo-Jacobian, and (3) the identity $\operatorname{diag}(\mathbf{a} \otimes \mathbf{b}) = \mathbf{a} \odot \mathbf{b}$ to obtain an estimator that requires only a *single*, joint perturbation across all players. Specifically, let $\mathbf{v}_\sigma = \mathbf{v} * G_\sigma$, that is, the smoothing of $\mathbf{v}$ by a Gaussian kernel of width σ. Then

$$\mathbf{v}_\sigma(\mathbf{x}) = (\mathbf{v} * G_\sigma)(\mathbf{x}) \tag{10}$$

$$= ((\operatorname{diag}\nabla\mathbf{u}) * G_\sigma)(\mathbf{x}) \tag{11}$$

$$= \operatorname{diag}(\nabla\mathbf{u} * G_\sigma)(\mathbf{x}) \tag{12}$$

$$= \operatorname{diag}\nabla(\mathbf{u} * G_\sigma)(\mathbf{x}) \tag{13}$$

$$= \operatorname{diag}\nabla\mathbf{u}_\sigma(\mathbf{x}) \tag{14}$$

$$= \operatorname{diag}\mathrm{E}_{\mathbf{z}\sim\mu}\,\tfrac{1}{\sigma}\mathbf{u}(\mathbf{x} + \sigma\mathbf{z}) \otimes \mathbf{z} \tag{15}$$

$$= \mathrm{E}_{\mathbf{z}\sim\mu}\,\tfrac{1}{\sigma}\operatorname{diag}\mathbf{u}(\mathbf{x} + \sigma\mathbf{z}) \otimes \mathbf{z} \tag{16}$$

$$= \mathrm{E}_{\mathbf{z}\sim\mu}\,\tfrac{1}{\sigma}\mathbf{u}(\mathbf{x} + \sigma\mathbf{z}) \odot \mathbf{z} \tag{17}$$

Therefore, we obtain the following unbiased estimator: $\mathbf{g} = \frac{1}{\sigma}\mathbf{u}(\mathbf{x} + \sigma\mathbf{z}) \odot \mathbf{z}$. In terms of indices, for clarity: $\mathbf{g}_i = \frac{1}{\sigma}\mathbf{u}(\mathbf{x} + \sigma\mathbf{z})_i\mathbf{z}_i$. With this new estimator, the number of utility function evaluations per iteration is now *constant* in the number of players, rather than linear. This dramatically reduces the number of utility function evaluations when there are many players. Therefore, it makes game solving significantly more efficient in many-player games, especially when the utility function is expensive to evaluate in terms of wall time, memory, money, or other resources such as real-world experiments. We call our method *joint-perturbation simultaneous pseudo-gradient (JPSPG)*, in contrast to the classical method, *simultaneous pseudo-gradient (SPG)*. To our knowledge, this is the first work to define the concept of the pseudo-Jacobian and apply it to the estimation of the simultaneous gradient. The intuition behind our method is that we estimate the analogue of the gradient (*i.e.*, Jacobian) of the *vector*-valued function that returns utilities for all n players *simultaneously* (and then select its diagonal), rather than treating the problem as having n different, separate *scalar*-valued functions whose gradients need to be estimated separately. This saves us work, because we need just 1 rather than n different function evaluations. For clarity, we emphasize that our method does not assume or require symmetry across players. It handles general utility functions on general strategy spaces. The utility function does not need to be symmetric across players. In fact, the players' strategy spaces need not be equal. For example, player 1's parameters could be 10-dimensional, player 2's parameters could be 20-dimensional, and so on. One can use our method in combination with any gradient-based learning dynamics from the literature that are based on the simultaneous gradient. These include learning dynamics such as standard simultaneous gradient ascent, extragradient ascent [Korpelevich, 1976], optimistic gradient ascent [Popov, 1980; Daskalakis and others, 2018], and so on. In the games that we tested on, simultaneous gradient ascent was sufficient to obtain convergence to an approximate NE.

5 Experiments

We test our approach against the classical approach on several continuous-action games, in particular on many kinds of auction and on continuous Goofspiel (which can be thought of as a kind of sequential auction with budget constraints). These are many-player games where each utility function evaluation is expensive, thus highlighting the problem of interest

and showing the benefits of our method. Specifically, each utility function evaluation requires solving a linear assignment problem, solving an integer linear program, running policies over multiple steps, *etc*., all of which are expensive operations. The games we test on cover various theoretical properties of interest, including various dimensionalities for each player's observation, various dimensionalities for each player's action, single-step vs. multi-step settings, *etc*. Our experimental hyperparameters are as follows. For each experiment, we run 8 trials. In each graph, solid lines show the mean across trials, and bands show the standard error of the mean. The classical method is shown in blue, while our method is shown in orange. We use a stepsize of 10^{-4}. For the Gaussian smoothing, we use a perturbation scale σ of 10^{-1}. We use a batch size per iteration of 256. To update parameters, we use the AdaBelief optimizer [Zhuang and others, 2020]. For each player's strategy network, we use a single hidden layer of size 64, the ReLU activation function, and He initialization [He and others, 2015] for initializing the network's weights. For our code, we used Python 3.12.3, JAX 0.4.30 [Bradbury and others, 2018], Flax 0.8.5 [Heek and others, 2023], Optax 0.2.3 [DeepMind and others, 2020], Matplotlib 3.9.1 [Hunter, 2007], and SciPy 1.14.0 [Virtanen and others, 2020]. Each experiment was run individually on one NVIDIA A100 SXM4 40GB GPU. GPUs are massively parallel, and can thus potentially perform several utility function evaluations in parallel. We emphasize that we allow the baseline that we compare against, ordinary SPG, to exploit this parallelism to the maximum extent possible. To do this, we use jax's automatic tensor parallelization and vectorization capabilities. We do this to put the baseline on the strongest possible footing, and thus make the comparison as conservative as possible. Despite this, our method still significantly outperforms the baseline. Furthermore, the relative advantage of our method would be even greater on a CPU, which is a more sequential architecture, or in any situation where massive parallelism cannot be exploited for whatever reason. We estimate the exploitability of the learned strategy profile as follows. First, we compute approximate BRs for each player via reinforcement learning, specifically OpenAI ES [Salimans and others, 2017]. For this, we use the same batch size and optimizer as before. The BRs are trained for 1024 iterations. Second, as described in §3, exploitability is defined as $\Phi(s) = \sum_{i \in \mathcal{I}} (u_i(b_i, s_{-i}) - u_i(s))$, where b_i is the BR for player i to s. We estimate each occurrence of u_i in this expression by averaging over 1024 samples of game play and subsequent payoffs for player i. In the next several subsections, we present each benchmark and our experimental results for them.

Multi-item unit-demand auction. An auction is a mechanism by which a set of *items* are sold to a set of *bidders*, who have valuations for items or sets thereof. Auctions play a central role in the study of markets and are used in a wide range of real-world contexts [Krishna, 2002], such as advertising, commodities, radio spectrum allocation, real estate, and more. To evaluate their method, Bichler *et al.* [2021] used auctions as a benchmark. Here, we consider a type of multi-item auction called a *unit-demand auction*. In this auction, we have n bidders and m non-identical items. Each bidder i has a private valuation v_{ij} for each item j. Furthermore, each bidder has *unit demand*, meaning that its value for a *bundle* of items is the same as that for the maximum-value item in that bundle: $v_i(S) = \max_{j \in S} v_{ij}$, where S is a bundle of items. Housing markets are often given as an example of unit-demand preferences. This model was first studied by Shapley and Shubik [1971]. This is a special case of a *limited-demand model* with K units in which each bidder has use for at most $L < K$ units, as described in Krishna [2002, §13.4.2 and §13.5.2]. The single-unit case corresponds to $L = 1$. For our experiment, we use a prior where bidder-item valuations v_{ij} are independently sampled uniformly at random from the unit interval. Each player i submits a bid b_{ij} for each item j. To allocate items, our auction mechanism assigns items to bidders in a way that maximizes the sum of bids across players. This requires solving a *linear assignment problem*, which can be described as follows. Given a bid matrix $b \in \mathbb{R}^{n \times m}$, compute a binary assignment $x \in \{0, 1\}^{n \times m}$ that satisfies the following:

$$\text{maximize} \quad \sum_{i \in [n]} \sum_{j \in [m]} b_{ij} x_{ij} \tag{18}$$

$$\text{subject to} \quad \sum_{i \in [n]} x_{ij} \leq 1 \qquad \forall j \in [m] \tag{19}$$

$$\sum_{j \in [m]} x_{ij} \leq 1 \qquad \forall i \in [n] \tag{20}$$

That is, maximize the sum of values subject to the constraint that each item is assigned to at most one bidder and each bidder receives at most one item. The linear assignment problem was first described in a seminal paper by Kuhn [1955], who introduced a solution approach called the Hungarian method. Subsequently, various algorithms have been devised in the literature. We use the modified Jonker-Volgenant algorithm [Jonker and Volgenant, 1988] given in Crouse [2016], as implemented in the Python scientific computing library SciPy [Virtanen and others, 2020], and reimplement it in JAX [Bradbury and others, 2018]. This algorithm has a time complexity of $O(n^3)$. Figure 1 shows the exploitability over the course of training for a unit-demand auction. Both our method and the classical method attain a similar exploitability for a given iteration count, but our method is dramatically faster in terms of run time (here expressed in seconds). This is explained by the fact that the standard method requires many more evaluations of the utility function per iteration, each of which requires solving an assignment problem (which is expensive), whereas ours only requires a *single* utility function evaluation. The advantage of our method over the baseline increases as the number of players increases.

Knapsack auction. Another type of auction is a *knapsack auction*. We follow the description given in Aggarwal and Hartline [2006]. In a knapsack auction, an auctioneer auctions off space in a knapsack of known capacity C. Each player seeks to place exactly one object in the knapsack. Player i values the placement of its object in the knapsack at v_i. The valuations are private data of each respective player. Each object takes up a certain amount of space in the knapsack. Player i's object takes space c_i, and these sizes are publicly known. Thus, the c_is are public while the v_is are private. Each player submits a

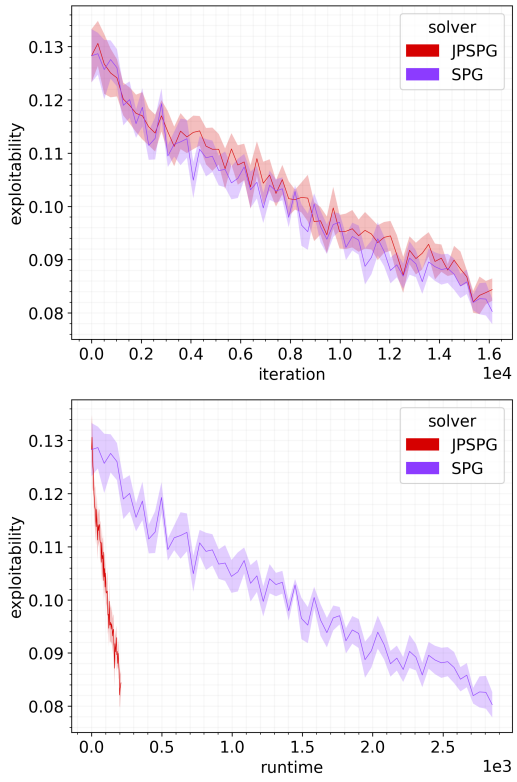

Figure 1: 20-player, 20-item unit-demand auction.

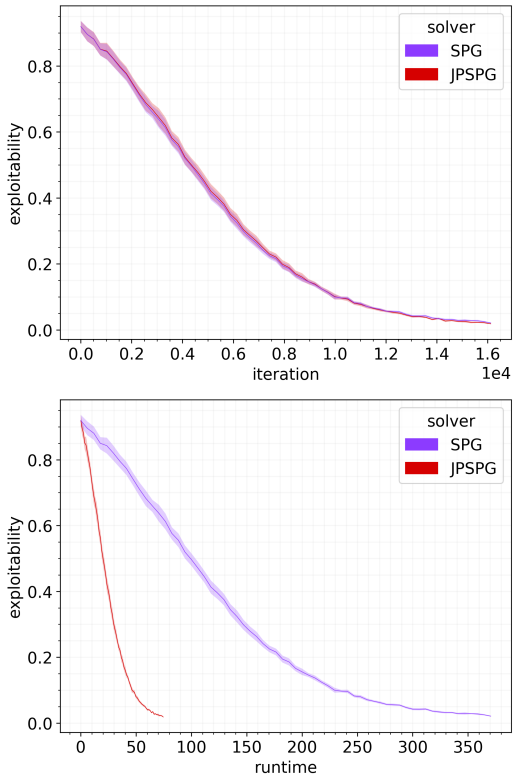

Figure 2: 20-player knapsack auction.

bid b_i. Among other words, knapsack auctions have been studied by Dütting *et al.* [2014] and Berg *et al.* [2010], who model the problem of bidding in ad auctions as a penalized multiple choice knapsack problem. As Aggarwal and Hartline [2006] note, the knapsack auction problem models several interesting applications. For example, consider running a single auction to sell advertising space on a web page over the course of a day. Suppose statistical information is available for each advertiser as to how many showings (*i.e.*, impressions) are necessary for to result in a user click-through and as well how many times the web page itself will be viewed in a day. The number of impressions necessary to generate a click-through corresponds to the c_is and the number of total views corresponds to the capacity of the knapsack, C. Each utility function evaluation requires solving an optimization problem of the following form: maximize $\mathbf{x} \cdot \mathbf{b}$ subject to $\mathbf{x} \cdot \mathbf{c} \leq C$ and $\mathbf{x} \in \{0,1\}^n$. Here, n is the number of players, $\mathbf{b}$ is the vector of stated values (bids) for each player, $\mathbf{c}$ is the corresponding vector of sizes, $\mathbf{x}$ is a binary vector indicating whether each player is included in the knapsack. Player i's final utility is $(v_i - b_i)x_i$. That is, it is the difference between their private value and their submitted bid, assuming they are included in the knapsack, and zero otherwise. This problem can be solved using *integer linear programming (ILP)*. For this, we use the `milp` function included in SciPy's library [Virtanen and others, 2020], which uses the HiGHS optimization solver [Huangfu and Hall, 2018; Hall and others, 2023]. Solving an integer program can be expensive (integer linear programming is NP-hard), so reducing the number of utility function evaluations during learning should result in a significant speedup. In our experiment, we sample the v_is and c_is from the standard uniform distribution, and sample C from the standard uniform distribution on $[0, n]$. Experimental results on the knapsack auction are shown in Figure 2. Our method requires fewer utility function evaluations per iteration and thus yields a dramatic improvement in training time for attaining the same exploitability.

Sequential auction for identical items. Consider a multi-item unit-demand auction in which identical items are sold *sequentially*, rather than simultaneously. We follow the description given in Krishna [2002, §15.1]. In this auction, K identical items are sold to $N > K$ bidders using a sequence of first-price sealed-bid auctions. Specifically, on each of K rounds, one of the items is auctioned using the first-price format, and the price at which it is sold—the winning bid—is announced. We focus on the *single-unit demand* setting, in which each bidder has use for at most one unit. Thus a bidder leaves the game once it has won an item. Each bidder has a private value v_i is that is sampled from the standard uniform distribution. On round K, a bidder's observation consists of its own private value as well as all the prices of the preceding $k-1$ rounds, $p_1, p_2, \ldots, p_{k-1}$. Results are shown in Figure 3. Our method yields a dramatic improvement in terms of the wall time required to reach a certain level of exploitability.

Continuous-action Goofspiel. Goofspiel, also known as *the game of pure strategy*, is a card game invented by mathematician Merrill Flood in the 1930s [Tucker, 1984]. This game is played with a standard 52-card deck. The cards of

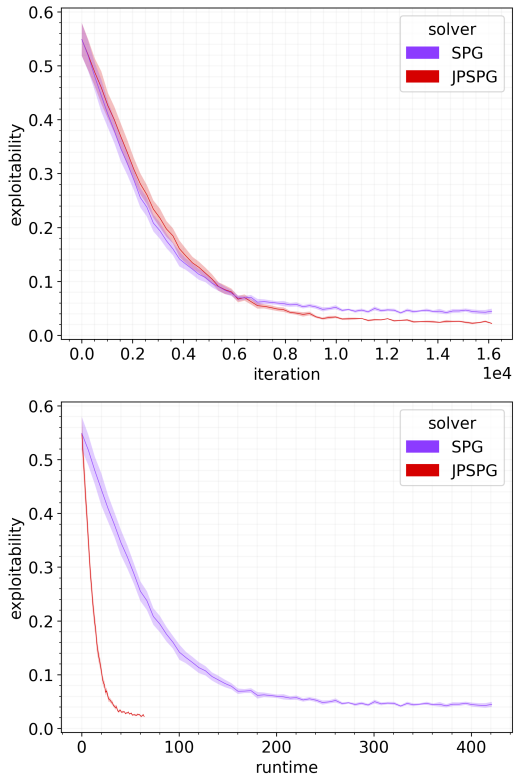

Figure 3: 20-player, 10-item sequential auction.

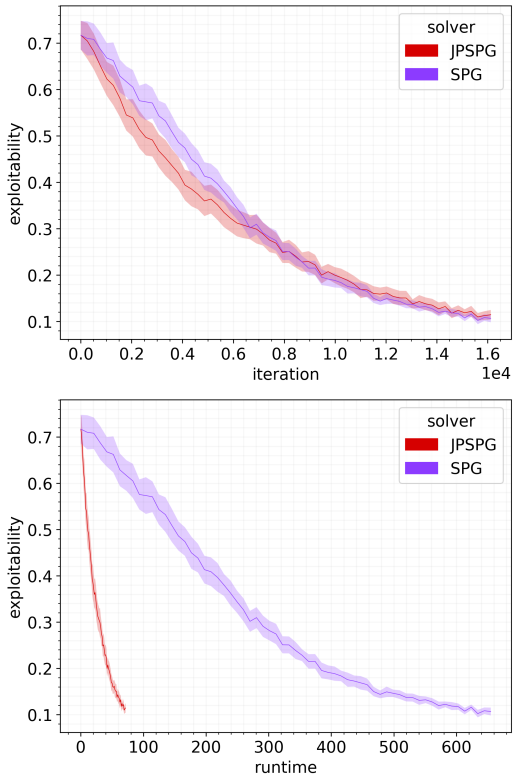

Figure 4: 20-player continuous Goofspiel.

one suit are given to one player, the cards of a second suit are given to the other player, and the cards of a third suit are shuffled and placed face down in the middle. The cards are valued from low to high as 1 (Ace), 2, 3, ..., 10, 11 (Jack), 12 (Queen), and 13 (King). A round consists of turning up the next card from the middle pile and letting the players simultaneously "bid" on this "prize" card. Players bid by choosing one of their own cards and revealing it at the same time as the other player. The player with the highest bid wins the value of the prize card. In a tie, the prize value is split between the players. All three cards are then discarded. The game ends after 13 rounds, and the winner is the player with the highest score. Because of its simple mechanics but complex strategy, Goofspiel is commonly used as an example in game theory and artificial intelligence, and has been studied extensively in the literature [Ross, 1971; Dror, 1989; Ferguson and Melolidakis, 2001; Grimes and Dror, 2013; Rhoads and Bartholdi, 2012; Lanctot *et al.*, 2014]. We consider the following *continuous-action* variant of Goofspiel. Instead of receiving a deck of discrete bids consisting of all cards of one suit, each player has a *continuous* budget that they can spend to bid on the prize card in each round. In each round, their continuous bid is subtracted from their budget. This can be thought of as a multi-round, multi-item, auction-like scenario with a budget constraint for each bidder. To allow the players to randomize over their 1-dimensional actions (the bids), we inject their strategy networks with 1-dimensional latent input noise in addition to the observation, as described in Martin and Sandholm [2023]. Figure 4 shows the exploitability over the course of training on continuous-action Goofspiel. Our method yields a significant improvement in the run time required to attain a certain level of exploitability.

6 Conclusions and Future Research

We tackled the problem of computing an approximate Nash equilibrium of a game with a black-box utility function, for which we lack access to gradients. To do this, we combined a standard gradient-based learning dynamics, simultaneous gradient ascent, with a new zeroth-order approach to computing the simultaneous gradient. Our method performs a *joint* perturbation on all players' strategies at once, rather than perturbing each one individually. This approach reduces the number of utility function evaluations per iteration from *linear* in the number of players to *constant* in the number of players. When utility function evaluation is expensive (*e.g.*, in terms of wall time, memory, or other resources), this can significantly reduce the cost of training. We compared our approach to the standard baseline on several benchmark games. Our experimental results confirm our hypothesis, namely, that our approach yields a dramatic improvement in the training time required to reach a certain level of exploitability. One possible direction for future research would be to test our method in combination with other gradient-based learning dynamics, such as extragradient and optimistic gradient ascent, in games where simultaneous gradient ascent does not converge to equilibrium. For instance, optimistic gradient ascent just needs an estimator of the simultaneous gradient as an oracle, which our method can provide.

Acknowledgments

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

References

[Aggarwal and Hartline, 2006] Gagan Aggarwal and Jason D. Hartline. Knapsack auctions. In *SODA*, 2006.

[Bäck *et al.*, 1997] Thomas Bäck, David B. Fogel, and Zbigniew Michalewicz. *Handbook of evolutionary computation*. IOP Publishing Ltd., 1997.

[Bäck, 1996] Thomas Bäck. *Evolutionary algorithms in theory and practice*. Oxford University Press, 1996.

[Balduzzi and others, 2018] David Balduzzi et al. The mechanics of n-player differentiable games. In *ICML*, 2018.

[Berahas and others, 2022] Albert S. Berahas et al. A theoretical and empirical comparison of gradient approximations in derivative-free optimization. *FoCM*, 2022.

[Berg and others, 2010] Jordan Berg et al. A knapsack-based approach to bidding in ad auctions. In *ECAI*. IOS Press, 2010.

[Bichler and others, 2021] Martin Bichler et al. Learning equilibria in symmetric auction games using artificial neural networks. *Nature Machine Intelligence*, 2021.

[Bichler *et al.*, 2023a] Martin Bichler, Max Fichtl, and Matthias Oberlechner. Computing Bayes–Nash equilibrium strategies in auction games via simultaneous online dual averaging. *Operations Research*, 2023.

[Bichler *et al.*, 2023b] Martin Bichler, Nils Kohring, and Stefan Heidekrüger. Learning equilibria in asymmetric auction games. *INFORMS Journal on Computing*, 2023.

[Bradbury and others, 2018] James Bradbury et al. JAX: Composable transformations of Python+NumPy programs, 2018.

[Crouse, 2016] David F. Crouse. On implementing 2D rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, 2016.

[Dasgupta and Maskin, 1986] P. Dasgupta and Eric Maskin. The existence of equilibrium in discontinuous economic games 1: Theory. *Review of Economic Studies*, 53:1–26, 1986.

[Daskalakis and others, 2018] Constantinos Daskalakis et al. Training GANs with optimism. In *ICLR*, 2018.

[Debreu, 1952] Gerard Debreu. A social equilibrium existence theorem. *PNAS*, 1952.

[DeepMind and others, 2020] DeepMind et al. The DeepMind JAX Ecosystem. http://github.com/google-deepmind, 2020. Accessed: 2025-05-16.

[Dror, 1989] Moshe Dror. Simple proof for Goofspiel: the game of pure strategy. *Advances in Applied Probability*, 1989.

[Duchi and others, 2015] John C. Duchi et al. Optimal rates for zero-order convex optimization: The power of two function evaluations. *IEEE Transactions on Information Theory*, 2015.

[Dütting *et al.*, 2014] Paul Dütting, Vasilis Gkatzelis, and Tim Roughgarden. The performance of deferred-acceptance auctions. In *EC*, 2014.

[Eiben and Smith, 2003] Agoston E. Eiben and James E. Smith. *Introduction to evolutionary computing*. Springer, 2003.

[Fan, 1952] Ky Fan. Fixed point and minimax theorems in locally convex topological linear spaces. *PNAS*, 1952.

[Ferguson and Melolidakis, 2001] Thomas S. Ferguson and Costis Melolidakis. Games with finite resources. *International Journal of Game Theory*, 2001.

[Fu and others, 2015] Michael C. Fu et al. *Handbook of simulation optimization*. Springer, 2015.

[Glicksberg, 1952] Irving Leonard Glicksberg. A further generalization of the Kakutani fixed point theorem, with application to Nash equilibrium points. *Proceedings of the American Mathematical Society*, 3(1):170–174, 1952.

[Goodfellow and others, 2014] Ian Goodfellow et al. Generative adversarial nets. In *NeurIPS*, 2014.

[Goodfellow and others, 2020] Ian Goodfellow et al. Generative adversarial networks. *Commun. ACM*, 2020.

[Grimes and Dror, 2013] Mark Grimes and Moshe Dror. Observations on strategies for Goofspiel. In *IEEE CIG*, 2013.

[Hall and others, 2023] Julian Hall et al. HiGHS–high performance software for linear optimization, 2023.

[He and others, 2015] Kaiming He et al. Delving deep into rectifiers: Surpassing human-level performance on ImageNet classification. In *ICCV*, 2015.

[Heek and others, 2023] Jonathan Heek et al. Flax: A neural network library and ecosystem for JAX. http://github.com/google/flax, 2023. Accessed: 2025-05-16.

[Hsieh *et al.*, 2021] Ya-Ping Hsieh, Panayotis Mertikopoulos, and Volkan Cevher. The limits of min-max optimization algorithms. In *ICML*, 2021.

[Huangfu and Hall, 2018] Qi Huangfu and J. A. Julian Hall. Parallelizing the dual revised simplex method. *Mathematical Programming Computation*, 2018.

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

[Jonker and Volgenant, 1988] Roy Jonker and Ton Volgenant. A shortest augmenting path algorithm for dense and sparse linear assignment problems. In *DGOR/NSOR*, 1988.

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

[Krishna, 2002] Vijay Krishna. *Auction Theory*. Academic Press, 2002.

[Kuhn, 1955] Harold W. Kuhn. The Hungarian method for the assignment problem. *Naval research logistics quarterly*, 1955.

[Lanctot and others, 2017] Marc Lanctot et al. A unified game-theoretic approach to multiagent reinforcement learning. In *NeurIPS*, 2017.

[Lanctot *et al.*, 2014] Marc Lanctot, Viliam Lisý, and Mark H. M. Winands. Monte Carlo tree search in simultaneous move games with applications to Goofspiel. In *Computer Games*, 2014.

[Letcher and others, 2019a] Alistair Letcher et al. Differentiable game mechanics. *JMLR*, 2019.

[Letcher and others, 2019b] Alistair Letcher et al. Stable opponent shaping in differentiable games. In *ICLR*, 2019.

[Li and Wellman, 2021] Zun Li and Michael P. Wellman. Evolution strategies for approximate solution of Bayesian games. In *AAAI*, 2021.

[Lockhart and others, 2019] Edward Lockhart et al. Computing approximate equilibria in sequential adversarial games by exploitability descent. In *IJCAI*, 2019.

[Martin and Sandholm, 2023] Carlos Martin and Tuomas Sandholm. Finding mixed-strategy equilibria of continuous-action games without gradients using randomized policy networks. In *IJCAI*, 2023.

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

[Mazumdar *et al.*, 2020] Eric Mazumdar, Lillian J. Ratliff, and S. Shankar Sastry. On gradient-based learning in continuous games. *SIAM Journal on Mathematics of Data Science*, 2020.

[Mertikopoulos and Zhou, 2019] Panayotis Mertikopoulos and Zhengyuan Zhou. Learning in games with continuous action sets and unknown payoff functions. *Mathematical Programming*, 2019.

[Metz and others, 2021] Luke Metz et al. Gradients are not all you need. *arXiv:2111.05803*, 2021.

[Milgrom and Weber, 1985] Paul Milgrom and Robert Weber. Distributional strategies for games with incomplete infromation. *Mathematics of Operations Research*, 10:619–632, 1985.

[Mohamed and others, 2020] Shakir Mohamed et al. Monte Carlo gradient estimation in machine learning. *JMLR*, 2020.

[Nash, 1950] John Nash. Equilibrium points in n-person games. *PNAS*, 36:48–49, 1950.

[Nesterov and Spokoiny, 2017] Yurii Nesterov and Vladimir Spokoiny. Random gradient-free minimization of convex functions. *FoCM*, 2017.

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

[Rechenberg and Eigen, 1973] Ingo Rechenberg and M. Eigen. *Evolutionsstrategie: Optimierung technischer systeme nach prinzipien der biologischen evolution*. Frommann-Holzboog Stuttgart, 1973.

[Rechenberg, 1978] Ingo Rechenberg. Evolutionsstrategien. In *Simulationsmethoden in der medizin und biologie*. Springer, 1978.

[Rhoads and Bartholdi, 2012] Glenn C. Rhoads and Laurent Bartholdi. Computer solution to the game of pure strategy. *Games*, 2012.

[Rosen, 1965] J. Ben Rosen. Existence and uniqueness of equilibrium points for concave n-person games. *Econometrica*, 1965.

[Ross, 1971] Sheldon M. Ross. Goofspiel—the game of pure strategy. *Journal of Applied Probability*, 8(3):621–625, 1971.

[Salimans and others, 2017] Tim Salimans et al. Evolution strategies as a scalable alternative to reinforcement learning. *arXiv:1703.03864*, 2017.

[Schwefel, 1977] Hans-Paul Schwefel. *Numerische optimierung von computer-modellen mittels der evolutionsstrategie*. Birkhäuser Basel, 1977.

[Shamir, 2017] Ohad Shamir. An optimal algorithm for bandit and zero-order convex optimization with two-point feedback. *JMLR*, 2017.

[Shapley and Shubik, 1971] Lloyd S. Shapley and Martin Shubik. The assignment game I: The core. *IJGT*, 1971.

[Timbers and others, 2022] Finbarr Timbers et al. Approximate exploitability: Learning a best response. In *IJCAI*, 2022.

[Tucker, 1984] Albert W. Tucker. The Princeton mathematics community in the 1930s: transcript number 11, 1984.

[Virtanen and others, 2020] Pauli Virtanen et al. SciPy 1.0: Fundamental algorithms for scientific computing in python. *Nature Methods*, 2020.

[Walton and Lisy, 2021] Michael Walton and Viliam Lisy. Multi-agent reinforcement learning in OpenSpiel: A reproduction report. *arXiv:2103.00187*, 2021.

[Wierstra and others, 2008] Daan Wierstra et al. Natural evolution strategies. In *IEEE CEC*, 2008.

[Wierstra and others, 2014] Daan Wierstra et al. Natural evolution strategies. *JMLR*, 2014.

[Willi and others, 2022] Timon Willi et al. COLA: Consistent learning with opponent-learning awareness. In *ICML*, 2022.

[Yi and others, 2009] Sun Yi et al. Stochastic search using the natural gradient. In *ICML*, 2009.

[Yu-Guan and others, 2019] Yu-Guan et al. On the convergence of single-call stochastic extra-gradient methods. In *NeurIPS*, 2019.

[Zhuang and others, 2020] Juntang Zhuang et al. AdaBelief optimizer: Adapting stepsizes by the belief in observed gradients. *NeurIPS*, 2020.