Evolution Strategies for Approximate Solution of Bayesian Games

Zun Li and Michael P. Wellman
University of Michigan, Ann Arbor
{lizun, wellman}@umich.edu

Abstract

We address the problem of solving complex Bayesian games, characterized by high-dimensional type and action spaces, many (> 2) players, and general-sum payoffs. Our approach applies to symmetric one-shot Bayesian games, with no given analytic structure. We represent agent strategies in parametric form as neural networks, and apply natural evolution strategies (NES) (Wierstra et al. 2014) for deep model optimization. For pure equilibrium computation, we formulate the problem as bi-level optimization, and employ NES in an iterative algorithm to implement both inner-loop best response optimization and outer-loop regret minimization. In simple games including first- and second-price auctions, it is capable of recovering known analytic solutions. For mixed equilibrium computation, we adopt an incremental strategy generation framework, with NES as strategy generator producing a finite sequence of approximate best-response strategies. We then calculate equilibria over this finite strategy set via a model-based optimization process. Both our pure and mixed equilibrium computation methods employ NES to efficiently search for strategies over the function space, given only black-box simulation access to noisy payoff samples. We experimentally demonstrate the efficacy of all methods on two simultaneous sealed-bid auction games with distinct type distributions, and observe that the solutions exhibit qualitatively different behavior in these two environments.

1 Introduction

Bayesian games (Harsanyi 1967) model incomplete information by encoding uncertainty over opponents' hidden information in terms of beliefs over player *types*. Types are drawn from a common knowledge prior distribution. Each player is informed of its own type, and decides its action strategically as a function of this private information. This framework provides a standard model for many economic games, such as auctions, and has informed the design of many real-world market-based systems, including mechanisms for online advertising. In this paper, we focus on one-shot, symmetric Bayesian games (SBG), where both type space and action space are subsets of multidimensional Euclidean spaces (McAdams 2003). Multi-object auctions (Christodoulou, Kovács, and Schapira 2008) provide a canonical example of this game class, with types as parameter vectors defining valuations for sets of goods, and strategies mapping such types to bids for these goods.

A *Bayesian-Nash equilibrium* (BNE) is a configuration of strategies that is stable in the sense that no player can increase the expected value of its outcome by deviating to another strategy. As our games are symmetric, it is natural to seek symmetric BNE, and to exploit symmetric representations for computational purposes. A classical approach (Milgrom and Weber 1982; McAfee and McMillan 1987) for deriving pure symmetric BNE is to solve a differential equation for a fixed point of an analytical best response mapping. However for many SBGs of interest, analytic solution is hindered by irregular type distributions and nonlinear payoffs, and dimensionality of type and action spaces.

Instead of seeking analytical solutions, we formulate the problem of computing BNE as a high-dimensional optimization, and present computational results that bridge classical economic theories and modern AI techniques. We parameterize agent strategies as neural networks, therefore approximate the original functional strategy space as a high-dimensional vector space of network weights. Our algorithmic approach is based on *natural evolution strategies* (NES) (Wierstra et al. 2014), a black-box optimization algorithm based on stochastic search in the parameter space. NES has been recently shown a competitive optimization technique for non-smooth environments like those often tackled by RL (Salimans et al. 2017). We consider it well suited for our purpose of equilibrium computation, as SBGs typically exhibit large payoff discontinuities (Reny 1999).

We present methods for computing both pure and mixed BNE (PBNE and MBNE). With player symmetry, computing PBNE can be cast as a minimax optimization. Our proposed algorithm employs one NES process to implement an approximate best response operator, and another NES to search for an approximate fixed-point of this operator via regret minimization. This approach to some extent mirrors the classical analytic approach mentioned above.

Finding stable profiles in pure strategies may be difficult for games endowed with complex strategic landscape. Indeed the existence of PBNE can be assured only with certain assumptions such as regularity of type distributions (Milgrom and Weber 1985). To search for approximate MBNE, we construct finite approximations of the original infinite game by discretizing the strategy space (Dasgupta and Maskin 1986), and consider mixed equilibria of these restricted games. We resort to an incremental strategy generation framework (McMahan, Gordon, and Blum 2003) to implement this approach, where NES again is employed as strategy generator producing a finite sequence of approximate best-response strategies. We then extract strategic information of a restricted strategy set by regressing a finite game model via supervised learning, and calculate a mixed equilibrium of this learned game as the solution.

2 Related Work

Although there is a rich literature in economics on characterizing properties of equilibria in Bayesian games (Milgrom and Weber 1985; Athey 2001; McAdams 2003; Reny 2011), the problem of solving for BNE computationally is less explored. It is known that computing PBNE for discrete strategy space (where both type and action are discrete) is NP-complete (Conitzer and Sandholm 2008), and computing PBNE for simultaneous Vickrey auctions is even PP-hard (Cai and Papadimitriou 2014). These seem to discourage attempts to develop general solvers for Bayesian games.

However efforts directed to specific classes of Bayesian games has not been inhibited, and over the years substantial results have been obtained both theoretically and empirically. Some work focuses on the two-player case: Reeves and Wellman (2004) derived analytical best response for Bayesian game with piece-wise linear payoff functions, and Ceppi, Gatti, and Basilico (2009) gave experimental results on the support-enumeration approach. Other research addresses symmetric Bayesian games with more players. Vorobeychik and Wellman (2008) used self-play implemented by simulated annealing to find PBNE in SBGs. We likewise employ a form of stochastic search, but avoiding the limitations of self-play (Balduzzi et al. 2019) and with methods that extend to high-dimensional settings. Wellman, Sodomka, and Greenwald (2017) tackled simultaneous Vickrey auctions by applying empirical game-theoretical analysis on a set of hand-crafted strategies solving for MBNE. They identified effective heuristic bidding strategies, however these strategies generally require computation exponential in the number of goods.

Rabinovich et al. (2013) considered Bayesian games with continuous type space and discrete action space, and provided both theoretical convergence results as well as experimental validations for the fictitious play algorithm. Recent work (Wang, Shen, and Zuo 2020) gave computational results for first-price auctions with discrete type space and continuous action space. For a more general class of multi-item auction, works (Christodoulou, Kovács, and Schapira 2008; Dütting and Kesselheim 2017) studied algorithms of theoretical interest for combinatorial Vickrey auctions, and Bosshard et al. (2017) presented experimental results on applying grid search to solve the Local-Local-Global auctions. For succinct game models, works (Singh, Soni, and Wellman 2004) and (Jiang and Leyton-Brown 2010) formulated the notion of incomplete information in graphical games and action-graph games, together with corresponding computational methods, respectively. Armantier, Florens, and Richard (2008) approximated the Bayesian game to be solved by a sequence of constrained games, and designed numerical methods by solving approximate partial differential equations for equilibrium computation. Nevertheless, none of the above works developed a computational method that scales with player numbers as well as the dimensions of type space and action space.

Solving Bayesian games is also relevant to real-world applications such as advertising auctions. Chawla and Hartline (2013) theoretically proved the uniqueness of symmetric BNE for a class of rank-based auction formats including generalized first-price auctions. Gomes and Sweeney (2014) derived a closed-form expression of the symmetric PBNE for generalized second-price auctions.

In work independent and concurrent with ours, Heidekrüger et al. (2021) also employ NES to compute BNE. Whereas we solve the game from a central perspective, they model the result of decentralized learning agents. Other key differences are that we assume and exploit symmetry, and develop methods to compute MBNE as well as PBNE. They in contrast consider a general asymmetric setting and focus on finding PBNE.

3 Preliminaries

3.1 Bayesian Games

One-shot, simultaneous-move symmetric Bayesian games provide a standard model for common strategic scenarios such as auctions (Krishna 2009). Formally, an SBG consists of $(\mathscr{N}, \mathscr{T}, \mathscr{A}, \mathcal{P}, u)$, with $\mathscr{N} = \{1, \dots, N\}$ being the set of agents, $\mathscr{T}$ the set of types and $\mathscr{A}$ the set of actions. We focus on the case where $\mathscr{T}$ and $\mathscr{A}$ are compact subsets of $\mathbb{R}^T$ and $\mathbb{R}^A$, with T and A the dimensions of type and action space, respectively.

In a play of the SBG, agents simultaneously and privately observe their types, drawn i.i.d. from a common knowledge distribution $t \sim \mathcal{P}$, then independently choose their respective actions conditional on this private information. More formally, an agent's *pure strategy* s is a deterministic mapping from the type space to the action space $s : \mathscr{T} \to \mathscr{A}$. We denote the set of all such mappings as $\mathscr{S}$. For computational purposes in this paper we represent a pure strategy as a neural network. Though the space of multilayer perceptrons of a fixed architecture may not perfectly coincide with $\mathscr{S}$, in effect we can represent any strategy to a close approximation due to the expressive power of deep models.

For playing action $a_n = s_n(t_n)$, agent n receives a real-valued payoff $u(a_n, \boldsymbol{a}_{-n} \mid t_n) : \mathscr{A} \times \mathscr{A}^{N-1} \times \mathscr{T} \to \mathbb{R}$, which depends on its own type and action along with the profile of other agents' actions $\boldsymbol{a}_{-n} \in \mathscr{A}^{N-1}$. Player symmetry entails that one's payoff value is permutation-invariant to the opponents' actions: $u(a_n, a_{\pi(1)}, \dots, a_{\pi(N)} \mid t_n) = u(a_n, a_1, \dots, a_N \mid t_n)$ for any permutation π of order $N-1$. For a given strategy profile $(s_1, \dots, s_N)$, the *ex interim* (EI) payoff is the expected payoff marginalized over opponents' types: $u(s_n, \boldsymbol{s}_{-n} \mid t_n) \triangleq \mathbb{E}_{\boldsymbol{t}_{-n} \sim \mathcal{P}^{N-1}}[u(s_n(t_n), s_1(t_1), \dots, s_N(t_N)) \mid t_n)]$, and the *ex ante* (EA) payoff averages over one's own type randomness $u(s_n, \boldsymbol{s}_{-n}) = \mathbb{E}_{t_n \sim \mathcal{P}}[u(s_n, \boldsymbol{s}_{-n} \mid t_n)]$.

A *mixed strategy* $\sigma \in \Delta(\mathscr{S})$ defines a probability measure over the pure strategy space and allows one to make stochastic decisions.[1] For a given mixed strategy profile $(\sigma_1, \ldots, \sigma_N)$, the EI and EA payoffs for agent n of type t_n are $u(\sigma_n, \boldsymbol{\sigma}_{-n} \mid t_n) \triangleq \mathbb{E}_{s_n \sim \sigma_n, \boldsymbol{s}_{-n} \sim \boldsymbol{\sigma}_{-n}}[u(s_n, \boldsymbol{s}_{-n} \mid t_n)]$ and $u(\sigma_n, \boldsymbol{\sigma}_{-n}) \triangleq \mathbb{E}_{t_n \sim \mathcal{P}}[u(\sigma_n, \boldsymbol{\sigma}_{-n} \mid t_n)]$.

For SBGs we are often interested in one's payoff given all $N-1$ others adopt the same strategy. We write $u(\sigma', \boldsymbol{\sigma})$ for the EA payoff of an agent choosing strategy σ' while the rest choose σ. For strategy σ and pure strategy s, we call $u(s, \boldsymbol{\sigma})$ the *deviation payoff* to pure strategy s against opponent mixture σ, or the *fitness* of s under σ.

3.2 Bayesian-Nash Equilibrium

In a symmetric profile, every agent adopts the same strategy. For symmetric games, we generally prefer to find solutions in symmetric profiles, and these are guaranteed to exist in a variety of settings (Nash 1951; Cheng et al. 2004; Chawla and Hartline 2013; Hefti 2017). The *symmetric regret* of strategy σ is given by $\textsc{Regret}(\sigma) \triangleq \max_{s \in \mathscr{S}} u(s, \boldsymbol{\sigma}) - u(\sigma, \boldsymbol{\sigma})$. Our goal is to search for σ that minimizes loss $\textsc{Regret}(\sigma)$, in other words, a symmetric profile where each agent approximately best responds to the others. If $\textsc{Regret}(\sigma) \leq \epsilon$, we call σ an (EA) ϵ-BNE. For the pure case (*i.e.*, $\sigma \in \mathscr{S}$) we more specifically label σ an (EA) ϵ-PBNE, and for the mixed case an (EA) ϵ-MBNE.

3.3 Black-Box Games

While many classic Bayesian games possess closed-form solutions (Krishna 2009), many others are intractable via analytical reasoning. To develop computational methods for general SBGs of interest, therefore, we adopt a more universal formulation and represent the game by a black-box payoff oracle $\mathcal{O} : \mathscr{S}^N \times \Omega \to \mathbb{R}^N$. In this black-box (also termed *simulation-based*) setting, the basic operation is a query by the game analyst, who submits a joint pure strategy $\boldsymbol{s} \in \mathscr{S}^N$ to $\mathcal{O}$, and receives in return a payoff vector $\boldsymbol{u} \in \mathbb{R}^N$ recording payoffs for each agent realized through agent-based simulation (with a random seed $\omega \in \Omega$). We assume the analyst is aware of $\mathscr{N}, \mathscr{T}, \mathscr{A}$ and player symmetry, but neither the type distribution $\mathcal{P}$ nor a direct representation of the payoff function u. Therefore the goal is to find approximate equilibria given only stochastic black-box simulation access to the game.

3.4 Natural Evolution Strategies

NES (Wierstra et al. 2014) belongs to a family of black-box optimization algorithms called *evolution strategies* (ES) that can be viewed as abstract versions of natural selection processes. The goal of NES is to maximize a fitness function $F(\boldsymbol{\theta})$ with respect to neural network weights $\boldsymbol{\theta}$, given only stochastic black-box access to its point values. Instead of directly optimizing F, the idea is to construct gradient estimators of a *Gaussian smoothing objective*

[1]A *behavioral strategy* $b : \mathscr{T} \to \Delta(\mathscr{A})$ is a mapping from the type space to the space of probability measures on the action space. It turns out that mixed and behavioral strategies can each be shown equivalent to the other under certain conditions (Milgrom and Weber 1985).

Algorithm 1: Natural Evolution Strategies

```
  Input: Black-box function $F$, hyperparameters $J, \alpha, \nu$
  Output: Approximate maximum $\boldsymbol{\theta}$ of $F$
1 Algorithm NES $(F, J, \alpha, \nu)$
2     Initialize $\boldsymbol{\theta}$;
3     for $i = 1, 2, \ldots$ do
4         Sample $\boldsymbol{\varepsilon}_1, \ldots, \boldsymbol{\varepsilon}_J \sim \mathcal{N}(0, \boldsymbol{I})$;
5         $\forall j, r_{j+} \leftarrow F(\boldsymbol{\theta} + \nu\boldsymbol{\varepsilon}_j), r_{j-} \leftarrow F(\boldsymbol{\theta} - \nu\boldsymbol{\varepsilon}_j); r_j \leftarrow r_{j+} - r_{j-}$;
6         $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} + \alpha\frac{1}{J\nu}\sum_j r_j\boldsymbol{\varepsilon}_j$;
7     end
8     return $\boldsymbol{\theta}, F(\boldsymbol{\theta})$;
```

$\mathbb{E}_{\boldsymbol{\varepsilon} \sim \mathcal{N}(0, \boldsymbol{I})}[F(\boldsymbol{\theta} + \nu\boldsymbol{\varepsilon})]$ with bandwidth hyperparameter ν, and apply stochastic gradient ascent accordingly. As a technique for optimizing neural networks, NES has been shown competitive with other back-propagation based methods of deep RL (Salimans et al. 2017). For the purpose of equilibrium computation, we employ NES as a subroutine to optimize deep strategies for different choices of fitness functions.

We elaborate our version of NES in Algorithm 1. Initialized with network weights $\boldsymbol{\theta}$, on each iteration NES constructs a finite difference approximation of the gradient through random search over the space of $\boldsymbol{\theta}$, and updates the deep model towards the direction of higher expected fitness values. To construct such gradients, NES first samples a population of noisy vectors $\boldsymbol{\varepsilon}_1, \ldots, \boldsymbol{\varepsilon}_J$ in the parameter space from a normal distribution. For each of such $\boldsymbol{\varepsilon}$-s, it perturbs $\boldsymbol{\theta}$ into a pair of antithetic variables $(\boldsymbol{\theta} \pm \nu\epsilon)$, and evaluates the corresponding fitness values as $r_{\pm}$. This ensures that the quantity $v = \frac{1}{2\nu}(r_+ - r_-)\boldsymbol{\varepsilon}$ is an unbiased estimator of $\nabla_{\boldsymbol{\theta}}\mathbb{E}_{\boldsymbol{\varepsilon}}[F(\boldsymbol{\theta} + \nu\boldsymbol{\varepsilon}))]$, and the stochastic gradient is constructed as the empirical average of all these v-s enabling gradient ascent updates.

4 Computing Pure Equilibrium via Minimax Optimization

4.1 Overview

In this section, we demonstrate how to compute PBNE in SBGs by exploiting game structure and the power of NES as a black-box optimizer. We showcase that due to player symmetry the problem of calculating a symmetric PBNE is equivalent to computing a minimax equilibrium in a reduced two-player zero-sum game, and therefore design a co-evolutionary algorithm (Algorithm 2) implemented by two NES processes to solve this bi-level optimization problem. In this approach, we employ one NES to compute an approximate best response and a regret value for a given pure strategy in the inner loop, and another NES in the outer loop to minimize this approximate regret function over the pure strategy space. We next develop the details of the algorithm.

4.2 A Minimax Formulation

Recall that computing PBNE is equivalent to minimizing $\textsc{Regret}$ over the pure strategy space, which can be formu-

Algorithm 2: Minimax-NES for PBNE

```
    Input: Payoff Oracle $\mathcal{O}$, hyperparameters
           $J_1, J_2, \alpha_1, \alpha_2, \nu_1, \nu_2$
    Output: Approximate PBNE $s_{\boldsymbol{\theta}}$
1   Function MinusRegret($\boldsymbol{\theta}$)
2       $V \leftarrow \mathcal{O}(s_{\boldsymbol{\theta}}, \boldsymbol{s_\theta})$;
3       $\boldsymbol{\theta'}, DEV \leftarrow$ NES($\mathcal{O}(\cdot, \boldsymbol{s_\theta}), J_1, \alpha_1, \nu_1$);
4       return $V - DEV$;
5   Algorithm MiniMax()
6       return NES(MinusRegret, $J_2, \alpha_2, \nu_2$)
```

lated as

$$\min_{s \in \mathscr{S}} \text{REGRET}(s) = \min_{s \in \mathscr{S}} \max_{s' \in \mathscr{S}} u(s', \boldsymbol{s}) - u(s, \boldsymbol{s}) \quad (1)$$

i.e., we can view this optimization as a two-player zero-sum game: a primary agent who aims to choose s that minimize REGRET, and an adversary who selects the best response s' against s to implement the REGRET. This concise formulation exploits player symmetry such that the $N-1$ opponents can be abstracted as one agent when computing a symmetric equilibrium (Hefti 2017; Vadori et al. 2020).

4.3 Inner Loop: NES as the Best Response Optimizer

To optimize objective (1), we need to acquire a best response strategy as well as a maximum deviation value for a given pure strategy $s_{\boldsymbol{\theta}}$ parameterized by $\boldsymbol{\theta}$. To achieve this we utilize NES and the payoff oracle $\mathcal{O}$ to optimize a neural strategy $s_{\boldsymbol{\theta'}}$ as an approximate best response, as marked in line 3 of Algorithm 2. Since to compute a best response against strategy $s_{\boldsymbol{\theta}}$ is to maximize the deviation payoff function $u(\cdot, \boldsymbol{s_\theta})$, we need to provide such stochastic value queries to this deviation function, by refactoring the payoff oracle as $\mathcal{O}(\cdot, \boldsymbol{s_\theta})$. $\mathcal{O}(\cdot, \boldsymbol{s_\theta})$ takes a deep strategy $s_{\boldsymbol{\theta'}}$ as input and outputs its stochastic fitness values under $s_{\boldsymbol{\theta}}$. This can be implemented by controlling one agent n adopting $s_{\boldsymbol{\theta'}}$ with the rest $s_{\boldsymbol{\theta}}$, sampling this joint profile many times and taking the average payoff of n across these samples of different type realizations as the final output. Then we pass this wrapped refactored payoff oracle to NES for automatic best response optimization.

4.4 Outer Loop: NES as the Regret Minimizer

The inner loop just described provides with an approximate maximum deviation value for any pure strategy, DEV in line 3 of Algorithm 2. We can hence estimate REGRET$(s_{\boldsymbol{\theta}})$ by first evaluating $u(s_{\boldsymbol{\theta}}, \boldsymbol{s_\theta})$ as V via payoff oracle $\mathcal{O}$, shown in line 2 of Algorithm 2, and set the regret estimator as $DEV - V$. To compute an approximate PBNE, we employ another NES in the outer loop that minimizes such surrogate regret function over the pure strategy space. To accomplish that we wrap a black-box function `MinusRegret` that returns $V - DEV$ for a given deep strategy, which is delivered to the outer-loop NES for regret minimization. Hence the interaction between the agent in the outer loop and the adversary in the inner loop each implemented by an NES process enables an efficient algorithm to compute symmetric PBNE in SBGs.

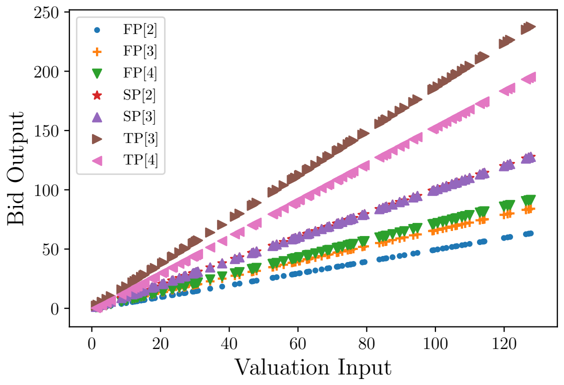


Figure 1: Point plots for strategy functions learned by minimax-NES in games with known analytical solutions

4.5 Results for Games with Analytical Solutions

As a preliminary experiment, we investigate the strategies learned by minimax-NES in those SBGs with known analytical solutions. The algorithmic specifications are detailed in Section 6.1 and the appendix (Li and Wellman 2021). Specifically, we consider[2] N-player unit-item first-price auctions, second-price auctions and third-price auctions, denoted as *FP*$[N]$ and *SP*$[N]$ and *TP*$[N]$. It is known that for uniform valuation distribution $U[0, \bar{t}]$ the canonical PBNE is $s(t) = (\frac{N-1}{N})t$ for *FP*$[N]$, $s(t) = t$ for *SP*$[N]$ and $s(t) = (\frac{N-1}{N-2})t$ for *TP*$[N]$. We plot the relation between the input valuation and output bid of the strategies produced by minimax-NES in these environments in Figure 1, where we let $\bar{t} = 128$. The results show that minimax-NES learns these canonical equilibria. Notice that classical derivations for these PBNE imposed a variety of constraints on the solutions, such as monotonicity (Riley and Samuelson 1981), while our method only implicitly relies on the expressiveness and differentiability of deep models yet still learned these canonical PBNE. We hypothesize these equilibria are the unique solutions representable by deep neural networks.

5 Computing Mixed Equilibrium via Incremental Strategy Generation

5.1 Overview

In this section, we consider computing mixed equilibria for SBGs by interleaving strategy exploration and equilibrium calculation. This approach fits a generic paradigm for reasoning games with complex strategy space called incremental strategy generation (ISG). ISG maintains a finite restricted set of pure strategies S that discretizes the intractable strategic landscape, and iteratively enlarges this set via best responses to explore rational regions of the game. A mixture over such representative set of strategies is computed in each iteration, serving as a quality mixed equilibrium of the full game. ISG

[2]In the appendix we also include experiments for games with more complex solutions such as all-pay auctions, but we did not successfully reproduce their canonical analytic results.

Algorithm 3: Incremental Strategy Generation

```
Input: Payoff Oracle $\mathcal{O}$. Meta-solver $MS$.
       Hyperparameters $J, \alpha, \nu$ ;
Output: A finite strategy set $S$, a mixed strategy $\sigma$
        over $S$
1 Initial strategy set $S = \{s_0\}$, a singleton distribution
    $\sigma$ with $\sigma(s_0) = 1$;
2 for $i = 1, 2, \ldots$ do
3     $\sigma \leftarrow MS(\mathcal{O}, S, \sigma)$;
4     $s', DEV \leftarrow \texttt{NES}(\mathcal{O}(\cdot, \boldsymbol{\sigma}), J, \alpha, \nu)$;
5     $S \leftarrow S \cup \{s'\}$;
6 end
```

had been shown great success in domains of two-player zero-sum stochastic games (McMahan, Gordon, and Blum 2003), security games (Jain et al. 2011; Bosansky et al. 2015; Wang et al. 2019; Wright, Wang, and Wellman 2019), extensive-form games (Bosansky et al. 2014), multiagent RL (Lanctot et al. 2017) and multiplayer team games (Zhang and An 2020).

We next elaborate the usage of ISG in our SBG context, as diagrammed in Algorithm 3. The ISG framework consists of two components: a meta-solver (MS) and a best response oracle (BR). Initialized as a singleton strategy set, on each iteration of ISG, MS takes the restricted strategy set S resulted from the previous iteration as input and outputs a probability mixture σ over S. The mixture is expected to constitute a quality equilibrium when none of the pure strategies is dominant over the others. Common meta-solvers include self-play (which puts all mass on the last strategy), fictitious play (uniformly mixing over S), replicator dynamics (that computes an NE in the restricted game) and other more developed ones (Balduzzi et al. 2019; Muller et al. 2020).

After MS had output σ, BR generates a new strategy into S that is a(n) (approximate) best response against opponent mixture σ. The functionality of BR is to explore and include strategies in regions where agents are more likely to exhibit rational behavior, producing a more robust strategy population and reinforcing the mixture quality calculated by MS in the next iteration. In our version of ISG we use NES to implement BR (line 4 of Algorithm 3) where in Algorithm 1 we let the fitness function F be $\mathcal{O}(\cdot, \boldsymbol{\sigma})$, being consistent with Section 4.

We next discuss our choices of meta-solvers.

5.2 Fictitious Play

The first choice of meta-solver is fictitious play (FP), which puts a uniform mixture on current strategy set S. FP-type algorithms had been shown great success in a variety of games beyond the version for tabular normal-form games (Rabinovich et al. 2013; Heinrich, Lanctot, and Silver 2015). We believe FP is a competitive baseline in our problem, since a uniform mixture may capture the diversity of the strategic landscape to some extent.

5.3 Nash Equilibrium

Our second choice of meta-solver is to output a Nash equilibrium of the restricted game with support S, which also reflects the double-oracle algorithm (McMahan, Gordon, and Blum 2003). We adopt projected replicator dynamics (RD) (Lanctot et al. 2017) to reach an equilibrium. Suppose we are given the exact payoff function u, in each iteration with opponent mixture σ, RD will update the probability mass of each pure strategy s by an amount proportional to the its present *aggregated deviation value* $\sigma(s)(u(s, \boldsymbol{\sigma}) - u(\sigma, \boldsymbol{\sigma}))$, after which an L2 projection (Wang and Carreira-Perpinán 2013) onto $\Delta(S)$ is operated to ensure it remain on the probability simplex. Therefore by evolving the mixture towards strategies with higher fitness values, the converged mixed strategy is expected to be stable at the end of the dynamics, and the new strategies are anticipated to be generated in rational regions.

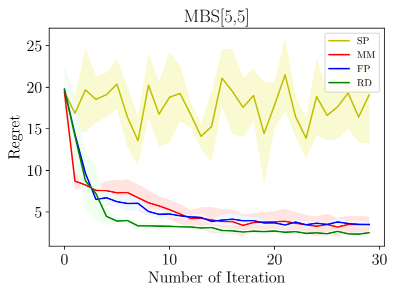
Model Learning However in our problem, the issue for replicator dynamics or other Nash-solvers is that we do not have the exact evaluation of deviation payoff $u(s, \boldsymbol{\sigma})$ but only its stochastic query values, which may require a significant number of samples to control the variance for each update, and inhibit Nash-solvers from converging to stable solutions. To reduce sample complexity and computational intractability, we adopt a supervised model-learning approach (Vorobeychik, Wellman, and Singh 2007; Wiedenbeck, Yang, and Wellman 2018; Sokota, Ho, and Wiedenbeck 2019; Li and Wellman 2020) that regresses the pure-strategy payoff function of this finite restricted game model, and further provides RD with deviation estimation for mixture computation. The key here is to exploit a succinct game representation. Notice that for finite symmetric games with M strategies, it suffices to record $u(s, f_1, \ldots, f_M)$, where s is the pure strategy chosen by the principal agent and f_m counts the fraction of its opponents choosing strategy m. Then by assuming the ground-truth payoff function varies smoothly with strategy counters, we can use a function approximator to learn such succinct representation by extracting correlations among different pure-strategy payoff entries.

More concretely, we regress the restricted game with $|S| = M$ by sampling a dataset from the payoff oracle and learning a value network $\hat{u} : \Delta(S) \to \mathbb{R}^M$ as the empirical game model. The input training feature of $\hat{u}$ is a vector of strategy counters, each dimension counting the fraction of other players choosing the corresponding strategy, and the output targets are the payoffs for each of the M pure strategies when adopted by the principal agent. Then to estimate the deviation payoffs under opponent mixture σ, we directly set σ as the input to the value network, and obtain M values as deviation payoff estimations. We consider this approach applies to finite symmetric games with many players, since the strategy counters are expected to conform the probability mixture due to law of large number.

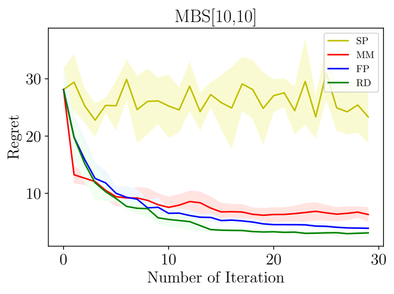
6 Experiments

6.1 Setups

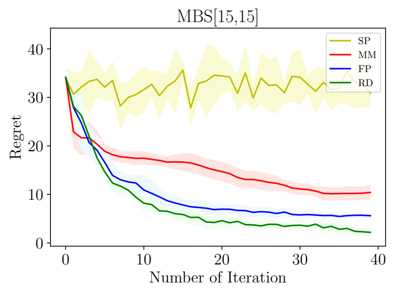
Algorithmic Configurations We represent each pure strategy as a two-layer perceptron with hidden node size 32



(a)



(b)



(c)

Figure 2: Results for market-based scheduling environments

for each layer separated by ReLU units. The hyperparameters for NES are tuned as follows. We fixed population size $J = 4 + 3\lfloor \log d \rfloor$ as the default setting adopted by Wierstra et al. (2014), where d is the number of parameters of the deep model. For every fitness function we apply a grid search to select the best bandwidth ν and learning rate α within certain ranges, to maximize the performance of the resulted NES. We adopt Adam optimizer (Kingma and Ba 2014) to automatically adjust the learning rate initialized with α during the stochastic gradient ascent process. In addition to the vanilla NES described in Section 3.4, our implementation employs a fitness-shaping trick (Wierstra et al. 2014) to enhance the robustness of the optimization process.

For black-box functions $\mathcal{O}(\cdot, \boldsymbol{s})$ and $\mathcal{O}(\cdot, \boldsymbol{\sigma})$, each query we run an agent-based simulation for 5000 times and take the corresponding average payoff values as the outputs. In the model-learning part of RD, for each empirical game model with $|S| = M$, we draw 2000 vectors of strategy counters from a Dirichlet distribution as training features, and for each feature we calculate M pure strategy payoffs via 500 Monte-Carlo samples as training targets. We adopt a two-layer network with hidden node size 32 each for game model learning.

Please refer to supplementary material (Li and Wellman 2021) for more details on implementation and hyperparameter selection.

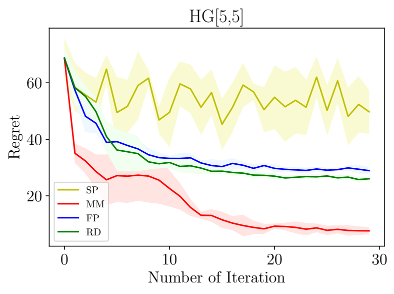
Environments We test our methods on two simultaneous sealed-bid auctions: market-based scheduling (MBS) (Reeves et al. 2005) and homogeneous-good auctions (HG) (Wellman, Sodomka, and Greenwald 2017), both of which are SBGs with multidimensional type space and action space possessing no analytical solution. We elaborate the game mechanisms as follows.

Market-Based Scheduling In an MBS environment, an agent's objective is to acquire enough number of slots to fulfil its task through strategic bidding. More specifically, for an MBS with K slots, each agent is rendered a type vector with dimension $K + 1$: an integer $\Lambda \in [1, K]$ specifying the total number of slots needed, and a valuation vector $v \in \mathbb{R}^K$, where v_k is the valuation realized if it acquired its Λ-th slot at the k-th auction. If it had not obtained Λ slots in the end then its valuation is 0. Λ is drawn from an exponential distribution, while v_k are constructed by first independently drawing K numbers uniformly from $[0, 50]$, and reordering the values to satisfy a non-increasing constraint. Each slot is allocated to the highest bidder and priced via a second-price payment rule, so an action is a bidding vector $b \in \mathbb{R}^K$ specifying the bids for each of the K auctions. The payoff of one agent is the difference between its realized valuation and payment. This auction format exhibits both complementarity and substitutability.

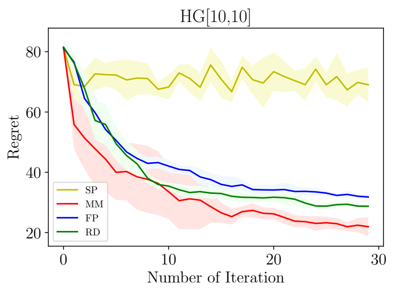
Homogeneous-Good Auctions In an HG of K goods, K Vickery auctions are simultaneously operated for selling homogeneous items. Each agent is rendered a type vector $t \in \mathbb{R}^K$, where t_k is the marginal valuation of acquiring the k-th good. t_1 is drawn uniformly from $[0, 128]$, with t_k drawn uniformly from $[0, t_{k-1}]$ for $k > 1$. An action is a vector of K bids for each of the K auctions. This auction format exhibits perfect substitutability.

In our experimental presentation we use $MBS[N, K]$ and $HG[N, K]$ to denote MBS and HG with N agents and K goods, respectively.

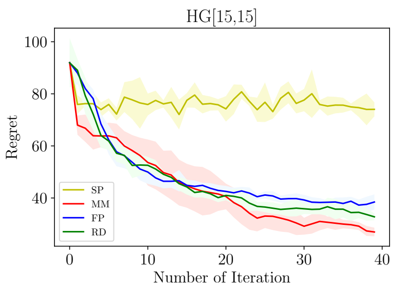
Evaluation Metric and Methods We compare four methods in all of our experiments: self-play and minimax that compute PBNE, with fictitious play and replicator dynamics solving for MBNE. Each method runs for $MAXT$ trials of different random seeds, with each trial continues for $MAXI$ iterations. Each of such iteration generates a new pure strategy, resulting in a restricted pure strategy set $\overline{S}$ with $|\overline{S}| = 4 \times MAXT \times MAXI$ totally. Denote $\sigma_{AL,i,t}$ the strategy output (which could be either pure or mixed) algorithm AL produces at iteration i of trial number t. Since we do not have an exact best response oracle, we estimate the regret of $\sigma_{AL,i,t}$ in the full game as $\max_{s \in \overline{S}} u(s, \boldsymbol{\sigma_{AL,i,t}}) - u(\sigma_{AL,i,t}, \boldsymbol{\sigma_{AL,i,t}})$, instead of using NES to compute an approximate best response. Then we measure the performance of an algorithm by plotting its regret curve averaged across different trials. We consider this evaluation approach utilizes the results the most and largely reduces the bias introduced by NES as an approximate best response operator. Furthermore to reduce statistical bias of taking maximum during regret calculations, we first estimate each of the $|\overline{S}|$ deviation values via 5000 Monte-Carlo samples, select the 10 strategies with highest scores, calculate the deviation values for these 10 again via 50000 samples and take the maximum of these as the final maximum deviation



(a)



(b)



(c)

Figure 3: Results for homogeneous good environments

estimation. We next specify the strategy output for each of the four algorithms.

Self-Play (SP): SP is also called iterated best response. At each iteration SP outputs a best response to the pure strategy of the previous iteration.

Minimax (MM): For Algorithm 2, we define one iteration as one step of gradient ascent (line 6 of Algorithm 1) in the outer loop, which outputs a pure strategy.

Fictitious Play (FP): On each iteration FP outputs a uniform mixture over S.

Replicator Dynamics (RD): RD outputs a Nash mixture for a restricted game, which is a mixed strategy.

6.2 Results

We test our methods on MBS and HG environments of 5, 10, and 15 agents with the same number of goods. Each experiment runs for 5 trials. The results are shown in Figures 2 and 3.

Our first observation is that self-play performs poorly in nearly all of our experiments. It tends to cycle in the strategic landscape and shows no explicit improvement from the initial strategy. This contrasts with the results of Vorobeychik and Wellman (2008), which were obtained for a different game with much lower-dimensional strategy space. In our context, we find that best responses cycle around the intransitive regions of the game.

Our second observation is that the RD meta-solver generally outperforms FP. FP outputs a uniform mixture of all strategies explored, which may include strategies generated early on that are not effective against those learned later. Equilibrium based methods, in contrast, will ignore strategies once they are no longer part of a solution. However the observed performance gaps between FP and RD are relatively small. The equilibrium-based method entails much greater computational cost than FP (detailed in supplementary material) due to model learning and training data sampling, thus FP could be an advantageous computational method in certain settings.

We are particularly interested in the performance of minimax-NES, which exhibited qualitatively different behavior in our two experimental environments. In MBS, minimax-NES produces strategies that are generally less robust than the mixed equilibria, whereas in HG it is able to reach pure strategies that surpass both FP and RD in terms of stability. This advantage is especially pronounced in environment $HG[5, 5]$. We attribute the difference to the distinct game characteristics induced by MBS and HG valuations. MBS environments produce more complex strategic landscapes, for (1) its type representation involves Γ as an integer which prevents an atomless type distribution that is usually required for the existence of pure equilibria (Milgrom and Weber 1985), and (2) complementarity in valuations makes strategy outcomes more sensitive to other-agent behavior. Therefore it may be difficult for minimax-NES to reach a pure strategy region that is robust globally. While for HG the types are vectors of marginal valuations which we hypothesize a representation more amenable for the deep models to extract strategic information.

Comparison to Hand-Crafted Strategies We compare the performance of bidding strategies derived by our methods against state-of-the-art strategies for simultaneous sealed-bid auctions that optimize the bid vector based on probabilistic price predictions. Specifically, our reference is to hand-crafted bidding strategies based on *self-confirming* predictions, defined as probability distributions over prices that result when all bidders optimize with respect to these distributions. Such strategies, which we denote SC, were found in an study employing empirical game-theoretical analysis by Wellman, Sodomka, and Greenwald (2017) to be effective against a broad range of bidding strategies from prior literature.

The idea of a self-confirming price prediction (SCPP) is that it summarizes opponents' behavior near an equilibrium. Assuming all the $N - 1$ opponents employ some strong heuristic bidding strategies, SC computes an approximate best response to the resulted SCPP, using a hand-crafted bid-generation methods. Since the calculation involves searching for an optimal bundle with respect to predicted price, effectively enumerating all possible bundles, SC generally takes exponential time in the number of goods. This also makes it more expensive to evaluate an action, compared with one forward pass of the neural strategies.

The version of SC we adopt in the following experiments is LocalBid initialized with ExpectedMU64 (Wellman, Sodomka, and Greenwald 2017). Due to the exponential computational cost we are only able to test on $MBS[5, 5]$ and $HG[5, 5]$ environments.

First we compare the robustness of SC equilibrium to our

Instance	SP	MM	FP	RD	SC
$MBS[5,5]$	19.1	3.51	3.47	2.51	5.30
$HG[5,5]$	49.7	7.62	28.9	26.0	11.0

Table 1: Regret of SC compared with other methods within $\overline{S}$

Instance	SP	MM	FP	RD
$MBS[5,5]$	0.0	3.46	0.74	1.71
$HG[5,5]$	0.45	3.05	0.0	0.0

Table 2: Regret of our methods with respect to SC

methods, by measuring their regret values with our restricted set $\overline{S}$, as shown in Table 1. The performances of our methods are measured by taking the outputs of the last iteration, averaged across different runs. The results showed that in $MBS[5,5]$ the SC equilibrium surpasses only self-play in terms of stability, while in $HG[5,5]$ it is more robust than methods SP, FP, RD, but is still inferior to MM. This demonstrates that our methods produce comparably or even more quality equilibria than SC globally.

Next we investigate more on the strategic dependencies among these strategies, by testing the robustness of our methods against SC. We here measure the regret values of our methods to SC, shown in Table 2. The results shows that SC especially exploits MM in both environments. This validates that MM produces near-equilibrium strategies and SC exploits such class of strategies according to its designs. We further consider the results suggest a strategic dependency among different strategies and illustrate that there exist no single strategy that can outperform the others in all cases.

7 Conclusion

In contrast to classic works in auction theory that seek analytical solutions, we formulate the problem of computing BNE as an empirical optimization problem. Scaling computational methods to the dimensions of types and actions as well as player number immediately calls for tractable solution representation and efficient optimization techniques. We found that combining the expressive power of deep models with NES as a black-box optimization technique effectively supports solution of a general class of complex symmetric Bayesian games.

Our pure equilibrium computation method, minimax-NES, parallels the classical analytical approach and could be regarded as a high-dimensional generalization of the global convergence method of Vorobeychik and Wellman (2008). By exploiting player symmetry, the method employs NES for best-response optimization and regret minimization. Here two NES processes are integrated in one algorithm to reach minimax solutions. Our mixed equilibrium computation method, ISG, employs NES as both best-response optimizer and strategy generator. We tested our methods on two simultaneous-auction games with qualitatively different properties, and found that the mixed equilibria showed lower regret on environments with more complex strategic landscape, while the solutions output by minimax-NES appeared to be more robust in games with smoother topology.

Our methods rely on the power of NES as a function searching tool in the strategy space. But since it is difficult to reach the true optimum in such function space we hypothesize certain biases could be introduced by the algorithm NES itself. Our evaluation approach was designed to mitigate this by measuring regret with respect to all the pure strategies we generated across experiment runs. In future work, we are interested in testing other optimization alternatives including genetic algorithms (Such et al. 2017) and comparing them with NES.

Acknowledgments

This work was supported in part by the US Army Research Office under MURI W911NF-18-1-0208.

References

Armantier, O.; Florens, J.-P.; and Richard, J.-F. 2008. Approximation of Nash equilibria in Bayesian games. *Journal of Applied Econometrics* 23(7): 965–981.

Athey, S. 2001. Single crossing properties and the existence of pure strategy equilibria in games of incomplete information. *Econometrica* 69(4): 861–889.

Balduzzi, D.; Garnelo, M.; Bachrach, Y.; Czarnecki, W. M.; Perolat, J.; Jaderberg, M.; and Graepel, T. 2019. Open-ended learning in symmetric zero-sum games. In *Thirty-Sixth International Conference on Machine Learning*, 434–443.

Bosansky, B.; Jiang, A. X.; Tambe, M.; and Kiekintveld, C. 2015. Combining compact representation and incremental generation in large games with sequential strategies. In *Twenty-Ninth AAAI Conference on Artificial Intelligence*, 812–818.

Bosansky, B.; Kiekintveld, C.; Lisy, V.; and Pechoucek, M. 2014. An exact double-oracle algorithm for zero-sum extensive-form games with imperfect information. *Journal of Artificial Intelligence Research* 51: 829–866.

Bosshard, V.; Bünz, B.; Lubin, B.; and Seuken, S. 2017. Computing Bayes-Nash equilibria in combinatorial auctions with continuous value and action spaces. In *Twenty-Sixth International Joint Conference on Artificial Intelligence*, 119–127.

Cai, Y.; and Papadimitriou, C. 2014. Simultaneous Bayesian auctions and computational complexity. In *Fifteenth ACM Conference on Economics and Computation*, 895–910.

Ceppi, S.; Gatti, N.; and Basilico, N. 2009. Computing Bayes-Nash equilibria through support enumeration methods in Bayesian two-player strategic-form games. In *2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology*, volume 2, 541–548.

Chawla, S.; and Hartline, J. D. 2013. Auctions with unique equilibria. In *Fourteenth ACM Conference on Electronic Commerce*, 181–196.

Cheng, S.-F.; Reeves, D. M.; Vorobeychik, Y.; and Wellman, M. P. 2004. Notes on equilibria in symmetric games. In *Sixth*

International Workshop on Game Theoretic and Decision Theoretic Agents.

Christodoulou, G.; Kovács, A.; and Schapira, M. 2008. Bayesian combinatorial auctions. In *International Colloquium on Automata, Languages, and Programming*, 820–832.

Conitzer, V.; and Sandholm, T. 2008. New complexity results about Nash equilibria. *Games and Economic Behavior* 63(2): 621–641.

Dasgupta, P.; and Maskin, E. 1986. The existence of equilibrium in discontinuous economic games, I: Theory. *Review of Economic Studies* 53(1): 1–26.

Dütting, P.; and Kesselheim, T. 2017. Best-response dynamics in combinatorial auctions with item bidding. In *Twenty-Eighth Annual ACM-SIAM Symposium on Discrete Algorithms*, 521–533.

Gomes, R.; and Sweeney, K. 2014. Bayes-Nash equilibria of the generalized second-price auction. *Games and Economic Behavior* 86: 421–437.

Harsanyi, J. C. 1967. Games with incomplete information played by Bayesian players, Part I. The basic model. *Management Science* 14(3): 159–182.

Hefti, A. 2017. Equilibria in symmetric games: Theory and applications. *Theoretical Economics* 12(3): 979–1002.

Heidekrüger, S.; Sutterer, P.; Kohring, N.; Fichtl, M.; and Bichler, M. 2021. Equilibrium learning in combinatorial auctions: Computing approximate Bayesian Nash equilibria via pseudogradient dynamics. In *AAAI-21 Workshop on Reinforcement Learning in Games*.

Heinrich, J.; Lanctot, M.; and Silver, D. 2015. Fictitious self-play in extensive-form games. In *Thirty-Second International Conference on Machine Learning*, 805–813.

Jain, M.; Korzhyk, D.; Vaněk, O.; Conitzer, V.; Pěchouček, M.; and Tambe, M. 2011. A double oracle algorithm for zero-sum security games on graphs. In *Tenth International Conference on Autonomous Agents and Multi-Agent Systems*, 327–334.

Jiang, A. X.; and Leyton-Brown, K. 2010. Bayesian action-graph games. In *Twenty-Third International Conference on Neural Information Processing Systems*, 991–999.

Kingma, D. P.; and Ba, J. 2014. Adam: A method for stochastic optimization. Preprint arXiv:1412.6980.

Krishna, V. 2009. *Auction theory*. Academic Press.

Lanctot, M.; Zambaldi, V.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Pérolat, J.; Silver, D.; and Graepel, T. 2017. A unified game-theoretic approach to multiagent reinforcement learning. In *Thirty-First International Conference on Neural Information Processing Systems*, 4190–4203.

Li, Z.; and Wellman, M. P. 2020. Structure learning for approximate solution of many-player games. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2119–2127.

Li, Z.; and Wellman, M. P. 2021. Evolution strategies for approximate solution of Bayesian games: Supplementary material. Avaliable at https://rezunli96.github.io/.

McAdams, D. 2003. Isotone equilibrium in games of incomplete information. *Econometrica* 71(4): 1191–1214.

McAfee, R. P.; and McMillan, J. 1987. Auctions and bidding. *Journal of Economic Literature* 25(2): 699–738.

McMahan, H. B.; Gordon, G. J.; and Blum, A. 2003. Planning in the presence of cost functions controlled by an adversary. In *Twentieth International Conference on Machine Learning*, 536–543.

Milgrom, P. R.; and Weber, R. J. 1982. A theory of auctions and competitive bidding. *Econometrica* 1089–1122.

Milgrom, P. R.; and Weber, R. J. 1985. Distributional strategies for games with incomplete information. *Mathematics of Operations Research* 10(4): 619–632.

Muller, P.; Omidshafiei, S.; Rowland, M.; Tuyls, K.; Perolat, J.; Liu, S.; Hennes, D.; Marris, L.; Lanctot, M.; Hughes, E.; et al. 2020. A generalized training approach for multiagent learning. In *Eighth International Conference on Learning Representations*.

Nash, J. 1951. Non-cooperative games. *Annals of Mathematics* 286–295.

Rabinovich, Z.; Naroditskiy, V.; Gerding, E. H.; and Jennings, N. R. 2013. Computing pure Bayesian-Nash equilibria in games with finite actions and continuous types. *Artificial Intelligence* 195: 106–139.

Reeves, D. M.; and Wellman, M. P. 2004. Computing best-response strategies in infinite games of incomplete information. In *Twentieth Conference on Uncertainty in Artificial Intelligence*, 470–478.

Reeves, D. M.; Wellman, M. P.; MacKie-Mason, J. K.; and Osepayshvili, A. 2005. Exploring bidding strategies for market-based scheduling. *Decision Support Systems* 39(1): 67–85.

Reny, P. J. 1999. On the existence of pure and mixed strategy Nash equilibria in discontinuous games. *Econometrica* 67(5): 1029–1056.

Reny, P. J. 2011. On the existence of monotone pure-strategy equilibria in Bayesian games. *Econometrica* 79(2): 499–553.

Riley, J. G.; and Samuelson, W. F. 1981. Optimal auctions. *The American Economic Review* 71(3): 381–392.

Salimans, T.; Ho, J.; Chen, X.; Sidor, S.; and Sutskever, I. 2017. Evolution strategies as a scalable alternative to reinforcement learning. Preprint arXiv:1703.03864.

Singh, S.; Soni, V.; and Wellman, M. 2004. Computing approximate Bayes-Nash equilibria in tree-games of incomplete information. In *Fifth ACM Conference on Electronic Commerce*, 81–90.

Sokota, S.; Ho, C.; and Wiedenbeck, B. 2019. Learning deviation payoffs in simulation-based games. In *Thirty-Third AAAI Conference on Artificial Intelligence*, 2173–2180.

Such, F. P.; Madhavan, V.; Conti, E.; Lehman, J.; Stanley, K. O.; and Clune, J. 2017. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning. Preprint arXiv:1712.06567.

Vadori, N.; Ganesh, S.; Reddy, P.; and Veloso, M. 2020. Calibration of shared equilibria in general sum partially observable Markov games. In *Thirty-Third International Conference on Neural Information Processing Systems*, 14118–14128.

Vorobeychik, Y.; and Wellman, M. P. 2008. Stochastic search methods for Nash equilibrium approximation in simulation-based games. In *Seventh International Joint Conference on Autonomous Agents and Multi-Agent Systems*, 1055–1062.

Vorobeychik, Y.; Wellman, M. P.; and Singh, S. 2007. Learning payoff functions in infinite games. *Machine Learning* 67(1-2): 145–168.

Wang, W.; and Carreira-Perpinán, M. A. 2013. Projection onto the probability simplex: An efficient algorithm with a simple proof, and an application. Preprint arXiv:1309.1541.

Wang, Y.; Shi, Z. R.; Yu, L.; Wu, Y.; Singh, R.; Joppa, L.; and Fang, F. 2019. Deep reinforcement learning for green security games with real-time information. In *Thirty-Third AAAI Conference on Artificial Intelligence*, volume 33, 1401–1408.

Wang, Z.; Shen, W.; and Zuo, S. 2020. Bayesian Nash equilibrium in first-price auction with discrete value distributions. In *Nineteenth International Conference on Autonomous Agents and Multi-Agent Systems*, 1458–1466.

Wellman, M. P.; Sodomka, E.; and Greenwald, A. 2017. Self-confirming price-prediction strategies for simultaneous one-shot auctions. *Games and Economic Behavior* 102: 339–372.

Wiedenbeck, B.; Yang, F.; and Wellman, M. P. 2018. A regression approach for modeling games with many symmetric players. In *Thirty-Second AAAI Conference on Artificial Intelligence*, 1266–1273.

Wierstra, D.; Schaul, T.; Glasmachers, T.; Sun, Y.; Peters, J.; and Schmidhuber, J. 2014. Natural evolution strategies. *Journal of Machine Learning Research* 15(1): 949–980.

Wright, M.; Wang, Y.; and Wellman, M. P. 2019. Iterated deep reinforcement learning in games: History-aware training for improved stability. In *Twentieth ACM Conference on Economics and Computation*, 617–636.

Zhang, Y.; and An, B. 2020. Converging to team-maxmin equilibria in zero-sum multiplayer games. In *Thirty-Seventh International Conference on Machine Learning*.

A More Implementation Details

All of our deep models and training processes are implemented by PyTorch, with ReLU as the nonlinear activation function and Xavier-uniform as the initialization method. Furthermore to prevent the action values explode to infinity we truncate the output of the last linear layer to $[0, \bar{t}]$, where $\bar{t}$ is the upper bound on the action values. $\bar{t} = 50$ for MBS and $\bar{t} = 128$ for HG. The value networks are trained with SmoothL1Loss with batch normalization.

For implementation of RD, we combine a support enumeration approach with the vanilla RD to search for a low-support Nash equilibrium. More concretely, for current support $\underline{S} \in S$, we test whether there is a strategy in the remaining restricted strategy set $S \backslash \underline{S}$ with beneficial deviation. If the beneficial deviation of a strategy is larger than some threshold, we will add this strategy to support $\underline{S}$, and recalculate the Nash mixture focusing on $\underline{S}$.

Algorithm 4: RD with Support Exploration

```
Input: Finite strategy set $S$, current support $\underline{S} \in S$,
       payoff oracle $\mathcal{O}$, Nash-Solver RD, initial
       mixture $\sigma$, threshold $E$
Output: Support $\underline{S} \in S$, a mixture $\sigma$ over $\underline{S}$
1 repeat
2     for $s \in S\backslash\underline{S}$ do
3         if $\mathcal{O}(s, \boldsymbol{\sigma}) > \mathcal{O}(\sigma, \boldsymbol{\sigma}) + E$ then
4             $\underline{S} \leftarrow \underline{S} \cup s$;
5         end
6     end
7     $\sigma \leftarrow RD(\mathcal{O}, \underline{S})$;
8 until Convergence;
```

B Time Scales

We test the computational cost for all of our methods, as shown in Table 3. The computational time is measured on 2x 3.0 GHz Intel Xeon Gold 6154 with 4 cores and RAM 32GB.

Instance	SP[s]	MM[s]	FP[s]	RD[s]
$MBS[5,5]$	8.49	272	11.0	240
$MBS[10,10]$	25.6	780	28.8	1015
$MBS[15,15]$	52.7	2331	62.0	2767
$HG[5,5]$	7.22	212	10.7	486
$HG[10,10]$	21.9	829	27.3	2166
$HG[15,15]$	53.5	1052	61.1	4015

Table 3: Average computational time per iteration

C Hyperparameter Selection

The detailed process for tuning hyperparameters is as follows. For minimax-NES we fix $\alpha_1 = 0.01, \nu_1 = 0.1$ as we found the selections differs a little. We tune α_2 by applying a grid search in range $\{0.01, \ldots, 0.1\}$, and ν_2 in $\{0.01, \ldots, 0.1\}$. The configuration we employed in our experiments is provided in Table 4.

Instance	α_2	ν_2
$MBS[5,5]$	0.05	0.05
$MBS[10,10]$	0.04	0.04
$MBS[15,15]$	0.02	0.08
$HG[5,5]$	0.05	0.1
$HG[10,10]$	0.06	0.04
$HG[15,15]$	0.05	0.07

Table 4: Hyperparameter selection for minimax-NES

D More Experiments

We test minimax-NES on unit-item N-player all-pay auction, denoted as $AP[N]$. For uniform distribution $U[0, \bar{t}]$,

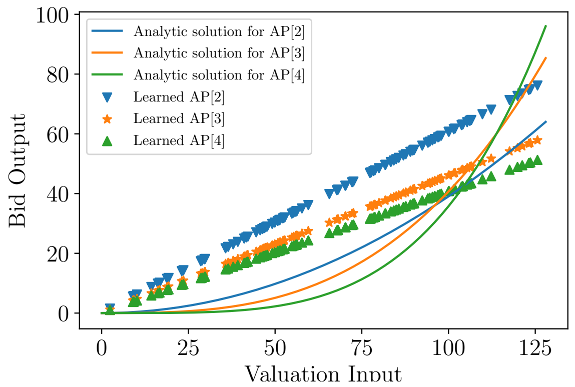


Figure 4: Point plot for strategy functions learned by minimax-NES in games with known analytical solutions

the canonical solution for all-pay auction is (Krishna 2009): $s(t) = (\frac{N-1}{N})\frac{t^N}{\bar{t}^{N-1}}$. As shown in figure 4, we find minimax-NES tends to converge to linear solutions under current architecture, where we let $\bar{t} = 128$.