Equilibrium Computation for Auction Games via Multi-Swarm Optimization

Nils Kohring*[1], Carina Fröhlich[1], Stefan Heidekrüger[1], Martin Bichler[1]

[1]Department of Computer Science
Technical University of Munich
85748 Garching, Germany
*nils.kohring@in.tum.de

Abstract

Auctions are prevalent in practice, yet their Bayesian Nash equilibria (BNE) are poorly understood. For realistic markets with multiple items and value interdependencies, the BNE often turn out to be intractable systems of partial differential equations. In the general case, finding equilibria is known to be computationally hard. We use neural networks and self-play to learn equilibrium bid functions. Unfortunately, the ex-post utility functions are discontinuous. Therefore, particle swarm optimization is employed as a gradient-free training method for the neural networks. We observe fast and robust convergence to approximate BNE in a wide variety of auctions while matching a state-of-the-art gradient-based method in some auction environments and outperforming it in others. The approach is evaluated on a variety of symmetric and asymmetric auction games and provides a robust method for learning in environments with discontinuous utility or loss functions.

Introduction

For many real-world markets, computing a Bayesian Nash equilibrium is challenging or requires restricting assumptions. Recently, numerical methods have been proposed that are able to approximate equilibria under mild assumptions. A promising direction is the framework of *online learning*, where players are treated as oblivious entities that face a repeated decision process with a priori unknown game rules and outcomes. This article is concerned with learning BNE in multiple auction formats, including challenging settings with symmetric and asymmetric prior type distributions. Asymmetry results in a heterogeneous set of agents and an uneven playing field with some agents at a strategic disadvantage that might hinder the learning of optimal behavior. Only recently, an algorithm called *neural pseudogradient ascent* (NPGA) has been introduced by Bichler et al. (2021). It models bidders' strategies as neural networks which follow the gradients of the bidders' utility functions. Unfortunately, ex-post utility functions are discontinuous and gradients cannot be computed directly (e.g., via backpropagation). NPGA relies on "pseudogradients" calculated via evolutionary strategies. Unfortunately, computing these pseudogradients is considerably more expensive compared with standard deep learning optimization techniques. There is also no apparent reason to rely on evolutionary strategies only.

This paper introduces a new gradient-free training method based on *particle swarm optimization* (PSO) as an alternative to pseudogradients computed via evolutionary strategies as they have been used in NPGA. The PSO algorithm maintains a swarm of solution candidates (also called particles) where each of them searches for an optimum. All particles adjust their trajectories based on their experience and some information exchange with other particles. For our purposes, each agent is represented by a separate swarm that tries maximizing its utility. These swarms repeatedly interact in self-play, which constitutes a standard reinforcement learning procedure. Applying PSO requires few assumptions about the problem and most crucially it does not assume differentiability nor convexity. A further advantage of PSO over evolutionary strategies is that it searches the parameter space more widely by maintaining a population of alternative candidate solutions. This process mitigates the risk of sticking to a non-optimal local equilibrium and enables the discovery of multiple solutions at once. It also does not require pretraining as is typically conducted when evolutionary strategies are used to compute pseudogradients.

The performance of PSO is evaluated and we provide a thorough experimental comparison to evolutionary strategies on some benchmark problems. Our empirical results show that gradient dynamics based on PSO are at least as efficient, and often converge faster compared with pseudogradients based on evolutionary strategies. The results might also be of relevance for other (multi-agent) learning tasks where the loss function is discontinuous.

Related Literature

This section provides an overview of prior work on equilibrium computation and on PSO that has been largely focused on static single-objective optimization.

Equilibrium Computation

One approach to equilibrium computation is to discretize the game by transforming the type and action

spaces into discrete ones. Then Nash's equilibrium existence theorem holds and many algorithms from traditional game theory are applicable. However, even when faced with an exact equilibrium in this finite game, one cannot tell whether this equilibrium is in any sense an ϵ-BNE of the original auction game (Cai and Papadimitriou 2014).

As a common means for equilibrium computation, gradient dynamics in games have been studied in evolutionary game theory and multi-agent learning. A common result for many settings and algorithms is that gradient-based learning rules do not necessarily converge to equilibria and may exhibit cycling behavior (Hsieh, Mertikopoulos, and Cevher 2021). However, these rules often achieve no-regret properties and thus converge to weaker coarse correlated equilibria (Hartline, Syrgkanis, and Tardos 2015).

Earlier approaches on finding equilibria in auctions were usually setting specific and relied on reformulating the equation as a differential equation, then solving this equation analytically or numerically (Vickrey 1961; Krishna 2009; Ausubel and Baranov 2019). Rabinovich et al. (2013) studied best-response dynamics on mixed strategies in auctions with finite action spaces. Recently, Bosshard et al. (2020) proposed a method to find BNE in combinatorial auctions that relies on smoothed best-response dynamics applicable to Bayesian games. Bichler et al. (2021) proposed a different method based on gradient dynamics. Conceptually, our approach shares large parts with NPGA but uses a more natural technique to train neural networks in a non-differentiable environment. Another line of research is not learning behavior from the bidders' point of view but learning the mechanism from the auctioneer's perspective (Dütting et al. 2019).

Particle Swarm Optimization

Generally, PSO falls into evolutionary computing and was introduced by Kennedy and Eberhart in 1995. Unlike traditional optimization techniques such as gradient-based methods, PSO tends to be more adaptable as no strict assumptions on the objective are required. It also tends to be resilient to premature convergence to local optima.

There are a some approaches for deploying PSO to reinforcement learning (Piperagkas et al. 2012; Hein et al. 2017), but we are not aware of any applications to competitive multi-agent reinforcement learning. Many proposals exist for adapting PSO for multi-objective optimization, and we refer the interested reader to the surveys of Reyes-Sierra, Coello et al. (2006) or Engelbrecht (2005). A crucial difference between multi-objective optimization and learning in games is that the players in a game have competing interests and collusion among them is not desired. When colluding, agents may be able to exceed the utility they would receive in equilibrium. This renders most of the proposed multi-objective PSO algorithms inapplicable in our scenario as they rely, to a large extent, on cooperative information sharing between the sub-objectives.

Convergence of PSO

The analysis of swarm convergence is twofold: first, one would hope for all particles agreeing on a single position at some point, and, second, this position should correspond to an actual optimum of the optimization task.

Based on results from dynamic system theory, Trelea (2003) and Van Den Bergh et al. (2002) analyzed convergence of PSO in the standard single-objective problem for a simplified deterministic variant. Both derived necessary and sufficient criteria for particle convergence under the rather strong *stagnation assumption*, which requires particles' personal and neighborhood best positions to be fixed. For the original stochastic version Jiang, Luo, and Yang (2007) analyzed convergence based on the expectations and variances of the particles' positions. Kadirkamanathan, Selvarajah, and Fleming (2006) provide sufficient conditions for asymptotic stability of particle dynamics based on Lyapunov stability analysis and the concept of passive systems on the stagnation assumption only. Moreover, assuming stagnation and well-defined expectations and variances, Bonyadi and Michalewicz (2015) derived criteria for stability. Just recently Cleghorn and Engelbrecht (2018) showed convergence for a large class of PSO variants while dropping the stagnation assumption.

Another direction that comes closer to the multi-agent learning setting is the investigation of convergence properties of multi-objective PSO which, however, is much less sophisticated. Chakraborty et al. (2011) show convergence behavior of Pareto-based multi-objective PSO algorithms via conditions on the inertia weight and acceleration coefficients. However, the analysis lacks an answer to the swarm distribution in the parameter space within the Pareto optimal solution set. Scheepers, Engelbrecht, and Cleghorn (2019) also provide a similar structured stability analysis for their multi-guide PSO algorithm.

Problem Statement

Bayesian Games and Combinatorial Auctions

We will consider auctions modeled as one-shot, simultaneous-move *Bayesian games* which are defined as quintuples $G = (\mathcal{I}, \mathcal{A}, \mathcal{V}, F, u)$. $\mathcal{I} = \{1, \dots, n\}$ describes the set of agents. Meanwhile, $\mathcal{V} = \mathcal{V}_1 \times \cdots \times \mathcal{V}_n$ is the set of type profiles, with $\mathcal{V}_i$ being the set of types available to agent $i \in \mathcal{I}$. $\mathcal{A} = \mathcal{A}_1 \times \cdots \times \mathcal{A}_n$ is the set of possible action profiles and $F\colon \mathcal{V} \to [0, 1]$ defines a joint prior probability distribution over type profiles that is assumed to be common knowledge among all agents.

We consider *sealed-bid combinatorial auctions* on $\mathcal{M} = \{1, \dots, m\}$ items. Bidders can be interested in any combination of items, thus the valuations and actions span over the space of possible bundles of items $\mathcal{K}$ which is of size 2^m. Therefore, each agent's type $v_i \in \mathcal{V}_i$ is given by a vector of *private valuations* over bundles, i.e., $v_i = (v_{i,k})_{k \in \mathcal{K}}$. Let $\mathcal{V}_i = \mathcal{A}_i = \mathbb{R}_+^k$. In

each game, all agents observe their types, $v \sim F$, where each agent i is informed of their type $v_i \in \mathcal{V}_i$ only.

Bidders then submit actions, called *bids* b_i on bundles of these items that indicate their demand. These bids are chosen according to some *strategy* or *bid function* $\beta_i : \mathcal{V}_i \to \mathcal{A}_i$ that maps individual valuations to a corresponding bid. We denote by Σ_i the resulting function space of all possible strategies of bidder i and by $\Sigma = \prod_i \Sigma_i$ the space of possible joint strategies. Note that the spaces Σ_i are infinite-dimensional.

The auctioneer collects these bids and applies some *auction mechanism* that determines the auction outcome: The mechanism maps bids to an allocation and payments. First, the allocation $x \in \mathcal{K}^n$ states for each bidder i the bundle $x_i \in \mathcal{K}$ he or she receives, s.t. each item $m \in \mathcal{M}$ is allocated at most once. Second, the payments that the bidders must pay to the auctioneer are denoted by $p \in \mathbb{R}^n$.

Each bidder's *ex-post* utility function is then determined based on all bidders' actions but only on their own type: $u_i : \mathcal{A} \times \mathcal{V}_i \to \mathbb{R}$. Commonly, one assumes quasi-linear utility functions given by $u_i : \mathcal{V}_i \times \mathcal{A} \to \mathbb{R}$,

$$u_i(v_i, b_i, b_{-i}) = v_i(x_i) - p_i, \tag{1}$$

that is, the utility of each player is given by how much he or she values the allocated goods minus the price he or she must pay for them.

Throughout this paper, we denote by the index $-i$ a profile of types, actions, or strategies for all agents but agent i.

Equilibria in Bayesian Games

Nash equilibrium (NE) is the central equilibrium solution concept in game theory. In an NE, no agent has an incentive to deviate unilaterally, given all other agents' equilibrium strategies. BNE extends this notion to incomplete-information games, calculating the expected utility $\overline{u}$ over the conditional distribution of opponent valuations v_{-i}. For a valuation $v_i \in \mathcal{V}_i$, action $b_i \in \mathcal{A}_i$ and fixed opponent strategies $\beta_{-i} \in \Sigma_{-i}$, we denote the *interim utility* of bidder i by

$$\overline{u}_i(v_i, b_i, \beta_{-i}) = \mathbb{E}_{v_{-i}|v_i}[u_i\left(v_i, b_i, \beta_{-i}(v_{-i})\right)]. \tag{2}$$

We also denote the *interim utility loss* of action b_i incurred by not playing the *best response* action, given v_i and β_{-i}, by

$$\overline{\ell}_i(b_i, v_i, \beta_{-i}) = \sup_{b_i' \in \mathcal{A}_i} \overline{u}_i(v_i, b_i', \beta_{-i}) - \overline{u}_i(v_i, b_i, \beta_{-i}).$$

Note that $\overline{\ell}_i$ generally cannot be observed in online settings because it requires knowledge of a best response.

An interim ϵ-BNE is a strategy profile $\beta^* = (\beta_1^*, \ldots, \beta_n^*) \in \Sigma$ such that no agent can improve his or her own interim expected utility by more than $\epsilon \geq 0$ by deviating from the common strategy profile. Thus, in an ϵ-BNE, we have: $\overline{\ell}_i\left(b_i, v_i, \beta_{-i}^*\right) \leq \epsilon$ for all $i \in \mathcal{I}$, $v_i \in \mathcal{V}_i$, and $b_i \in \mathcal{A}_i$. A 0-BNE is simply called BNE. Thus, in a BNE, every bidder's strategy maximizes his/her expected interim utility given opponent strategies everywhere on his/her type space $\mathcal{V}_i$. While BNE are often defined at the *interim* stage of the game, we also consider *ex-ante* Bayesian equilibria as strategy profiles that concurrently maximize each player's *ex-ante* expected utility $\tilde{u}$. We analogously define $\tilde{u}$ and the *ex-ante utility losses* $\tilde{\ell}$ of a strategy profile $\beta \in \Sigma$ by

$$\tilde{u}_i(\beta_i, \beta_{-i}) = \mathbb{E}_{v_i \sim F_{v_i}}[\overline{u}_i(v_i, b_i, \beta_{-i})] \tag{3}$$

and

$$\tilde{\ell}_i(\beta_i, \beta_{-i}) = \sup_{\beta_i' \in \Sigma_i} \tilde{u}_i(\beta_i', \beta_{-i}) - \tilde{u}_i(\beta_i, \beta_{-i}). \tag{4}$$

Then, an ex-ante BNE $\beta^* \in \Sigma$ can be characterized by the equations $\tilde{\ell}_i(\beta_i^*, \beta_{-i}^*) = 0$ for all $i \in \mathcal{I}$.

Particle Swarm Optimization

This section introduces the customized PSO algorithm and compare it to the gradient-based method afterward. In both approaches, each bidder's strategy is represented by a *policy network* $\beta_i(v_i) = \pi_i(v_i; \theta_i)$ that is specified by a neural network architecture and a corresponding parameter vector $\theta_i \in \mathbb{R}^{d_i}$. These network parameters are the values to be learned. Thus, we have transformed the original infinite-dimensional problem into a finite-dimensional problem over parameter vectors. In the empirical part of this study, we restrict ourselves to fully connected feed-forward neural networks with *rectified linear unit* (ReLU) activations in the output layer, which ensure nonnegative bids—the only feasibility constraint in the auctions under study.

PSO for Equilibrium Computation

The PSO algorithm maintains a swarm of particles where each particle searches for an optimum largely on its own. For this, all particles adjust their trajectories based on their experience and some limited information exchange with other particles.

For our purposes, each agent i is represented by a separate swarm that tries maximizing i's utility. These swarms repeatedly interact in self-play with one another, constituting a standard reinforcement learning procedure. To move around the search space, a particle indexed by p from the population of size P keeps track of its position $\theta_p \in \mathbb{R}^{d_i}$, its individually best found solution $P_p \in \mathbb{R}^{d_i}$ so far, and its velocity $v_p \in \mathbb{R}^{d_i}$ denoting its current momentum in space. To conduct the search, the particle also receives some information from other members of the swarm: for some predefined subset of neighboring particles, p has access to the best solution P_g previously discovered by this neighborhood. Both best positions, the individual's and the neighborhood's best position, are updated if, in the current iteration t, a position was discovered that reaches a higher utility. Particles adjust their search direction according to their knowledge of these two best positions and their current movement: The velocity of particle p at time step $t+1$ is defined as

$$v_p^{t+1} = w v_p^t + c r_1^t \left(P_p^t - \theta_p^t\right) + c r_2^t \left(P_g^t - \theta_p^t\right). \tag{5}$$

The first term of this equation corresponds to the momentum the particle keeps from its current direction and speed. The second part (*cognition component*) corresponds to some attraction force to its current best solution, whereas the last term (*social component*) corresponds to the attraction to the neighborhood's best position. The *inertia weight* w, and *acceleration coefficient* c are hyperparameters and control the weighting of the terms. Meanwhile, the variables r_1^t and r_2^t are uniformly drawn from the unit interval in each iteration (independently for each particle and each dimension) for an increased parameter space exploration.

Algorithm 1: Particle Swarm Optimization for Equilibrium Computation

input: P initial random policies and velocities, parameters w and c, reevaluation frequency k, batch size
for $t = 1, 2, \ldots$ **do**
  Sample a batch of valuation and observation profiles
  Calculate the utility of current strategy profile for all particles by playing against opponents
  for each agent $i \in \mathcal{I}$ **do**
    for each particle $p \in \{1, \ldots, P\}$ **do**
      if $t \bmod k = 0$ **then**
        Reevaluate utility of individual best position P_p
      end if
      Update individual best position P_p
      Update neighborhood's best position P_g
    end for
    for each particle $p \in \{1, \ldots, P\}$ **do**
      Update particle p's current velocity v_p^t via Equation 5
      Update particle p's current parameters via $\theta_p^t = \theta_p^{t-1} + v_p^t$
    end for
  end for
end for

The communication channel between the individual particles is called the topology of the swarm. Following Kennedy and Mendes (2002), we will stick with the *von Neumann* topology, where the neighborhood of a particle consists of the four adjacent particles when the swarm is arranged as a two-dimensional grid. This topology is generally considered a good tradeoff between the two extremes of no information sharing and globally shared information among all particles.

Originally, PSO was introduced for supervised learning settings. The most notable issue for application in games is the instationarity: the best solution may change over time, depending on the strategy changes of the opponents. This includes the possibility of receiving a higher utility than the one achievable in BNE. Crucially, even though we expect the changes in utility to be rather small, this prevents the PSO from being self-correcting. Thus, we will follow Carlisle and Dozier (2000) who proposed, for dynamic single-objective problems, reevaluating the saved utility of the current best positions every k iterations. $k = 10$ was found to be an appropriate choice for our game settings. Additionally, the opposing swarms always compete against the best positions of the opponents only: other design choices, like averaging the utility over all opposing particles or letting one particle only compete against a single opposing particle, would be much more computationally expensive and lead to higher variances. Algorithm 1 summarizes the steps required for PSO in games.

Parameters of the PSO. The social and cognitive components of the velocity guide the particles to optimal areas and leverage their process from individual hill climbers to a combined search, based on shared knowledge, in the area of the current best solutions. The cognitive part allows for divers and throughout sampling of the whole solution space. The social part leverages the particles from individual hill climbers to a swarm combining personal knowledge. By adding a random component, particles can furthermore overshoot their targets and explore new regions of the search space. Although the stochastic component improves the swarm's exploratory behavior and prevents premature convergence to sub-optimal points, it also adds undesirable behavior to the swarm. Convergence is slowed down, and a higher fluctuation of the solution quality can be observed. Therefore, selecting values for the control parameters aims to let particles explore the areas around and beyond their target while keeping the velocities stable.

In particle swarm optimization, it is common practice to select an inertia weight of $w = 0.7290$ and set the acceleration coefficients $c = 1.4945$. This setup was retrieved from the convergence analysis and consecutive introduction of the constriction factor provided by Clerc and Kennedy (2002). However, memory reevaluation is not considered in this approach. In a non-stationary setting, the quality of the remembered positions is expected to change often. Particles are likely to switch their attraction points more frequently. In such cases, they are forced to regularly adjust their trajectories and velocities towards varying points, resulting in erratic and rapid movements. Frequent recalculations of fitness values induce additional randomness into the system, increasing the stochastic energy of the search and thus the variance of the solution quality. We found lower values for the control parameters to be more efficient in our non-stationary setting. Although this modification is necessary to control the particle's behavior, it should be conducted with care as low parameters are more likely to get stuck in sub-optimal areas.

Comparison Between PSO and Gradient-Based Learner

Most notably, PSO maintains a swarm of different particles at all times whereas NPGA only samples other parameters close to the current parameters such that a gradient direction can be approximated. Once the gradient is calculated the samples are all discarded. Therefore, PSO can be expected to search the parameter space more thoroughly.

Additionally, NPGA is usually subject to pretraining the model toward a truthful strategy, because, occasionally, initialized strategies end up in areas of the strategy space where learning is impossible, for example: a strategy that always loses and thus does not receive valuable feedback. Meanwhile, PSO does not require pretraining as particles with a bad initialization benefit from shared information of more successful members of the swarm.

The swarm conducts a parallel distributed search. Thus, the global best position is not adapted step-wise and may be relocated to an entirely different area of the search space during one update step. However, this is not the case for a gradient ascent scheme employed by NPGA. Network parameter modifications are expected to be in a range proportional to the chosen perturbation noise. NPGA further smoothens the learning trajectories by using a standard gradient ascent technique (e.g., Adam) to perform an update step. Moreover, the algorithm consistently makes small updates to the parameter values, whereas PSO stochastically updates the strategy only when a better global solution is found, by simply replacing the present parameters of the strategy network with the values of the current global best position. Furthermore, only the global best position is used to evaluate the loss; hence, it may not represent the particle's true learning curve as they adjust themselves on their neighborhood bests, not on the overall global best.

PSO can be implemented using only basic arithmetic and scalar comparison operations, which can easily be vectorized. Compared to NPGA, PSO is more memory intense as it maintains all particles' positions and velocities. Additionally, the saved best positions need reevaluation every few iterations. However, the bottleneck of NPGA and PSO is the sampling and the approximation of the expected utilities of the population's candidate solutions. Thus, the computational resources and running times for NPGA and PSO are almost indistinguishable.

Evaluation

For a fair comparison, we will stick to the evaluation procedure and the metrics from Bichler et al. (2021) and extend them to the new asymmetric game settings. All of these metrics will be estimated via Monte-Carlo integration over a large sample of H valuations.

To evaluate the quality of strategy-profiles β learned by NPGA, we will provide three metrics: When we have access to the analytical solution BNE β^*, we can simply check whether $\beta \to \beta^*$ by approximating the distance in function space:

$$L_2(\beta_i) \;=\; \left(\frac{1}{H}\sum_{v_i}\left(\beta_i(v_i) - \beta_i^*(v_i)\right)^2\right)^{\frac{1}{2}}. \qquad (6)$$

Additionally, we report each agent's relative utility loss $\mathcal{L}$ that results from unilaterally deviating from the BNE strategy profile by playing the learned strategy β_i instead:

$$\mathcal{L}_i(\beta_i) \;=\; 1 - \frac{\hat{u}_i(\beta_i, \beta^*_{-i})}{\hat{u}_i(\beta^*_i, \beta^*_{-i})}. \qquad (7)$$

However, we are also interested in judging the quality of β when no BNE is known. To do so, we also estimate the potential gains of deviating from the current strategy profile $\hat{\ell}_i \approx \tilde{\ell}_i(\beta_i; \beta_{-i})$. However, this additional metric is expensive: To calculate the estimator $\hat{\ell}$ we introduce a grid of W equidistant points b_w per bidder, covering the action spaces $\mathcal{A}_i$. Now, given a valuation v_i and a bid b_i, we approximate the interim utility loss $\overline{\ell}(v_i, b_i, \beta_{-i})$ of b_i at v_i via

$$\hat{\lambda}_i(v_i, b_i, \beta) = \max_w \frac{1}{H}\sum_h u\left(v_i, b_w, \beta_{-i}(v_{h,-i})\right) - u\left(v_i, b_i, \beta_{-i}(v_{h,-i})\right).$$

Note that the batch H only runs across opponent valuations v_{-i}. To evaluate $\hat{\lambda}_i$ at a single v_i, we thus need $(W{+}1){\cdot}H$ auction evaluations. We can now estimate the ex-ante loss

$$\hat{\ell}_i(\beta) = \frac{1}{H}\sum_h \max_w \hat{\lambda}_i(v_{h,i}, b_w, \beta).$$

Finally, we can calculate the approximate relative utility loss for bidder i as

$$\hat{\mathcal{L}}_i(\beta) \;=\; 1 - \frac{\hat{u}_i(\beta)}{\hat{u}_i(\beta) + \hat{\ell}_i(\beta)}. \qquad (8)$$

Results

In the following subsections, we start by describing the analytical BNE solution, followed by a summary of the experimental results. Figure 1 depicts the learning curves, and Table 1 lists the complete results. The code for reproducing the experiments will be made available on Github at `bnelearn`.

Hyperparameters

We use common hyperparameters across both algorithms, NPGA and PSO, (where applicable) and all settings: Fully connected neural networks with two hidden layers of 10 nodes each with SeLU (Klambauer et al. 2017) activations. For the gradient estimation in NPGA, a population size of $P = 64$ is chosen that matches the population size of PSO. Finding a unique set of parameters that works best in all possible cases is unlikely, but the following ones were found to work sufficiently well for our experiments: $w = 0.5$, $c = 0.8$, and a reevaluation frequency k of 10. All experiments were performed on a single Nvidia GeForce 2080Ti and batch sizes in Monte-Carlo sampling were chosen to maximize GPU-RAM usage: A learning batch size of 2^{17}, a primary evaluation batch size for the computation of $\mathcal{L}$ and L_2 of $H = 2^{20}$, and a secondary evaluation batch size of $H = 2^{11}$ and a grid size of $W = 2^{10}$ for calculating of $\hat{\mathcal{L}}$. Each experiment is repeated 20 times over 2,000 iterations each.

Single-Item Auctions

First, small single-item auctions are analyzed.

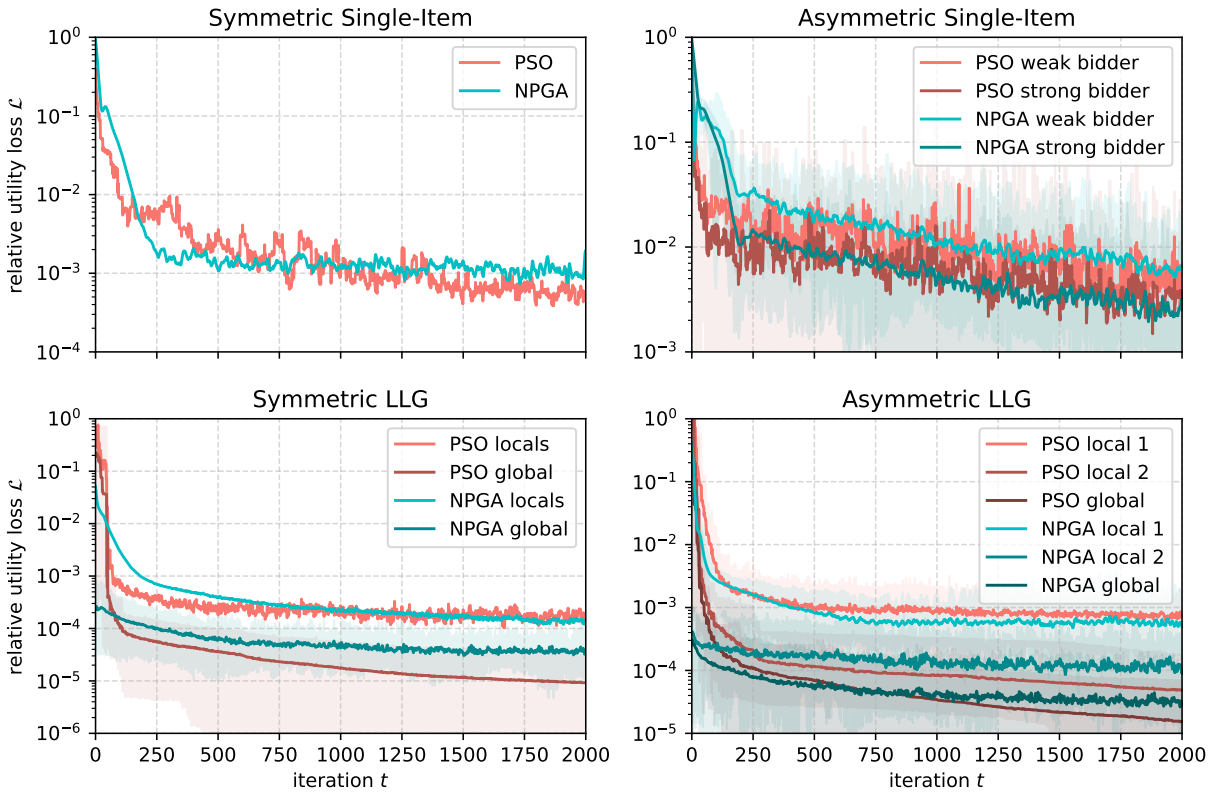

Figure 1: Comparison of both algorithms on the four auction games considered in this article. The learning bidders' expected relative utility loss $\mathcal{L}$ from Equation 7 is measured in equilibrium at each iteration. The mean (opaque line) and standard deviation (shaded area) over 20 runs are depicted.

Symmetric Single-Item Auction. Here the single-item first-price sealed-bid (FPSB) auction with two bidders with uniform priors on the interval $(0, 1)$ will be considered. The symmetric equilibrium strategy is for both bidders to bid half their valuations.

Asymmetric Single-Item Auction. We consider asymmetric single-item auctions. In FPSB auctions, a unique BNE is known for n risk-neutral players with arbitrary but symmetric prior valuations, n risk-averse bidders under symmetric uniform priors (Menezes and Monteiro 2005), and for two players with asymmetric uniform valuation distributions with a stronger and a weaker player (Plum 1992). Asymmetric prior distributions are harder to analyze analytically. We again chose an environment where the analytical BNE is known with two bidders having uniform prior distributions with overlapping valuations on $(0, 1/2)$ and $(0, 1)$, respectively (Plum 1992). In equilibrium, the weaker bidder learns to bid more aggressively than the strong bidder.

Both algorithms learn a stable strategy yielding good results, with the PSO having a slightly lower loss. The relative utility loss is below 1% for both the strong and the weak bidder. As was expected, the stronger agent with a strategic advantage can decrease its relative utility loss further than the weaker agent (Table 1).

Combinatorial Auctions

Multi-item auctions of multiple heterogeneous goods have received significant attention in the past decade in the form of combinatorial auctions. In particular, core-selecting combinatorial auctions have become popular for their use in spectrum auctions worldwide (Bichler and Goeree 2017). For several environments an analytical BNE is available. We consider two specific settings of a combinatorial auction: (1) the well-known local-local-global (LLG) environment (Goeree and Lien 2016) with single-minded bidders, who are only interested in one package and (2) a specific version of it that introduces asymmetries and incentives for overbidding.

Symmetric LLG Auction. The LLG setting includes two local bidders and one global bidder that bid on $m = 2$ items. Local bidders 1 and 2 are each interested in bundle $\{i\}$, whereas the global bidder wants package $\{1, 2\}$ of both items. Each bidder submits a bid $b_i \in \mathbb{R}_+$ for their respective bundle. The setting has been extensively studied in the context of different *core-selecting* pricing rules, as commonly used in real-world spectrum auctions (Day and Cramton 2012; Goeree and Lien 2016). Closed-form solutions of a unique,

Table 1: Results of both algorithms in the four settings considered in this study. Each setting is run for 2,000 iterations and averaged over 20 repetitions. The mean and standard deviation are shown. The better performing algorithm is depicted in bold.

		L_2		$\mathcal{L}$		$\hat{\mathcal{L}}$	
game	bidder	NPGA	PSO	NPGA	PSO	NPGA	PSO
sym. 1-item		0.010 (0.01)	**0.006 (0.00)**	0.002 (0.00)	**0.000 (0.00)**	0.012 (0.00)	**0.011 (0.01)**
asym. 1-item	weak	0.012 (0.00)	**0.007 (0.00)**	0.007 (0.00)	**0.003 (0.00)**	**0.028 (0.01)**	0.039 (0.04)
	strong	0.012 (0.01)	**0.010 (0.01)**	0.003 (0.01)	**0.002 (0.00)**	0.020 (0.01)	**0.010 (0.00)**
sym. LLG	locals	0.012 (0.00)	**0.010 (0.00)**	**0.000 (0.00)**	0.000 (0.00)	0.020 (0.02)	**0.016 (0.01)**
	global	0.012 (0.00)	**0.005 (0.00)**	0.000 (0.00)	**0.000 (0.00)**	0.009 (0.01)	**0.007 (0.01)**
asym. LLG	local 1	**0.040 (0.02)**	0.452 (0.26)	0.000 (0.00)	**0.000 (0.00)**	0.015 (0.01)	**0.007 (0.01)**
	local 2	**0.162 (0.08)**	0.572 (0.02)	**0.000 (0.00)**	0.001 (0.00)	0.014 (0.01)	**0.011 (0.01)**
	global	0.045 (0.02)	**0.029 (0.02)**	0.000 (0.00)	**0.000 (0.00)**	**0.007 (0.01)**	0.007 (0.01)

symmetric BNE under three such rules are known in the LLG setting for both independent and correlated priors: the nearest Vickrey-Clarke-Groves (nearest-VCG) rule, nearest-zero, and nearest-bid rules. The interested reader is referred to Ausubel and Baranov (2019) for details. Under these rules, it has been shown that the global bidder is bidding truthfully in the BNE. The local bidders' BNE strategies differ in each payment rule. For brevity, we only evaluate the algorithms on the nearest-zero payment rule with independent priors.

We again observe that NPGA and PSO converge to the BNE. In fact, after low hundreds of iterations, the relative utility loss for all bidders decreases below 0.1%. The loss of the local bidders reaches a similar level for both algorithms, whereas the loss for the global bidder is lower when learning with PSO.

Asymmetric LLG Auction. Ott and Beck (2013) introduced this highly asymmetric core-selecting combinatorial auction and is also based on the LLG domain. Especially interesting is the incentive of overbidding for a bidder, which practitioners observed in spectrum auctions. Again, there is one seller, a set of three bidders $n = 3$, and two goods $m = 2$. The two local bidders are interested only in the single-item bundles and the global bidder has a value only for the bundle of both items. Unlike in the default LLG setting, Ott and Beck (2013) defined bidder 2 as *favored*, meaning that he or she pays VCG prices for every realization of bids and every optimal assignment. By this setup, bidder 1 has an interesting BNE strategy whereas both opponents can report their valuations truthfully in equilibrium: Bidder 1 places bids for both her desired item and the package of both items. Thus, bidder 1's bid for the package is always larger than his/her bid for the individual desired item, indicating positive demand for the other item even though he/she does not value it. The exact BNE strategy can be found in their original work.

A look at Table 1 reveals that the loss incurred by not playing the BNE strategy in equilibrium $\mathcal{L}$ decreases below 1% for all bidders where PSO slightly outperforms NPGA. Some remarks regarding the BNE and the learned strategies should be made. First, bidder 2 learned to not bid at all for the bundle, explaining the high values for L_2 for both algorithms. However, bidding zero yields the same outcomes as bidding the valuation of item B as long as the truthful bid on item B alone is maintained. Second, for some runs of the PSO algorithm, bidder 1 learns to bid on item B instead of the bundle. Therefore, again the L_2 values are high, whereas the actual relative utility loss reaches competitive values. We conclude that the bidders in this state also reach the BNE outcome as long as none of them deviate. Third, both methods can recover the incentives for overbidding, but they fail capturing some details at the lower end due to the negligible impact on the overall utility.

Conclusion

This paper explores equilibrium learning via neural networks and self-play in Bayesian games. The utility functions exhibit discontinuities, which require non-standard learning methods for neural networks. We propose PSO as it searches a wider range of parameter settings and does not require pretraining. In experiments, we validate the approach on standard single-item and combinatorial auctions, which constitute a pivotal problem in algorithmic game theory with many practical applications. We find that PSO converges to approximate BNE for central benchmark problems in this field, and in some environments, it converges even faster than NPGA with evolutionary strategies. The method can provide an effective numerical tool to compute approximate BNE for combinatorial auctions and for other Bayesian games, without setting-specific customization, all while running on consumer hardware and leveraging GPU-parallelization.

In future research, it will be interesting to explore the theoretical guarantees of PSO in games, design more suitable algorithmic adaptions for the instationarity, and scale the approach to more realistic market scenarios.

References

Ausubel, L. M.; and Baranov, O. 2019. Core-Selecting Auctions with Incomplete Information. *International Journal of Game Theory*.

Bichler, M.; Fichtl, M.; Heidekrüger, S.; Kohring, N.; and Sutterer, P. 2021. Learning Equilibria in Symmetric Auction Games using Artificial Neural Networks. *Nature Machine Intelligence*, 3.

Bichler, M.; and Goeree, J. K. 2017. *Handbook of spectrum auction design*. Cambridge University Press.

Bonyadi, M. R.; and Michalewicz, Z. 2015. Stability analysis of the particle swarm optimization without stagnation assumption. *IEEE Transactions on Evolutionary Computation*, 20(5): 814–819.

Bosshard, V.; Bünz, B.; Lubin, B.; and Seuken, S. 2020. Computing Bayes-Nash Equilibria in Combinatorial Auctions with Verification. *Journal of Artificial Inelligence Research*, 69: 531–570.

Cai, Y.; and Papadimitriou, C. 2014. Simultaneous Bayesian auctions and computational complexity. In *Proceedings of the fifteenth ACM conference on Economics and computation*, 895–910.

Carlisle, A.; and Dozier, G. 2000. Adapting particle swarm optimization to dynamic environments. In *International conference on artificial intelligence*, volume 1, 429–434. Citeseer.

Chakraborty, P.; Das, S.; Roy, G. G.; and Abraham, A. 2011. On convergence of the multi-objective particle swarm optimizers. *Information Sciences*, 181(8): 1411–1425.

Cleghorn, C. W.; and Engelbrecht, A. P. 2018. Particle swarm stability: a theoretical extension using the non-stagnate distribution assumption. *Swarm Intelligence*, 12(1): 1–22.

Clerc, M.; and Kennedy, J. 2002. The particle swarm-explosion, stability, and convergence in a multidimensional complex space. *IEEE transactions on Evolutionary Computation*, 6(1): 58–73.

Day, R. W.; and Cramton, P. 2012. Quadratic Core-Selecting Payment Rules for Combinatorial Auctions. *Operations Research*, 60(3): 588–603.

Dütting, P.; Feng, Z.; Narasimhan, H.; Parkes, D.; and Ravindranath, S. S. 2019. Optimal auctions through deep learning. In *International Conference on Machine Learning*, 1706–1715. PMLR.

Engelbrecht, A. 2005. Fundamentals of Computational Swarm Intelligence. John Wiley & Sons, Chichester, UK.

Goeree, J. K.; and Lien, Y. 2016. On the Impossibility of Core-Selecting Auctions. *Theoretical Economics*, 11(1): 41–52.

Hartline, J.; Syrgkanis, V.; and Tardos, E. 2015. No-Regret Learning in Bayesian Games. In Cortes, C.; Lawrence, N. D.; Lee, D. D.; Sugiyama, M.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 28*, 3061–3069. Curran Associates, Inc.

Hein, D.; Hentschel, A.; Runkler, T.; and Udluft, S. 2017. Particle swarm optimization for generating interpretable fuzzy reinforcement learning policies. *Engineering Applications of Artificial Intelligence*, 65: 87–98.

Hsieh, Y.-P.; Mertikopoulos, P.; and Cevher, V. 2021. The limits of min-max optimization algorithms: Convergence to spurious non-critical sets. In *International Conference on Machine Learning*, 4337–4348. PMLR.

Jiang, M.; Luo, Y. P.; and Yang, S. Y. 2007. Stochastic convergence analysis and parameter selection of the standard particle swarm optimization algorithm. *Information processing letters*, 102(1): 8–16.

Kadirkamanathan, V.; Selvarajah, K.; and Fleming, P. J. 2006. Stability analysis of the particle dynamics in particle swarm optimizer. *IEEE Transactions on Evolutionary Computation*, 10(3): 245–255.

Kennedy, J.; and Eberhart, R. 1995. Particle swarm optimization. In *Proceedings of ICNN'95-international conference on neural networks*, volume 4, 1942–1948. IEEE.

Kennedy, J.; and Mendes, R. 2002. Population structure and particle swarm performance. In *Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600)*, volume 2, 1671–1676. IEEE.

Klambauer, G.; Unterthiner, T.; Mayr, A.; and Hochreiter, S. 2017. Self-Normalizing Neural Networks. *arXiv:1706.02515 [cs, stat]*.

Krishna, V. 2009. *Auction Theory*. Academic press.

Menezes, F. M.; and Monteiro, P. K. 2005. *An Introduction to Auction Theory*. OUP Oxford.

Ott, M.; and Beck, M. 2013. Incentives for Overbidding in Minimum-Revenue Core-Selecting Auctions. F16-V3. Kiel und Hamburg: ZBW - Deutsche Zentralbibliothek für Wirtschaftswissenschaften.

Piperagkas, G. S.; Georgoulas, G.; Parsopoulos, K. E.; Stylios, C. D.; and Likas, A. C. 2012. Integrating Particle Swarm Optimization with Reinforcement Learning in Noisy Problems. In *Proceedings of the 14th Annual Conference on Genetic and Evolutionary Computation*, GECCO '12, 65–72. New York, NY, USA: Association for Computing Machinery. ISBN 9781450311779.

Plum, M. 1992. Characterization and Computation of Nash-Equilibria for Auctions with Incomplete Information. *International Journal of Game Theory*, 20(4): 393–418.

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.

Reyes-Sierra, M.; Coello, C. C.; et al. 2006. Multi-objective particle swarm optimizers: A survey of the state-of-the-art. *International journal of computational intelligence research*, 2(3): 287–308.

Scheepers, C.; Engelbrecht, A. P.; and Cleghorn, C. W. 2019. Multi-guide particle swarm optimization for

multi-objective optimization: empirical and stability analysis. *Swarm Intelligence*, 13(3): 245–276.

Trelea, I. C. 2003. The particle swarm optimization algorithm: convergence analysis and parameter selection. *Information processing letters*, 85(6): 317–325.

Van Den Bergh, F.; et al. 2002. *An analysis of particle swarm optimizers*. Ph.D. thesis, University of Pretoria.

Vickrey, W. 1961. Counterspeculation, Auctions, and Competitive Sealed Tenders. *The Journal of finance*, 16(1): 8–37.