Bilevel Entropy based Mechanism Design for Balancing Meta in Video Games

Sumedh Pendurkar[*]
Texas A&M University
College Station, Texas, USA
sumedhpendurkar@tamu.edu

Chris Chow
Niantic Inc.
San Francisco, California, USA
cchow@nianticlabs.com

Luo Jie
Niantic Inc.
San Francisco, California, USA
rogerluo@nianticlabs.com

Guni Sharon
Texas A&M University
College Station, Texas, USA
guni@tamu.edu

ABSTRACT

We address a mechanism design problem where the goal of the designer is to maximize the entropy of a player's mixed strategy at a Nash equilibrium. This objective is of special relevance to video games where game designers wish to diversify the players' interaction with the game. To solve this design problem, we propose a bi-level alternating optimization technique that (1) approximates the mixed strategy Nash equilibrium using a Nash Monte-Carlo reinforcement learning approach and (2) applies a gradient-free optimization technique (Covariance-Matrix Adaptation Evolutionary Strategy) to maximize the entropy of the mixed strategy obtained in level (1). The experimental results show that our approach achieves comparable results to the state-of-the-art approach on three benchmark domains "Rock-Paper-Scissors-Fire-Water", "Workshop Warfare" and "Pokemon Video Game Championship". Next, we show that, unlike previous state-of-the-art approaches, the computational complexity of our proposed approach scales significantly better in larger combinatorial strategy spaces.

KEYWORDS

Game Meta Balance; Mechanism Design

ACM Reference Format:
Sumedh Pendurkar, Chris Chow, Luo Jie, and Guni Sharon. 2023. Bilevel Entropy based Mechanism Design for Balancing Meta in Video Games. In *Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), London, United Kingdom, May 29 – June 2, 2023,* IFAAMAS, 9 pages.

1 INTRODUCTION

We address a mechanism design problem in which a designer is required to tune a set of parameters of a game (referred as the *game meta*) such that the player's mixed strategy at Nash equilibrium (distribution over strategies) is of maximal entropy. The main motivation behind this problem is video games design. Such games commonly consist of an initial item selection phase where the player attempts to select the best possible combination of game items with respect to the game outcome. In some cases, video games suffer from having an imbalanced game meta leading to degenerate item selection strategies. In such cases, players commonly select an item set from a known subset of 'good item sets', irrespective of their style and preference of play to stay competitively viable in the game. This phenomenon commonly results in reduced players' engagement with the game and renders content added by developers useless as it is sparingly used. As such, balancing the game meta is an important problem for video game designers.

Although a significant amount of research devoted to learning optimal game playing agents [8, 31, 33, 36] exists, there has been little emphasis on balancing the game meta. Consequently, the game meta is often tuned by the game designers manually. Following the *Hearthstone AI Competition* [7], some works have attempted to address the game meta balance problem using standard optimization techniques that were specifically adapted to the Hearthstone domain [5, 37]. However, these approaches are not generally applicable since they are (1) specific for deck-building based games [22], (2) assume players do not change strategies with the change in the game meta, and (3) are computationally intractable for complex games as they utilize win-rate as the evaluation metric and thus have to average over numerous simulations to get a reasonable estimate of win-rates in games with highly stochastic outcomes.

We propose to address these limitations by considering the entropy of the player's mixed strategy over strategy space at Nash equilibrium as the objective function to be maximized as a proxy for game meta balance. We provide a bi-level optimization method for this objective which, unlike previous work, generalizes to games that include a combinatorial item selection phase. We also discuss a degenerate solution to the proposed objective and present a regularization based method to address it. Moreover, we discuss how regularization allows for the incorporation of (human) game designers' preferences into the objective function. Finally, we empirically show that, unlike existing methods, our approach scales significantly better in larger combinatorial strategy spaces.[1]

2 PRELIMINARIES

We consider a **N**ormal-**F**orm **G**ame with **P**arameterized **P**ayoffs denoted $NFG2P_\theta$ where θ is the set of tunable parameters (game meta) which influence the game's outcome. These parameters are assumed to be controlled by the game designer and not by the players. $NFG2P_\theta$ is composed of:

[*]Significant portion of this research was performed during internship at Niantic Inc.

Proc. of the 22nd International Conference on Autonomous Agents and Multiagent Systems (AAMAS 2023), A. Ricci, W. Yeoh, N. Agmon, B. An (eds.), May 29 – June 2, 2023, London, United Kingdom. © 2023 International Foundation for Autonomous Agents and Multiagent Systems (www.ifaamas.org). All rights reserved.

[1]See https://github.com/nianticlabs/metagame-balance for our code.

- N, a finite set of players.
- S_i, a strategy space for each player, $i \in \{1, ..., N\}$.
- $u_i : S_1 \times S_2 \times ... \times S_N \times \mathbb{R}^{|\theta|} \to \mathbb{R}$, a payoff function mapping strategy profiles (strategy per player). and the set of tunable parameters θ to the expected payoff for player i.

Definition 2.1 (Mixed Strategy). A mixed strategy for player i, denoted by σ_i, is a probability distribution over the player's strategy space. That is, $\sigma_i \in \mathbb{R}^{|S_i|}$ **subject to:** sum-of-elements$(\sigma_i) = 1$, min-element$(\sigma_i) \geq 0$. An agent following a mixed strategy, σ_i, is assumed to randomly sample a strategy $s_i \sim \sigma_i$ at every game instance.

Definition 2.2 (Mixed Strategy Nash Equilibrium). A mixed strategy profile, σ^*, is a mixed strategy Nash equilibrium (MSNE) if and only if the following two conditions hold for every player $i \in N$:

(1) Every pure strategy, $s_i \in S_i$, that is assigned a non-zero probability by σ_i^* yields the same expected payoff. That is, $\forall s_i \in S_i,\ \sigma_i^*(s_i) > 0 \implies \mathbb{E}_{s_{-i} \sim \sigma_{-i}^*}[u_i(s_i, s_{-i}; \theta)] = \mathbb{E}_{s \sim \sigma^*}[u_i(s; \theta)]$. We use $\sigma_i(s)$ to represent the probability assigned to strategy s_i under distribution σ_i.
(2) Every pure strategy $s_i \in S_i$ that is assigned a zero probability by σ_i^* yields an expected payoff that is no better than that yielded by strategies assigned with non-zero probability. That is, $\forall s_i \in S_i,\ \sigma_i^*(s_i) = 0 \implies \mathbb{E}_{s_{-i} \sim \sigma_{-i}^*}[u_i(s_i, s_{-i}; \theta)] \leq \mathbb{E}_{s \sim \sigma^*}[u_i(s; \theta)]$.

2.1 Problem Definition

We consider the following game meta balance mechanism design problem.
Input: (1) an $NFG2P_\theta$, (2) a player index, i.
Output: the tunable parameters, θ, that maximize the entropy of the player's i strategy (over strategy space) at the expected MSNE. Formally, the optimization objective is given by

$$\arg\max_{\theta} \mathbb{E}_{\sigma_i^* \sim NFG2P_\theta}[\mathcal{H}(\sigma_i^*)] \tag{1}$$

Where $\mathcal{H}(\sigma_i) = -\sum_{s \in S_i} \sigma_i(s) \log \sigma_i(s)$ is the entropy function.

We assume that all players sample a strategy from their MSNE. This can be viewed as all players playing to achieve high payoffs, and maximizing entropy over such strategies at MSNE would increase the "uniformity" or diversity of selected strategies [4]. As a result, this paper considers Equation 1 to be a parameters assignment resulting in a balanced game meta.

Definition 2.3 (Combinatorial Strategy Space). The strategy space for a given player, i, is denoted *k-order combinatorial* if it is defined by k items selected from a set, $M = \{m_1, m_2, \dots\}$, of available items.

Note, the sampling can be done with or without replacement resulting in $|S_i| = |M^k|$ or $\binom{|M|}{k}$ respectively. We refer to k as the number of *item selections*.

2.2 Related Work

The work presented in this paper is related to two well studied disciplines, namely, mechanism design from game theory, and game meta balance from the video games literature. In this section, we provide the relevant context leading to our contribution.

2.2.1 Mechanism Design. Mechanism design is a sub-field of game theory, where the *game designer* chooses the game structure in terms of the payoff functions. Mechanism design is commonly proposed for optimizing social interactions, e.g., for auctions [21, 26], traffic routing [27–29], load balancing [10] or elections [23]. While our game meta balance problem, as defined above, is a mechanism design problem, it is unique in the sense that it does not allow the game designer to arbitrarily define the payoff functions. Instead, we assume that the game designer can impact some parameters of the payoff functions which might have a limited or noisy impact on the perceived payoffs. Moreover, our game meta balance problem defines a unique objective function that aims to maximize the entropy of the player's mixed strategy over strategy space at MSNE. This contrasts with common mechanism design solutions [16, 19, 35] which aim to maximize social welfare, fairness, and/or incentive compatibility in various contexts.

2.2.2 Game Meta Balance. Initial works [6] on game meta balance relied on insights derived manually from data collected by a heuristic based AI player. They manually tuned the parameters of the game based on statistics like win-rates, objectives completed etc. de Mesentier Silva et al. [5] proposed using an evolutionary algorithm to balance the game meta specifically for deck balancing games [22]. Their evaluation objective is centered around having a 50% win-rate on each of the decks in the game. Further, Fontaine et al. [9] proposed to generate quality diverse decks by using an algorithm termed MAP-Elites [20]. These methods make a limiting assumption that the player's strategy does not change with the change in the payoff matrix. That is, they use a fixed playing agent to evaluate the game meta balance while changing the game meta. This solution is of limited applicability as players often re-evaluate and tune their strategy following changes in the game's meta. Hernandez et al. [14] proposed to solve the problem with a two stage solution: (1) optimize the game meta θ (2) apply Monte Carlo Tree Search (MCTS) [2] to approximate optimal strategies for players under θ. They propose to minimize the mean squared error (MSE) of the *Empirical Response Graph (ERG)* between a target payoff matrix and the payoff matrix under the current θ. The ERG of a payoff matrix is a matrix where all of the negative values in the payoff matrix are substituted with zero [14]. They assume a payoff matrix when the game meta is balanced is known beforehand and is referred as the target payoff matrix. The payoff matrix under θ is obtained by averaging the game outcome over numerous runs for every possible pure strategy profile. Their approach (referred as ERG) was shown to achieve state-of-the-art results on a toy *rock-paper-scissor* domain and the *workshop warfare* domain.

2.2.3 Problems with ERG. Nonetheless, ERG suffers from three major shortcomings (1) ERG assumes a known target payoff that might not hold true in complex video games; (2) ERG uses a sample-based approach for estimating the payoff matrix at each iteration. The computational complexity for doing so scales exponentially with the number of items in a combinatorial strategy space. This is also identified by the authors [14] under Section 7 as an "issue, especially for more computationally intensive games"; (3) minimum

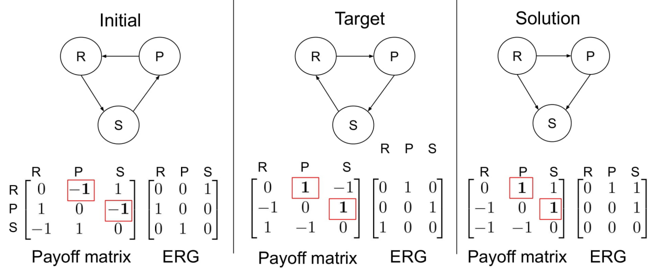


Figure 1: Motivating example: Rock (R) - Paper (P) - Scissor (S) game where $\arg\max_\theta$ **Empirical Response Graph (ERG) does not imply a balanced game meta.** θ **consists of two elements highlighted by a red square in the image which is the payoff between the two pure strategy profiles 1) paper-scissor 2) paper-rock.**

ERG does not always represent a balanced game meta. To illustrate this, consider the rock-paper-scissor game shown in Figure 1 where the designer can only change the payoff between two pure strategy profiles 1) paper-scissor 2) paper-rock. Given $NFG2P^i_{\theta_0}$ with initial θ_0 (as shown in Figure 1) the task of their method is to tune the parameters such that the MSE between the ERG of payoff matrix associated with $NFG2P^i_{\theta_0}$ and ERG of payoff matrix associated with target $NFG2P^t_\theta$ (as shown in Figure 1) is minimized. It is easy to see the $NFG2P^i_{\theta_0}$ and $NFG2P^t_\theta$ have balanced game meta as all the player pure strategies are equally viable. However, the resulting $NFG2P^i_{\theta^*}$ after minimizing θ over ERG (minimum possible ERG of 1) is imbalanced as selecting rock results in the highest expected payoff regardless. This issue can often be observed in cases where some entries in payoff matrix (outcomes) cannot be influenced by the tunable parameters, that is, the target payoff matrix cannot be achieved under a given $NFP2G_\theta$. This leads to partial matching between the game's payoff matrix and the target one which can be detrimental in many cases as shown in the above example.

Our approach does not suffer from these disadvantages as our approach (1) does not assume a known target payoff matrix. Instead, it targets any max-entropy solution, (2) is scalable in combinatorial strategy spaces, and (3) sets an optimization objective function that directly correlates to the game meta balance.

3 BI-LEVEL GAME META BALANCE

To solve the optimization objective in Equation 1, we propose to use a bi-level optimization approach: (1) compute an expected MSNE for a given θ and player i, referred to as *inner level* optimization and (2) apply a black-box optimization method to find $\arg\max_\theta$, referred to as *outer level* optimization. We refer to our method as Bi-level Game Meta Balance (BiGMB).

In this section, we first introduce a method to approximate the MSNE for combinatorial strategy spaces (to solve the inner level optimization), followed by the definition of the outer level objective for combinatorial strategy spaces as well as an approximation of the objective. Further, we demonstrate a problem that can lead to degenerate solutions and consequently present a modification to the optimization objective. Lastly, we describe our choice of black-box optimizer for the outer level optimization.

3.1 Approximate Nash Equilibrium

In order to approximate MSNE for a given $NFG2P_\theta$ with a combinatorial strategy space, we propose to use a Monte Carlo (MC) variant of *Nash Q learning* [15] referred as *Nash MC learning*. We propose to use the following Markov Decision Process formulation for player i, $MDP_i < X_i, A, R_i, T_i >$.

- State Space X_i: a subset of selected items $x \subseteq M$ for player i such that $0 \le |x| \le k$.
- Initial State $x_i^0 = \emptyset$,
- Terminal States $X_i^T \subset X_i = \{x : |x| = k\}$.
- Action Space A: select one item from the available set of items M.[2]
- Reward Function $R_i(x) : X_i \longrightarrow \mathbb{R}$: Reward function for player i. The reward is 0 for $x \notin X_i^T$. For $x \in X_i^T$, that is, x is a valid combinatorial strategy, s_i, we set $R_i(x_i) = u_i(s_i, s_{-i}, \theta)$.
- Transition Function $T_i(x, a) : X_i \times A \longrightarrow X_i$: The transition function is deterministic $x' \longleftarrow x \cup a$ where x' is the next state and a is the item selected at the current step.

A value function $V_i^{\pi_i} : X_i \longrightarrow \mathbb{R}$ is defined by $V_i^{\pi_i}(x) = E_{x_t \sim \pi_i|x}[R_i(x_t)]$ which gives the expected returns when policy π_i is followed. We parameterize the policy as *softmax* distribution given by

$$\pi_i(a|x) = \frac{exp(V_i([x, a]))}{\sum_{a'} exp(V_i([x, a']))} \tag{2}$$

We use a specific MC-based learning method denoted TD(1) [32],

Algorithm 1 Nash Monte Carlo (MC) learning

```
1:  Input: $NFG2P_\theta$, $MDP_i$ for player index $i$
2:  Initialize: value function $V_i$ arbitrarily e.g., $V_i^0(x) \longleftarrow 0 \; \forall x \in X_i$, $t \longleftarrow 0$
3:  Output: policy $\pi_i$
4:  do
5:      $x \longleftarrow \emptyset$
6:      while $|x| \le k$ do
7:          $a \sim \pi_i(\cdot|x)$
8:          add $T(x, a)$ to $x$                    ▷ Add item $a$ to $x$
9:      end while
10:     $NFG2P_\theta$.player$_i$.item_set $\longleftarrow x$
11:     observe $reward \sim NFG2P_\theta$
12:     while $x \neq \{\}$ do
13:         $V_i^{t+1}(x) \longleftarrow \alpha \cdot V_i^t(x) + (1 - \alpha) \cdot reward$
14:         pop item from $x$          ▷ most recent added item to $x$
15:     end while
16:     $t \longleftarrow t + 1$
17: while $|V_i^t - V_i^{t-1}| \le \epsilon$
18: return $\pi_i$ following Equation 2 over $V_i^t$
```

instead of the 1 step, TD(0), based approach in the original Nash Q learning. This is motivated by the fact that MC based methods usually have less bias compared to bootstrapping-based approaches [32]. Note that, on the other side of this coin, MC methods

[2]An item might not be available for selection if sampling is done without replacement.

generally have higher variance compared to bootstrapping methods, which results from stochasticity in the environment. However, as the transition function in our case is deterministic, we believe that the sample variance of MC based methods is minimal.

Algorithm 1 refers to the Nash MC learning given the MDP for player i. In order to achieve MSNE, MC Nash learning requires to learn policy for every player $i \in \{1, ..., N\}$ and each player's V_i is updated after every outcome of $NFG2P_\theta$. The policy for the player (required for Equation 1) returned by Algorithm 1 is further used to calculate the entropy required for outer level optimization.

3.2 Entropy in Combinatorial Strategy Space

In this section, we discuss the objective function used for the outer level optimization specifically for combinatorial strategy spaces. According to Equation 1 and Definition 2.3, given a player i, θ is optimal if the entropy of the player's strategy over combinatorial strategy space at MSNE is maximized. However, in some cases with combinatorial strategy spaces, video game designers do not want every single possible pure strategy profile to be viable. Instead, they prefer to maximize the entropy of the resulting player's distribution of an item being selected when the player follows the strategy sampled from MSNE. For example, consider a shooting based game where the task of the players is to select $k = 2$ guns for the combat, and the task of the designer is to tune the parameters of the guns which denote how powerful it is, and until what range. Game designers do not usually want to make combinatorial strategies like picking two distinct shot guns (good at only close-range combat) viable in the game. Instead, game designers tend to set the parameters such that each gun in the game is viable and is selected by players in some or other combination. Formally, let Y be a random variable that models an item being selected by the player i's policy at MSNE (π_i). Thus, our objective for combinatorial strategy spaces can be given by:

$$\mathcal{H}(Y) = -\sum_{j=1}^{j=|M|} P(Y = m_j) \log P(Y = m_j) \tag{3}$$

To calculate P(Y) given π_i we introduce some additional terms for simplicity. Let Y^t be a random variable over the items that are picked at t^{th} time instance by the π_i. $P(Y)$ can be represented by

$$P(Y) = \frac{1}{k} \sum_{t=1}^{k} P(Y^t) \tag{4}$$

Note, the normalization term $(1/k)$ is required as the item is equally likely to be picked at each time instance. This results from the Definition 2.3 that item ordering does not matter. Each $P(Y^t)$ can be given by

$$P(Y^t) = \sum_{b_l=1}^{b_l=|M|} P\left(Y^t \bigcap_{l=1}^{t-1} (Y^l = m_{b_l})\right) \tag{5}$$

$$= \sum_{b_j, b_l=1}^{b_j, b_l=|M|} P\left(Y^i | \bigcap_{l=1}^{t-1} Y^l = m_{b_l}\right) \prod_{j=1}^{t-1} P\left(Y^j = m_{b_j} | \bigcap_{l=1}^{j-1} Y^l = m_{b_l}\right) \tag{6}$$

We use the law of total probability in Equation 5 where we sum over all possible combinations of selections of size $t - 1$. Each individual term $P(Y^t | \bigcap_{l=1}^{t-1} Y^l)$ in Equation 6 is the policy π_i as it models distribution over the next item given previous selections.

The elements over which the sum is performed in Equation 6, grows combinatorially with increasing k and becomes computationally intractable. To address this issue, we propose a tractable approximation for $P(Y)$ given by $\hat{P}(Y) := P(Y^1)$. This approximation results from the assumption that $P(Y^i) = P(Y^j)$ for $0 \leq i, j \leq |M|$. Note, Algorithm 1 approximates a policy that maximizes the expected payoff [15, 32]. There can be multiple policies that are optimal with respect to the expected payoff in combinatorial action spaces, biased towards selecting specific order. Thus, the assumption, $\hat{P}(Y) := P(Y^1)$ does not hold true in general.

3.3 Regularization

In some cases, the tunable parameters, θ, include a set of item attributes that define the item's impact on the game's outcome. Such cases are prone to degenerate solutions where assigning all items the exact same attribute values leads to a max-entropy MSNE. For example, this might happen when assigning all Pokemon in the VGC domain the exact same strength, health, and special abilities. Such an outcome is, obviously, not desirable. Moreover, most items in video games have special traits and game designers do not want to deviate away from these original traits. To address this, we add a regularization term to the entropy-based objective (Equation 3). Our regularized objective is defined by:

$$\max_{\theta} \mathcal{H}(Y) - ||\mathbf{r} \odot \theta - \mathbf{r} \odot \theta_0||_2 \tag{7}$$

where θ_0 is the initial game parameters, commonly provided by the game designer, and $\mathbf{r} \in \mathbb{R}^{|\theta|}$ is a regularization vector, which denotes how far can the deviation be for each of the tunable parameters, and $\odot$ is element-wise multiplication. Note, we use a vector $\mathbf{r}$ so that it gives the game designer an option to weigh the restrictions over the various parameters in θ, based on game specific requirements.

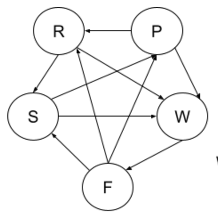
3.4 Algorithms for Outer Level Optimization

To optimize over θ (Equation 7) we use a black-box optimizer, Covariance Matrix Adaptation Evolution Strategy (CMA-ES) [11]. CMA-ES is a gradient free optimization technique that also models the pairwise correlations between the parameters. We suspect such pairwise correlation exists between the tunable parameters θ for the domains presented in the experimental section. Further, CMA-ES was shown to perform well on non-convex functions and 'difficult functions' in large dimensional spaces [13]. Considering high dimensional spaces $|\theta|$, especially in more complex games, we believe CMA-ES would be most effective for our problem.

4 EXPERIMENTS

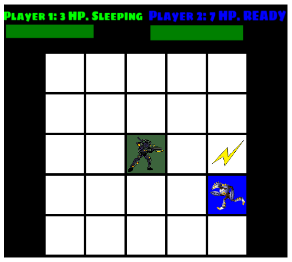
The experimental section is designed to answer the following questions:

(1) How does BiGMB compare against the current state-of-the-art baseline in cases where the strategy space is low dimensional?
(2) How well does the computational complexity of BiGMB scale with the dimensionality of the strategy space? specifically:
    (a) increase in item selections, k.



$$\begin{array}{c|ccccc} & R & P & S & F & W \\ \hline R & 0 & -1 & 1 & -1 & 1 \\ P & 1 & 0 & -1 & -1 & 1 \\ S & -1 & 1 & 0 & -1 & 1 \\ F & 1 & 1 & 1 & 0 & -1 \\ W & -1 & -1 & -1 & 1 & 0 \end{array}$$

(a) Rock-Paper-Scissors-Fire-Water (RPSFW)



(b) Workshop Warfare (WW)

(c) Pokemon VGC (VGC)

Figure 2: Domains used for experimentation.

   (b) increase in the size of the item set, $|M|$.
(3) How does a change in the regularization parameter $\mathbf{r}$ affect the solution quality?
(4) How does the approximation of outer level objective ($\hat{P}(Y) := P(Y^1)$) affect the performance of BiGMB?

4.1 Domains

For our experimental section, we consider three zero-sum symmetric, two-player benchmark domains, (1) Rock-Paper-Scissors-Fire-Water (RPSFW), (2) Workshop Warfare (WW), and (3) Pokemon Video-Game-Championship (VGC). Our choice of RPSFW and WW follows the domains used in the previous state-of-the-art publication [14]. RPSFW is a minor variation of rock paper scissors considered by Hernandez et al. [14] where the game meta is known to be not balanced. That is, the entropy of the player's strategy at MSNE is not uniform. Our choice of VCG follows Reis et al. [25] who proposed the benchmark domain (VGC) specifically for game meta balance.

4.1.1 Rock Paper Scissors Fire Water (RPSFW). ***(a)*** *Description:* RPSFW is an extension of the rock paper scissors game with the addition of two items namely fire and water [34]. ***(b)*** *Strategy Space:* The task of the players is to select $k = 1$ item from an item set M = {rock, paper, scissors, fire, water}. ***(c)*** *Payoff Function:* Fig 2a shows the win-loss relationship graph between the items, along with the payoff matrix. ***(d)*** *Parameters* θ: The set of tunable parameters θ are set as the values above the diagonal of the payoff matrix. Thus $|\theta|$ = 10.

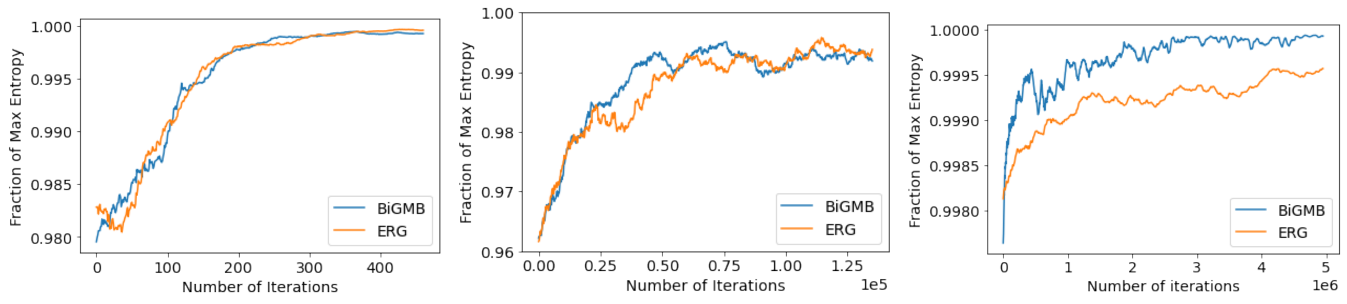
4.1.2 Workshop Warfare (WW). ***(a)*** *Description:* Workshop Warfare is a 5x5 grid game [14] where the task of the player is to deplete the health of the opponent bot to 0. At each timestep, the player can either stand still, move in any valid direction, or use a special attack that is specific to each bot. We use MCTS policy for the gameplay following [14]. Figure 2b shows a screenshot from the game. We refer the reader to Hernandez et al. [14] for further details. ***(b)*** *Strategy Space*: The task of each player is to select $k = 1$ item or bot from a set of three game items M = {Nail Bot, Saw bot, Torch Bot}. ***(c)*** *Payoff Function:* A player receives a payoff of 1 if the health of the opponent reaches 0, -1 otherwise. ***(d)*** *Parameters* θ*:* The parameters θ consist of some common parameters across bots/items namely health, cooldown for the attack, damage (based on attack), and cooldown between moves. Some bot specific statistics, namely torch range and duration (for torch bot), damage boost, and duration (for saw bot) are also included in θ. The total number of parameters is $|\theta| = 16$. Our choice follows [14].

4.1.3 Pokemon VGC (VGC). ***(a)*** *Description:* Pokemon VGC is a simulator that has been recently proposed by Reis et al. [25] for three challenge tracks, with game meta balance being one of them. Each Pokemon has 4 moves, health and a type (like water, electric, etc). At each time instance, the player can either switch to another valid Pokemon, do nothing, or select a move. There are known fixed number of moves with certain parameters that are shared amongst Pokemons. Further, each move can have some special effects which can change the damage dealt with the move or reduce the damage taken. We refer the reader to [25] for further details. All Pokemons are generated following [25]. We use a heuristic based policy to play the game which randomly switches the Pokemon (with a given probability) else samples a move with probability move power $\times$ move accuracy + constant. Figure 2c shows a screenshot from the simulator. ***(b)*** *Strategy Space*: The task of the players is to select k Pokemons from a total of $|M|$ Pokemons. The simulator allows to change k and M, thus these are set specific to each experiment. When the number of selections was greater than 1 ($k > 1$) sampling without replacement was considered. ***(c)*** *Payoff Function:* A player receives a payoff of 1 if the health of all opponent's Pokemons reaches 0, a payoff of -1 if the health of all player's Pokemons reaches zero, else payoff of 0 (timeout). ***(d)*** *Parameters* θ*:* The set of parameters θ for the game are set as three parameters of the every move namely power, accuracy, and maximum number of times the move can be used. Given that there are 69 moves, $|\theta| = 207$. Note that, $|\theta|$ does not change with k and $|M|$.

4.2 Setup

4.2.1 Outer level Optimization: The population size was set to $4 + 3log(|\theta|)$ following the literature [12]. All parameters in θ were normalized by scaling to a range [0,1]. The initial variance was set to 0.7, 0.3, 0.1 for RPSFW, WW, VGC respectively. The variance was constant across all parameters in θ.

4.2.2 Value Function Representation: For RPSFW and WW the size of M is relatively small with sizes 5 and 3 respectively. Thus, we use an array (or a table) as the value function. For VGC we use neural networks as the value function. The items in VGC have common attributes like (power, accuracy, health, etc.), which can be exploited by neural networks as they are known to generalize over the data [30]. The neural network was set to have two hidden



(a) RPSFW **(b) WW** **(c) VGC**

Figure 3: Comparison of BiGMB with ERG on three domains, namely RPSFW, WW, VGC. On the x-axis we have the number of total game iterations, on the y-axis we have the fraction of entropy values. For VGC $|M| = 10$ is considered with $k = 2$. The plots are smoothed over a window of 25, 100, 50 iterations for RPSFW, WW, and VGC respectively.

layers with sizes 128, 64. Mean Squared Error (MSE) was used as the loss function. Adam [18] was used as the optimizer. As we observed high variance when updates were performed on a single sample, we sampled multiple outcomes of games until the number of states to be updated exceeded 30 to reduce the variance. PyTorch [24] was used to implement neural networks. Finally, distinct value functions were used for each of the players.

4.2.3 Inner level Optimization: The maximum number of iterations in inner level optimization was decided based on the convergence of value function, averaging over running multiple runs. In the case of games where function approximation was used (i.e., VGC), the number of inner level iterations was decided based on the convergence of the payoffs obtained by the players against a set of pre defined agents provided by Reis et al. [25]. We observed convergence to an expected payoff ~ 0.4 denoting that the learnt agent significantly outperformed the baselines. The value function is not reinitialized after updates to θ. This was done following our observation that the policy converged faster than random initialization, especially in later iterations where θ starts converging.

4.2.4 ERG Implementation: We are not aware of any generic algorithms to solve game meta balance for our formulation. Thus, we adapt the previous state-of-the-art ERG [14] as our baseline by making a few changes required for a fair comparison. First, we use CMA-ES instead of Bayesian optimization for ERG as well as BiGMB. This is preferred when $|\theta|$ is large, like in VGC. Second, ERG assumes the target win rates (or payoffs) to be known. To overcome that, we make a similar choice for the target payoff matrix for RPSFW as ERG with an addition of two pure strategy profiles (or items, fire, and water) and the same choice as ERG for WW. For VGC we use a target payoff matrix with all 0's. Further, to calculate the current payoff matrix ERG suggests running numerous simulations for every possible pure strategy profile. This number was set to 50 for WW as suggested in [14] and lower (10) for VGC due to computational constraints. All other hyperparameters like policy used for gameplay (along with hyperparameters of policy), the initial variance of CMA-ES, and population size, were assigned the same value as BiGMB for a fair comparison.

4.2.5 Evaluation Methods: We use the fraction of maximum entropy which is given as the ratio of the entropy of the player's strategy (over strategy profiles) at MSNE for the θ (Equation 3) and entropy of the discrete uniform distribution of size $|M|$ as the performance metric. Note, we calculate the ratio with respect to discrete uniform distribution as it is known that it has maximum entropy [4]. We use the same metric to evaluate both BiGMB as well as ERG. The target payoff matrix for all domains is a (specific) optimal solution to Equation 3, thus ERG targets optimal solution to Equation 3. For the study of convergence, we compare the number of times the game outcome was sampled also referred as *game iterations*. Our choice is motivated by our observation that the time required to complete a game iteration (sampling a game outcome) dominates the time required for an optimization step in Algorithm 1 as well as calculating the L2 loss for the ERG approach. We say that a method has *converged* to the best solution if the fraction of max entropy exceeds 0.999 at any iteration.

4.2.6 Other Details: The strategy at MSNE is the same for all players in symmetric games, thus we simply use the policy of player 1 for evaluation of outer level objective. We set $\mathbf{r}$ as a scalar value r times vector of ones $[1, ..., 1]^T$ of the dimension $|\theta|$. Unless stated otherwise r was set to 0.001.

4.3 Comparison on Problems with Low Dimensional Strategy Spaces

Figure 3 shows the comparison of BiGMB against the baseline ERG on problems with relatively low dimensional combinatorial strategy spaces. For RPSFW, it can be seen that both of the approaches have similar performance. Note that, as RPSFW allows tuning the payoff matrix directly, ERG does not require any game iterations for estimating the payoff matrix (win rates). Therefore, we consider both of the methods require a single game iteration per outer level optimization iteration to have a fair comparison. In practical scenarios this can be easily achieved by replacing Algorithm 1 with a known solver [1].[3] Similar to RPSFW, BiGMB has competitive performance

[3]We were able to converge on the value function in 400 game iterations with a learning rate of 0.01.

for the WW domain. For VGC with $|M| = 10$ and $k = 2$ we can see that BiGMB clearly outperforms ERG as BiGMB requires fewer game iterations to converge to a solution with maximal entropy. Note that, we observed ERG also converges to a solution with maximal entropy, but it required 1.5×10^7 game iterations. Thus, from these experiments, we can conclude that BiGMB is competitive to ERG on problems with relatively small combinatorial strategy spaces. Figure 3 also suggests that with the growing dimensionality of strategy spaces (from RPSFW or WW to VGC) BiGMB outperforms ERG, which is further investigated in the following section. This answers Question (1) stated at the beginning of Section 4.

4.4 Scalability Study

In this section, we study the scalability trends across increasing k and $|M|$ values in the VGC domain.

4.4.1 Scalability across k. Table 1 shows the comparison of BiGMB against ERG with an increasing number of selections k and a constant $|M| = 10$. We see that BiGMB requires fewer game iterations with increasing selection size k. ERG, on the other hand, requires an exponentially growing number of game iterations for a single evaluation of outer level optimization objective. This is evident as they require a total of $0.5 \times \binom{|M|}{k} \times \binom{|M|-1}{k}$ number of game iterations for evaluation of their objective. For $k = 4$ we can see that BiGMB requires fewer game iterations for convergence than required for a single evaluation of the ERG objective. Thus, the quicker convergence, especially with the growing number of selections k highlights the advantages of BiGMB as compared to ERG. Figure 4 shows the fraction of maximum entropy achieved by BiGMB against number of game iterations for different values of k for VGC domain. We can infer that the optimization problem gets easier with increasing values of k. We suspect this occurs specifically for VGC domain because of 'type advantages' in the game, where certain types of Pokemons are inherently weak to other types of Pokemons. For instance, a Pokemon with type water is powerful in game against a Pokemon with fire type. Thus, with smaller values of k it is difficult to find θ such that the game meta is balanced as that might lead the player's strategy at MSNE to exploit a single type. On the contrary, with larger values of k, diverse selections across types are possible which can help to mitigate this problem and make the objective easier to optimize.

Table 1: Comparison against ERG with increasing k and constant $|M| = 10$ for VGC domain. The number of simulations for ERG was set to 50 following [14] for calculations.

k	Game Iterations Until Convergence (BiGMB)	Game Iterations for single evaluation of objective (ERG)
1	$\leq 3.0 \times 10^6$	$\sim 2.25 \times 10^3$
2	$\leq 1.5 \times 10^6$	$\sim 4.95 \times 10^4$
3	$\leq 5.0 \times 10^5$	$\sim 3.60 \times 10^5$
4	$\leq 5.0 \times 10^5$	$\sim 1.10 \times 10^6$

4.4.2 Scalability across $|M|$. Table 2 shows a comparison of BiGMB against ERG with increasing size of $|M|$ while keeping the number of selections as $k = 2$. We observe that the number of game iterations required for BiGMB does not grow exponentially as compared to that required by ERG for a single evaluation with growing values of $|M|$. Similar to the previous case, we see that with $|M| \geq 30$ BiGMB requires fewer game iterations as compared to game iterations required for a single evaluation for ERG. Further, Figure 5 shows the performance of BiGMB against number of game iterations for increasing $|M|$. Table 2 and Figure 5 also show that with increasing $|M|$ BiGMB is (1) relatively slower to converge. (2) entropy values decrease on average. We suspect this is because of the growing complexity of the problem specifically for VGC domain. As discussed in Section 4.1.3, the size of $|\theta|$ is constant irrespective of

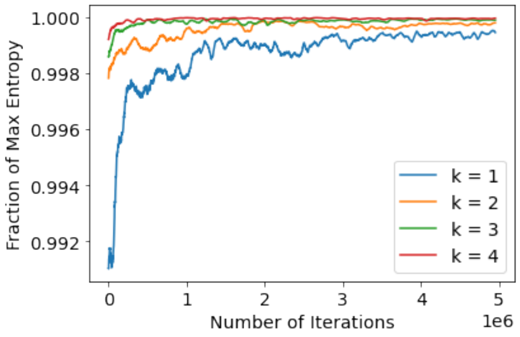


Figure 4: A study of scalability of BiGMB with increasing value of k on VGC domain and constant $|M| = 10$.

Table 2: Comparison against ERG with increasing $|M|$ and constant $k = 2$ for VGC domain. Number of simulations for ERG was set to 50 following [14] for calculations.

$\|M\|$	Game Iterations Until Convergence (BiGMB)	Game Iterations for single evaluation of objective (ERG)
5	$\leq 2 \times 10^5$	$\sim 2.25 \times 10^3$
10	$\leq 3 \times 10^5$	$\sim 4.95 \times 10^4$
20	$\leq 3 \times 10^5$	$\sim 8.97 \times 10^5$
30	$\leq 5 \times 10^5$	$\sim 4.72 \times 10^6$
50	$\leq 5 \times 10^5$	$\sim 3.75 \times 10^7$

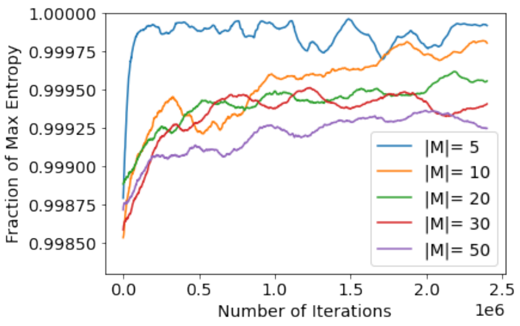


Figure 5: A study of scalability of BiGMB with increasing value of $|M|$ on VGC domain and constant $k = 2$.

changes in $|M|$ and k and consists of parameters of the moves which are shared amongst Pokemons. Thus, with the growing number of $|M|$ the problem becomes more difficult as a change in parameters of a move affects a greater number of Pokemons.

From these two studies we can conclude that unlike ERG, BiGMB scales significantly better with increasing values of $|M|$ and k in combinatorial action spaces. The experiments also suggest that convergence of BiGMB (quality and time) is dependent on the inherent 'complexity' of the game resulting from the choice of θ and not specifically on the dimensionality of combinatorial strategy spaces. This answers Question (2a) and (2b) stated at the beginning of Section 4.

4.5 Impact of Regularization

Figure 6 shows a plot of maximum entropy of the solution obtained by optimizing Equation 7 with increasing value of r on the VGC domain. From the plot we can see that with $r \leq 1$ the entropy value is fairly constant. We suspect the slight variations are due to stochasticity of evaluations. Further, with $r > 1$ we see a sharp decline in the entropy of the solution. This suggests that there might be multiple global optimal solutions to maximal entropy, and convergence to them can be controlled with value of r. However, after a certain value of r, the regularization term dominates the objective and CMA-ES returns the initial solution as the optimal solution. This answers Question (3) stated at the beginning of Section 4.

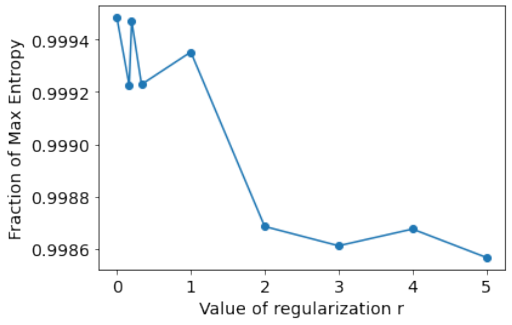


Figure 6: Impact of increasing value of regularization parameter r on the solution quality. The experiments are carried out on VGC domain with $|M| = 10$ and $k = 2$ with a maximum of 10^6 game iterations.

4.6 Performance of $\hat{P}(Y)$

Figure 7 shows the performance when entropy over $\hat{P}(Y)$ is maximized while evaluating based on the original objective (entropy over $P(Y)$) as discussed in Section 3.2. From the figure, we can see that the orange and green line have similar shapes, with the green light being slightly below orange line. That is, while optimizing over entropy of $\hat{P}(Y)$, values of entropy of $P(Y)$ and $\hat{P}(Y)$ are similar at every single iteration. Further, both, optimizing over entropy of $P(Y)$ and $\hat{P}(Y)$ converge in a similar number of game iterations. We see a dip when optimizing over entropy of $\hat{P}(Y)$ around 6×10^5 iterations, which we suspect might be due to exploration. This experiment suggests that the approximation of $P(Y) := \hat{P}(Y)$ can be viable, especially in domains with relatively larger values of k where computational complexity can be dominated by the calculation of $P(Y)$. This answers Question (4) of our experimental section.

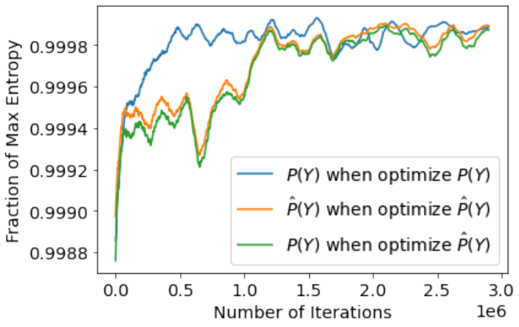


Figure 7: Evaluation of entropy of $\hat{P}(Y)$ as objective with respect to the original objective on the VGC domain with $|M| = 10$ and $k = 3$. The blue line represents entropy values of $P(Y)$ when optimized on entropy over $P(Y)$. The orange and green line represent entropy of $\hat{P}(Y)$ and $P(Y)$ respectively when optimized on $\hat{P}(Y)$.

5 SUMMARY

In this paper, we formulate the game meta balance problem as bi-level mechanism design problem where the objective of the game designer is to maximize the entropy of the player's mixed strategy over strategy space at Nash equilibrium. We propose to use a bi-level optimization method where we (1) use Nash Monte-Carlo learning to approximate the mixed strategy Nash equilibrium (2) use Covariance Matrix Adaptation Evolutionary Strategy to maximize the entropy of the player's mixed strategy at Nash equilibrium that is scalable by design as opposed to previous state-of-the-art. The experimental results show that the performance of our approach (BiGMB) is competitive with ERG, the state-of-the-art, on problems with low dimensional strategy spaces. The results further show that BiGMB scales to large combinatorial strategy spaces where ERG fails to complete a single evaluation of their objective.

6 LIMITATIONS & FUTURE WORK

Sample Complexity: BiGMB solves the inner level optimization until convergence for every evaluation of the outer level objective, which results in relatively poor sample complexity. Recent approaches [3, 17] have tackled bi-level optimization in the context of gradient based optimization, by obtaining unbiased estimates for inner level solution by sampling, resulting in significantly better sample complexity. In the future, we intend to explore such stochastic methods for game meta balance problem.

Empirical evaluation on N-player games: The choice of domains was limited by the availability of benchmark games that allow a change in the game meta θ. We intend to develop benchmark domains with extensions to N-player non-symmetric games and benchmark game meta balance algorithms.

REFERENCES

[1] David Avis, Gabriel D Rosenberg, Rahul Savani, and Bernhard Von Stengel. 2010. Enumeration of Nash equilibria for two-player games. *Economic theory* 42, 1 (2010), 9–37.

[2] Guillaume Chaslot, Sander Bakkes, Istvan Szita, and Pieter Spronck. 2008. Monte-carlo tree search: A new framework for game ai. In *Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment*, Vol. 4. 216–217.

[3] Tianyi Chen, Yuejiao Sun, and Wotao Yin. 2021. Closing the gap: Tighter analysis of alternating stochastic gradient methods for bilevel problems. *Advances in Neural Information Processing Systems* 34 (2021), 25294–25307.

[4] Keith Conrad. 2004. Probability distributions and maximum entropy. *Entropy* 6, 452 (2004), 10.

[5] Fernando de Mesentier Silva, Rodrigo Canaan, Scott Lee, Matthew C Fontaine, Julian Togelius, and Amy K Hoover. 2019. Evolving the hearthstone meta. In *2019 IEEE Conference on Games (CoG)*. IEEE, 1–8.

[6] Fernando de Mesentier Silva, Scott Lee, Julian Togelius, and Andy Nealen. 2017. AI-based playtesting of contemporary board games. In *Proceedings of the 12th International Conference on the Foundations of Digital Games*. 1–10.

[7] Alexander Dockhorn and Sanaz Mostaghim. 2019. Introducing the hearthstone-AI competition. *arXiv preprint arXiv:1906.04238* (2019).

[8] Adrien Ecoffet, Joost Huizinga, Joel Lehman, Kenneth O Stanley, and Jeff Clune. 2019. Go-explore: a new approach for hard-exploration problems. *arXiv preprint arXiv:1901.10995* (2019).

[9] Matthew C Fontaine, Scott Lee, Lisa B Soros, Fernando de Mesentier Silva, Julian Togelius, and Amy K Hoover. 2019. Mapping hearthstone deck spaces through map-elites with sliding boundaries. In *Proceedings of The Genetic and Evolutionary Computation Conference*. 161–169.

[10] Daniel Grosu and Anthony T Chronopoulos. 2004. Algorithmic mechanism design for load balancing in distributed systems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34, 1 (2004), 77–84.

[11] Nikolaus Hansen. 2006. *The CMA Evolution Strategy: A Comparing Review.* Springer Berlin Heidelberg, Berlin, Heidelberg, 75–102. https://doi.org/10.1007/3-540-32494-1_4

[12] Nikolaus Hansen. 2016. The CMA evolution strategy: A tutorial. *arXiv preprint arXiv:1604.00772* (2016).

[13] Nikolaus Hansen, Anne Auger, Raymond Ros, Steffen Finck, and Petr Pošík. 2010. Comparing results of 31 algorithms from the black-box optimization benchmarking BBOB-2009. In *Proceedings of the 12th annual conference companion on Genetic and evolutionary computation*. 1689–1696.

[14] Daniel Hernandez, Charles Takashi Toyin Gbadamosi, James Goodman, and James Alfred Walker. 2020. Metagame Autobalancing for Competitive Multiplayer Games. In *2020 IEEE Conference on Games (CoG)*. IEEE, 275–282.

[15] Junling Hu and Michael P Wellman. 2003. Nash Q-learning for general-sum stochastic games. *Journal of Machine Learning Research* 4, Nov (2003), 1039–1069.

[16] Zhiyi Huang and Sampath Kannan. 2012. The exponential mechanism for social welfare: Private, truthful, and nearly optimal. In *2012 IEEE 53rd Annual Symposium on Foundations of Computer Science*. IEEE, 140–149.

[17] Prashant Khanduri, Siliang Zeng, Mingyi Hong, Hoi-To Wai, Zhaoran Wang, and Zhuoran Yang. 2021. A near-optimal algorithm for stochastic bilevel optimization via double-momentum. *Advances in Neural Information Processing Systems* 34 (2021), 30271–30283.

[18] Diederick P Kingma and Jimmy Ba. 2015. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*.

[19] Noman Mohammed, Hadi Otrok, Lingyu Wang, Mourad Debbabi, and Prabir Bhattacharya. 2009. Mechanism design-based secure leader election model for intrusion detection in MANET. *IEEE transactions on dependable and secure computing* 8, 1 (2009), 89–103.

[20] Jean-Baptiste Mouret and Jeff Clune. 2015. Illuminating search spaces by mapping elites. *arXiv preprint arXiv:1504.04909* (2015).

[21] Roger B Myerson. 1981. Optimal auction design. *Mathematics of operations research* 6, 1 (1981), 58–73.

[22] Joel Niklaus, Michele Alberti, Vinaychandran Pondenkandath, Rolf Ingold, and Marcus Liwicki. 2019. Survey of artificial intelligence for card games and its application to the Swiss game Jass. In *2019 6th Swiss Conference on Data Science (SDS)*. IEEE, 25–30.

[23] Peter C Ordeshook et al. 1986. Game theory and political theory. *Cambridge Books* (1986).

[24] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. 2017. Automatic differentiation in pytorch. (2017).

[25] Simão Reis, Luís Paulo Reis, and Nuno Lau. 2021. VGC AI Competition-A New Model of Meta-Game Balance AI Competition. In *2021 IEEE Conference on Games (CoG)*. IEEE, 01–08.

[26] Tuomas Sandholm and Andrew Gilpin. 2006. Sequences of take-it-or-leave-it offers: Near-optimal auctions without full valuation revelation. In *Proceedings of the Fifth International Joint Conference on Autonomous Agents and Multiagent Systems*. 1127–1134.

[27] Guni Sharon. 2021. Alleviating Road Traffic Congestion with Artificial Intelligence. In *Proceedings of International Joint Conference on Artificial Intelligence*. 4965–4969.

[28] Guni Sharon, Stephen D Boyles, Shani Alkoby, and Peter Stone. 2019. Marginal Cost Pricing with a Fixed Error Factor in Traffic Networks.. In *Proceedings of Autonomous Agents and Multiagent Systems*. 1539–1546.

[29] Guni Sharon, Michael W Levin, Josiah P Hanna, Tarun Rambha, Stephen D Boyles, and Peter Stone. 2017. Network-wide adaptive tolling for connected and automated vehicles. *Transportation Research Part C: Emerging Technologies* 84 (2017), 142–157.

[30] Jocelyn Sietsma and Robert JF Dow. 1991. Creating artificial neural networks that generalize. *Neural networks* 4, 1 (1991), 67–79.

[31] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al. 2017. Mastering the game of go without human knowledge. *nature* 550, 7676 (2017), 354–359.

[32] Richard S Sutton and Andrew G Barto. 2018. *Reinforcement learning: An introduction.* MIT press.

[33] Maciej Świechowski, Tomasz Tajmajer, and Andrzej Janusz. 2018. Improving hearthstone AI by combining MCTS and supervised learning algorithms. In *2018 IEEE Conference on Computational Intelligence and Games (CIG)*. IEEE, 1–8.

[34] Rustam Tagiew. 2009. Hypotheses about typical general human strategic behavior in a concrete case. In *Congress of the Italian Association for Artificial Intelligence*. Springer, 476–485.

[35] Chunchun Wu, Sheng Zhong, and Guihai Chen. 2014. A strategy-proof spectrum auction for balancing revenue and fairness. In *2014 IEEE 11th Consumer Communications and Networking Conference (CCNC)*. IEEE, 827–832.

[36] Deheng Ye, Guibin Chen, Wen Zhang, Sheng Chen, Bo Yuan, Bo Liu, Jia Chen, Zhao Liu, Fuhao Qiu, Hongsheng Yu, et al. 2020. Towards playing full moba games with deep reinforcement learning. *Advances in Neural Information Processing Systems* 33 (2020), 621–632.

[37] Yulun Zhang, Matthew C Fontaine, Amy K Hoover, and Stefanos Nikolaidis. 2022. Deep surrogate assisted MAP-elites for automated hearthstone deckbuilding. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 158–167.