mathematics

MDPI

Review

A Survey on Population-Based Deep Reinforcement Learning

Weifan Long [1], Taixian Hou [1], Xiaoyi Wei [1], Shichao Yan [1], Peng Zhai [1,2,3,*] and Lihua Zhang [1,4,5,*]

[1] Academy for Engineering and Technology, Fudan University, Shanghai 200433, China
[2] Ji Hua Laboratory, Foshan 528251, China
[3] Engineering Research Center of AI and Robotics, Ministry of Education, Shanghai 200433, China
[4] Institute of Meta-Medical, Fudan University, Shanghai 200433, China
[5] Jilin Provincial Key Laboratory of Intelligence Science and Engineering, Changchun 130013, China
* Correspondence: pzhai@fudan.edu.cn (P.Z.); lihuazhang@fudan.edu.cn (L.Z.)

Abstract: Many real-world applications can be described as large-scale games of imperfect information, which require extensive prior domain knowledge, especially in competitive or human–AI cooperation settings. Population-based training methods have become a popular solution to learn robust policies without any prior knowledge, which can generalize to policies of other players or humans. In this survey, we shed light on population-based deep reinforcement learning (PB-DRL) algorithms, their applications, and general frameworks. We introduce several independent subject areas, including naive self-play, fictitious self-play, population-play, evolution-based training methods, and the policy-space response oracle family. These methods provide a variety of approaches to solving multi-agent problems and are useful in designing robust multi-agent reinforcement learning algorithms that can handle complex real-life situations. Finally, we discuss challenges and hot topics in PB-DRL algorithms. We hope that this brief survey can provide guidance and insights for researchers interested in PB-DRL algorithms.

Keywords: reinforcement learning; multi-agent reinforcement learning; self play; population play

MSC: 68T42

Citation: Long, W.; Hou, T.; Wei, X.; Yan, S.; Zhai, P.; Zhang, L. A Survey on Population-Based Deep Reinforcement Learning. *Mathematics* **2023**, *11*, 2234. https://doi.org/10.3390/math11102234

Academic Editors: Jian Dong and Marjan Mernik

Received: 31 March 2023
Revised: 7 May 2023
Accepted: 8 May 2023
Published: 10 May 2023

1. Introduction

Reinforcement learning (RL) [1] is a highly active research field in the machine learning community with decades of development. However, traditional RL methods have limited performance when it comes to complex, high-dimensional input spaces. Deep reinforcement learning (DRL) [2] addresses this issue by using deep neural networks as function approximators, allowing agents to use unstructured data for decision-making. DRL has shown impressive performance on a range of tasks, including game playing, robotics, and autonomous driving. There are many impressive research works from different fields which were achieved through DRL, such as gaming (AlphaGo [3], AlphaZero [4], AlphaStar [5]), nuclear energy (fusion control [6]), and mathematics (AlphaTensor [7]). While DRL has become increasingly popular due to its effectiveness and generality, there are many real-world applications that require multiple agents' cooperation or competition. A multi-agent system is usually employed to research problems that are difficult or impossible for a single agent. Multi-agent reinforcement learning (MARL) is one of the effective approaches to multi-agent system problems [8]; it has been used to address problems in a variety of domains, including robotics, distributed control, telecommunications, and economics [9]. However, many real-world applications can be described as large-scale games of imperfect information, which require a lot of prior domain knowledge to compute a Nash equilibrium, especially in a competitive environment. This can be a major challenge for traditional MARL methods. Population-based reinforcement learning (PB-DRL) has emerged as a popular solution, allowing for the training of robust policies without any prior domain knowledge that can generalize to all policies of other players.

Population-based approaches take advantage of the high parallelism and large search space typically found in optimization problems. This method has demonstrated remarkable performance in MARL, resulting in exceptional performance in games such as Pluribus [10] and OpenAI Five [11] without any expert experience. Czarnecki et al. [12] conducted research on the importance of population-based training techniques for large-scale multi-agent environments, which can include both virtual and real-world scenarios. According to their study, population-based methods offer several benefits for training robust policies in such environments, including diversity of strategies, scalability, and adaptability to changing conditions. The population diversity in PB-DRL allows for a more robust policy because it can handle a wider range of situations and scenarios. This can be particularly important in real-world applications where there may be plenty of variability and uncertainty. By using a population-based approach, the policy can be trained to be more robust and adaptable to different situations, which is crucial for success in real-world applications.

In contrast to prior surveys, the motivation of our survey lies instead in recent PB-DRL algorithms and applications specifically, which helps to achieve surprisingly outstanding performance. Accordingly, we also introduce general frameworks of PB-DRL. In this survey, we give an account of PB-DRL approaches and associated methods. We start with reviewing selected ideas from game theory and multi-agent reinforcement learning. Then, we move on to present several recent promising approaches, applications, and frameworks of PB-DRL. Here, we first give the idea of milestones and briefly describe others in each kind of PB-DRL; applications for how to use the corresponding methods will then be introduced. Finally, we finish by discussing challenges and hot topics of PB-DRL. Given the extensive literature from the MARL community and adjacent fields, we acknowledge that our survey is not exhaustive and may not cover all prior work. Specifically, in this survey, our focus is on PB-DRL algorithms including multi-agent reinforcement learning by using self-play-related technology, evolution-based training methods for reinforcement learning, and general frameworks. We conducted a comprehensive literature search following the guidelines for Systematic Literature Reviews (SLRs) [13] to conduct a comprehensive literature search on PB-DRL. We searched four databases that are widely used and recognized in the field of software engineering: Google Scholar, IEEE Xplore, ACM Digital Library, and DataBase Systems and Logic Programming. We used advanced search options to limit the results to peer-reviewed articles published in English from 2018 to 2023, as this survey is focused more on recent works and we used snowballing method to cover previous milestones. We used the following search string: ("population-based" AND "reinforcement learning") OR ("evolution algorithm" AND "reinforcement learning") OR ("self-play" AND "reinforcement learning"). The initial search yielded 200 papers from Google Scholar (capturing the first 20 pages of search results), 127 papers from IEEE Xplore, 381 papers from ACM Digital Library, and 80 papers from DataBase Systems and Logic Programming, resulting in a total of 788 papers before screening. We screened the titles and abstracts of these papers based on their relevance to our survey topic, which is PB-DRL methods and applications. We used the following inclusion criteria: (1) the paper must focus on PB-DRL or a related concept (e.g., evolutionary algorithm, self-play); (2) the paper must report novel approaches, significant results, or comparative evaluations related to PB-DRL. To ensure the high quality of the references, we also screened based on the publisher; famous conferences and journals such as Nature, Science, NeuralPS, ICML, and ICLR were included. After screening, we included nearly 30 papers for further analysis and excluded others for various reasons such as being out of scope, being duplicates, or being of low quality. We then used snowballing to complement our database search and ensure that we did not miss any relevant studies, i.e., we checked the references of the included papers to identify any additional relevant studies. Our aim is to provide an overview of recent developments in PB-DRL, and we hope that it can garner more attention and be applied in various industries.

2. Background

Population-based deep reinforcement learning is an approach that addresses the limitations of traditional reinforcement learning algorithms. PB-DRL algorithms maintain a population of agents that explore the environment in parallel and learn from each other to improve their collective performance. This approach has shown significant improvements in terms of sample efficiency and generalization in various applications. In this section, we start with necessary knowledge which may help to understand PB-DRL algorithms.

2.1. Game Theory

Game theory and MARL are closely related fields, as game theory provides a theoretical framework for analyzing and understanding strategic interactions between multiple agents, while MARL provides a practical approach for agents to learn optimal strategies through experience. Research in this survey usually considers a normal-form game or extensive-form game. A normal-form game refers to a game where players make decisions simultaneously without knowing the decisions made by other players, whereas an extensive-form game refers to a game where players make decisions sequentially and can observe the decisions made by other players before making their own decisions.

Normal-form games represent the game by way of a matrix, represented by a tuple $(\boldsymbol{\pi}, U, n)$, where n is the number of players, $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n)$ is a collection of strategies for all players, and $U : \boldsymbol{\pi} \rightarrow R^n$ is a payoff table mapping each joint policy to a scalar utility for each player. Each player aims to maximize their own payoff. Assume that π_i is a mixed strategy of player i and π_{-i} refers to the joint mixed strategies except π_i. $U_i(\pi_i, \pi_{-i})$ is the expected payoff of player i. Then, Nash equilibrium can be defined.

Definition 1 (Nash equilibrium)**.** *A mixed strategy profile* $\boldsymbol{\pi}^* = (\pi_1^*, \ldots, \pi_n^*)$ *is a Nash equilibrium if for all player i:*

$$\max_{\pi_i'} U_i(\pi_i', \pi_{-i}) = U_i(\pi_i^*, \pi_{-i})$$

Intuitively, in a Nash equilibrium, no player has an incentive to change their current strategy unilaterally because doing so would result in no benefits or even negative returns. Therefore, the strategy profile remains stable. π_i^* is the best response (BR) of agent i.

In extensive-form games, players make decisions sequentially, with each player's action influencing the subsequent decisions of other players. These games generalize the normal-form game formalism for sequential decision-making scenarios. Every finite extensive-form game has an equivalent normal-form game [14], and an approximation of Nash equilibrium called ϵ-Nash equilibrium or approximate Nash equilibrium is typically considered. The corresponding BR is referred to as the ϵ-best response.

Definition 2 (ϵ-Nash equilibrium)**.** *A mixed strategy profile* $\boldsymbol{\pi}^* = (\pi_1^*, \ldots, \pi_n^*)$ *is a* ϵ*-Nash equilibrium if for all player i:*

$$U_i(\pi_i^*, \pi_{-i}) \geq U_i(\pi_i', \pi_{-i}) - \epsilon, \quad \textit{for any policy } \pi_i' \textit{ of player } i$$

In the context of game theory, a real-life game can be incredibly complex, making it impractical to explicitly list out all the possible strategies. As a result, researchers often construct an "empirical game" that is smaller in size but still captures the essential features of the full game. This empirical game is built by discovering strategies and using meta-reasoning techniques to navigate the strategy space. In the general framework of PB-DRL algorithms, this empirical game is also known as the "meta-game." It starts with a single policy and grows by adding policies that approximate the best responses to the meta-strategy of the other players. In other words, the meta-game is a simplified version of the real game that captures its essential features, allowing PB-DRL algorithms to learn effective strategies in a computationally efficient manner.

2.2. Multi-Agent Reinforcement Learning

Reinforcement learning is a learning approach that aims to find the optimal way of mapping situations to actions to maximize a numerical reward signal [1]. It is often formalized as a Markov Decision Process (MDP) that addresses sequential decision-making problems in environments with discrete time-steps. Deep reinforcement learning combines RL with deep neural networks, resulting in improved performance compared to RL without neural networks. DRL can handle larger state spaces, enabling it to process larger input matrices such as images, and it can learn more complex policies with fewer hand-specified features to represent state information [15,16]. Some well-known DRL algorithms used in various applications include DQN [17], TD3 [18], and PPO [19].

Multi-agent reinforcement learning refers to sequential decision-making with multiple agents, which poses additional challenges due to the fact that taking action can influence the rewards of other agents. MARL tasks are commonly divided into cooperative (the agents work together to reach a common goal, like Overcooked), competitive (each agent has its own reward function and acts selfishly to maximize only its own expected cumulative reward, like Go), and mixed settings (each agent has an arbitrary but agent-unique reward function, like football). PB-DRL has shown good performance in applications with these settings [4,20,21].

MARL algorithms can be classified into six different learning paradigms, as outlined by Yang et al. [22]: (1) independent learners with shared policy, (2) independent learners with independent policies, (3) independent learning with shared policy within a group, (4) one central controller controlling all agents, (5) centralized training with decentralized execution (CTDE), and (6) decentralized training with networked agents. Types with independent learners (type 1-3) use independent reinforcement learning, where each agent treats the experience of other agents as part of its (non-stationary) environments [23,24]. This makes it a suitable approach for problems with a large number of agents or a high-dimensional action space. However, it may result in overfitting to the other agents during training and insufficient generalization during execution [25]. The fourth type of the learning paradigm can be seen as a single-agent RL method, which has a problem in large-scale strategy space. Type 5, CTDE, is a popular approach where agents can exchange information with others during training and act independently during execution [26,27]. This allows for efficient training but may suffer from communication overhead during execution. Decentralized training with networked agents, type 6, involves each agent learning its own policy based on local observations and communication with its neighbors in a networked structure [28]. This can improve scalability and adaptability but may require significant communication among agents.

PB-DRL algorithms belong to the CTDE paradigm and maintain a large, diverse population to train a robust policy that can generalize to non-communication situations. In PB-DRL, a population of agents explores the environment in parallel, allowing them to learn from each other to improve their collective performance. This approach has been shown to be more sample-efficient and to generalize better in various applications, including cooperative, competitive, and mixed settings, which makes it a promising technique for addressing the challenges in multi-agent reinforcement learning.

3. Population-Based Deep Reinforcement Learning

In recent years, PB-DRL has emerged as a promising research direction in the field of DRL. The key idea behind PB-DRL is to employ multiple agents or learners that interact with their environment in parallel and exchange information to improve their performance. This approach has shown great potential in achieving superior results compared to traditional single-agent methods. In this survey paper, we focus on several popular population-based methods, including naive self-play, fictitious self-play, population play, evolutionary-based methods, and the general framework. We will discuss the basic concepts, advantages, and limitations of each method to provide a comprehensive overview of the current state of the

art in this field. For a brief overview of a selected subset of methods for PB-DRL, see also Table 1.

Table 1. A selected subset of research areas and recent algorithms or frameworks for PB-DRL.

Category	Algorithm	Advantages	Disadvantages	Descriptions
Naive Self-Play	Naive SP [29]	Simplicity, Effectiveness	Overfitting, Instability	Playing against a mirrored copy of the agent
Fictitious Self-Play	Fictitious play [30]	Flexibility	Exploration requirement	Players choose the best response to a uniform mixture of all previous policies at each iteration
	FSP [31]	Robust and efficient learning	Exploration requirement, Sensitive initialization	Extending FP to extensive-form games
	NFSP [32]	Handling complex environments, High scalability	Instability, Hyperparameter tuning	Combining FSP with neural network function approximation
	ED [33]	Efficient, No requirement of average strategies	Exploration requirement, Scalability issues	Directly optimizes policies against worst-case opponents
	δ-Uniform FSP [34]	Simplicity	Limited Exploration	Learning a policy that can beat older versions of itself sampled uniformly at random
	Prioritized FSP [5]	Simplicity	Limited Exploration	Sampling the policies of opponents by their expected win rate
Population-Play	PP [35]	Exploration, Diversity	Inefficiency	Training a population of agents, all of whom interact with each other
	FCP [20]	Zero-shot collaboration	Inefficiency, Prone to researcher biases	Using PP to train a diversity policy pool of partners which is used to train a robust agent in cooperative setting
	Hidden-Utility Self-Play [36]	Modeling human bias	Inefficiency, Domain knowledge requirement	Following the FCP framework and uses a hidden reward function to model human bias to human–AI cooperation problem
Evolution-based Method	PBT [37]	Efficient search, Dynamic hyperparameter tuning	Resource-intensive, Complex implementation	Online evolutionary process that adapts internal rewards and hyperparameters
	MERL [38]	No requirement for reward sharping	Computationally expensive	Split-level training platform without requiring domain-specific reward shaping
	CERL [39]	Efficient sampling	Resource-intensive	Using a collective replay buffer to share all information across the population
	DERL [40]	Diverse solution	Computationally expensive	Decoupling the processes of learning and evolution in a distributed asynchronous manner
General Framework	PSRO [25]	Robust learning, Zero-sum convergence	Low scalability, Computationally expensive	Using DRL to compute best responses to a distribution over policies and empirical game-theoretic analysis to compute new meta-strategy distributions
	$PSRO_{rN}$ [41]	Open-ended learning, Non-transitive cycle	Low scalability, Computationally expensive	A framework based on PSRO but defines and uses the effective diversity to encourage diverse skills
	Diverse PSRO [42]	Open-ended learning, Low exploitability	Low scalability, Computationally expensive	Introducing a new diversity measure based on a geometric interpretation of games modelled by a determinantal point process to improve diversity
	Pipeline PSRO [43]	Scalability, High efficiency	Approximation errors	Maintaining a hierarchical pipeline of reinforcement learning workers to improve the efficiency of PSRO
	α-PSRO [44]	General-sum many-player game	Resource-intensive, Solver dependence	Using an α-Rank method as the meta-solver which is a critical component of PSRO

3.1. Naive Self-Play

Self-play (SP) is an open-ended learning training scheme that trains by playing against a mirrored copy of itself without any supervision in various stochastic environments. Compared with expert opponents, SP has shown more amazing performance in many complex problems. The simplest and most effective SP method is naive self-play, first proposed in [29]. As shown in Figure 1, the opponent (mirrored agent) uses the same policy network, i.e., the opponent downloads the latest policy network while the agent updates its policy network. Denote π as a policy being trained, $\boldsymbol{\pi^{zoo}}$ as a policy zoo, π' as the policy

set of the opponents, Ω as the policy sampling distribution, and G as the gating function for $\boldsymbol{\pi}^{zoo}$ [45]. The policy sampling distribution Ω is

$$\Omega(\pi'|\boldsymbol{\pi}^{zoo}, \pi) = \begin{cases} 1, & \forall \pi' \in \boldsymbol{\pi}^{zoo} : \pi' = \pi \\ 0. & Otherwise \end{cases} \tag{1}$$

Since the policy zoo $\boldsymbol{\pi}^{zoo}$ only keeps the latest version of policy π, it always clears the old policies $\boldsymbol{\pi}^{zoo}$ and inserts π, $\boldsymbol{\pi}^{zoo} = \pi$.

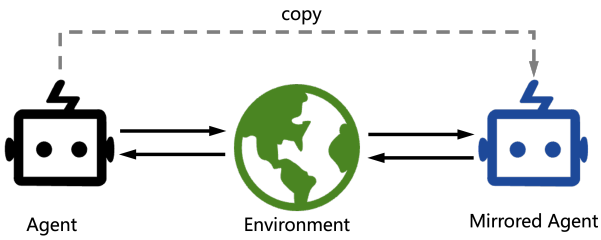

Figure 1. Overview of naive self-play.

A variety of works have followed this method since naive self-play is simple and effective. TD-Gammon [46] features naive SP to learn a policy by using TD(λ) algorithm. At that time, this work outperforms supervised learning with expert experience. AlphaGo [3] defeated the world champion of Go in 2017; it uses a combination of supervised learning on expert datasets and SP technology. SP is used to update the policy and to generate more data. SP-based applications have been developed rapidly in both academia and industry. One year after AlphaGo, AlphaZero [47] gained prominence. In contrast to AlphaGo, AlphaZero does not require domain-specific human knowledge but achieves outstanding performance. Instead, it learns the game policy by playing against itself, using only the game rules.

Naive SP is also a solution for handling many-to-many environments, as demonstrated by JueWu [48] which uses this approach for two players controlling five heroes in Honor of Kings during lineup and random lineup stages. Another study applied naive SP to an open-ended environment (hide-and-seek [49]), showing that it can lead to emergent auto-curricula with many distinct and compounding phase shifts in agent strategy.

Despite its effectiveness, naive SP may not be sufficient to learn a robust policy due to the lack of diversity in opponent policies. Fictitious self-play is a solution to this problem, where the agent plays against a mixture of its previous policies and fictional policies that are generated by sampling from a distribution over policies learned during training.

3.2. Fictitious Self-Play

Fictitious play, introduced by Brown [30], is a popular method for learning Nash equilibrium in normal-form games. The premise is that players repeatedly play a game and choose the best response to a uniform mixture of all previous policies at each iteration. As shown in Figure 2, fictitious self-play (FSP) [31] is a machine learning framework that implements generalized weakened fictitious play in behavioral strategies in a sample-based fashion. It can avoid cycles by playing against all previous policies. FSP iteratively samples episodes of the game from SP. These episodes constitute datasets that are used by reinforcement learning to compute approximate best responses and by supervised learning to compute perturbed models of average strategies.

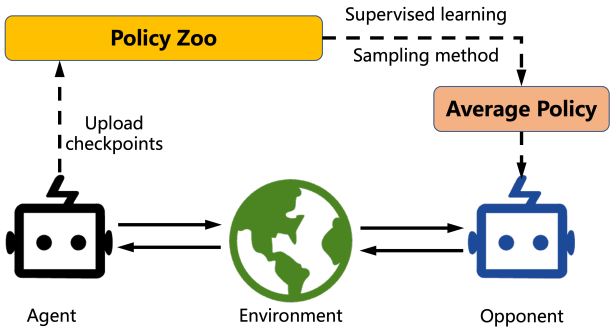

Figure 2. Overview of fictitious self-play.

Neural fictitious self-play (NFSP) [32] combines FSP with neural network function approximation. NFSP keeps two kinds of memories. One, denoted as $\mathcal{M}_{RL}$, was used for storing experience of game transitions, while the other, $\mathcal{M}_{SL}$, stored the best response behavior. Each agent computed an approximate best response β from $\mathcal{M}_{RL}$ and updated its average policy Π by supervised learning from $\mathcal{M}_{SL}$. In principle, each agent could learn the best response by playing against the average policies of other agents. However, the agent cannot get its best response policy β, which is needed to train its average policy Π, and its average policy Π is needed for the best response training of other agents. NFSP uses the approximation of anticipatory dynamics of continuous-time dynamic fictitious play [50], in which players choose the best response to the short-term predicted average policy of their opponents, $\Pi_t^{-i} + \eta \frac{d}{dt}\Pi_t$, where η is the anticipatory parameter. NFSP assumes $\beta_{t+1} - \Pi_t \approx \frac{d}{dt}\Pi_t$ as a discrete-time approximation. During play, all agents mixed their actions according to $\sigma = \Pi + \eta(\beta - \Pi)$. By using this approach, each agent could learn an approximate best response with predicted average policies of its opponents. In other words, the policy sampling distribution of all agents Ω is

$$\Omega(\pi) = \begin{cases} \beta, & \textit{with probability } \eta \\ \Pi. & \textit{with probability } 1-\eta \end{cases} \tag{2}$$

M_{RL} uses a circular buffer to store transition in every step, but M_{SL} only inserts transition while agent follows the best response policy β.

The Exploitability Descent (ED) algorithm [33] is a PB-DRL method that directly optimizes policies against worst-case opponents without the need to compute average policies. In contrast to NFSP algorithm, which requires a large reservoir buffer to compute an approximate equilibrium, ED focuses on decreasing the "exploitability" of each player, which refers to how much a player could gain by switching to a best response. The algorithm has two steps for each player on each iteration. The first step is identical to the FP algorithm, where the best response to the policy of each player is computed. The second step performs gradient ascent on the policy to increase the utility of each player against the respective best responder, aiming to decrease the exploitability of each player. In a tabular setting with Q-values and L2 projection, the policy gradient ascent update is defined by equation

$$\begin{aligned} \boldsymbol{\theta}_S^t &= P_{\ell_2}(\boldsymbol{\theta}_S^{t-1} + \alpha^t \langle \nabla_{\boldsymbol{\theta}_S} \boldsymbol{\pi}_{\boldsymbol{\theta}}^{t-1}(S), \boldsymbol{Q}^{\boldsymbol{b}}(S) \rangle) \\ &= P_{\ell_2}(\boldsymbol{\theta}_S^{t-1} + \alpha^t \boldsymbol{Q}^{\boldsymbol{b}}(S)), \end{aligned} \tag{3}$$

where $\boldsymbol{Q}^{\boldsymbol{b}}(S)$ is the expected return at state S with joint policy set $\boldsymbol{b}$, P_{ℓ_2} is the L2 projection, $\nabla_{\boldsymbol{\theta}_S} \boldsymbol{\pi}_{\boldsymbol{\theta}}^{t-1}(S)$ is an identity matrix, and α is the step size. In other words, the ED algorithm directly optimizes policies against worst-case opponents, making it a promising approach for addressing games with complex strategy spaces.

A related approach from another perspective is $\delta - Uniform$ FSP [34], which learns a policy that can beat older versions of itself sampled uniformly at random. The authors use a percentage threshold $\delta \in [0, 1]$ to select the old policies that are eligible for sampling from the policy zoo $\boldsymbol{\pi}^{zoo}$, i.e., the opponent strategy π' is sampled from

$$\Omega(\pi' | \boldsymbol{\pi}^{zoo}, \pi) = Uniform(\delta | \boldsymbol{\pi}^{zoo} |, | \boldsymbol{\pi}^{zoo} |) \tag{4}$$

Significantly, the algorithm is the same as naive SP while $\delta = 1$. After every episode, the training policy is always inserted into the policy zoo $\boldsymbol{\pi}^{zoo}$. Thus, $\boldsymbol{\pi}^{zoo}$ is updated with $\boldsymbol{\pi}^{zoo} = \boldsymbol{\pi}^{zoo} \cup \pi$.

While AlphaStar does use FSP as one of its learning algorithms, Prioritized FSP is actually a modification proposed by the AlphaStar team in their subsequent paper [5]. The authors argue that many games are wasted against players that are defeated in almost 100% of games while using regular FSP and propose Prioritized FSP which samples policies by their expected win rate. Policies that are expected to win with higher probability against the current agent have higher priority and are sampled more frequently. The opponent sampling distribution Ω can be written as

$$\Omega(\pi' | \boldsymbol{\pi}^{zoo}, \pi) = \frac{f(P(\pi \ beats \ \pi'))}{\sum_{\Pi \in \boldsymbol{\pi}^{zoo}} f(P(\pi \ betas \ \Pi))} \tag{5}$$

where f is a weighting function, e.g., $f(x) = (1 - x)^p$. The policy zoo named league in the paper is complex; we will introduce the update method latter.

OpenAI Five also employs a similar method, as described in [11]. The method consists of training with a naive self-play approach for 80% of the games and using past sampling policies for the remaining 20%. Similar to the Prioritized FSP method, OpenAI Five uses a dynamic sampling system that relies on a dynamically generated quality score q. This system samples opponent agents according to a softmax distribution, where the probability of choosing an opponent p is proportional to e^q. If OpenAI Five wins the game, q is updated with a learning rate constant η as follows:

$$q = q - \frac{\eta}{Np} \tag{6}$$

where N is the size of policy zoo. At every 10 iterations, the policy of the current agent will be added to the policy zoo with an initial quality score equal to the maximum quality score in the zoo.

While self-play can bring remarkable performance improvements to reinforcement learning, it performs poorly in non-transitive games because it always plays against itself. Specifically, the opponent's policy only samples from one policy, which means the training agent only learns from a single type of opponent. This approach works well in situations where a higher-ranked player can always beat a lower-ranked player. Population-based training methods bring more robust policies.

3.3. Population-Play

Another population-based method for multi-agent systems is population-play (PP), which builds upon the concept of SP to involve multiple players and their past generations [5,35], as shown in Figure 3. With PP, a group of agents is developed and trained to compete not only with each other but also with agents from prior generations.

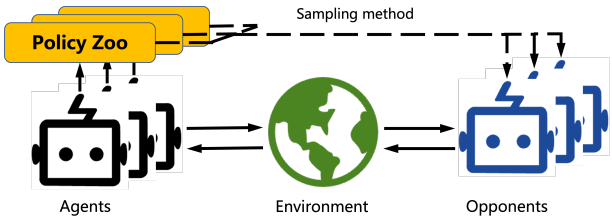

Figure 3. Overview of (naive) population-play.

To train an exceptional agent, AlphaStar [5] maintains three types of opponent pools: Main Agents, League Exploiters, and Main Exploiters. Main Agents are trained with a combination of 35% SP and 50% PFSP against all past players in the league, and the agent plays an additional 15% of matches against opponents who had previously been beaten but are now unbeatable, as well as past opponents who had previously exploited the weaknesses of the agent. League Exploiters are used to find a policy that league agents cannot defeat. They are trained using PFSP against agents in the league and added to the league if they defeat all agents in the league with a winning rate of more than 70%. Main Exploiters play against Main Agents to identify their weaknesses. If the current probability of winning is less than 20%, Main Exploiters employ PFSP against players created by Main Agents. Otherwise, Main Exploiters play directly against the current Main Agents.

For the Win (FTW) [35] is a training method designed for the game of Capture the Flag, which involves training a diverse population of different agents by having them learn from playing with each other. The training process involves sampling agents from the population to play as teammates and opponents, which is done using a stochastic matchmaking scheme that biases co-players to be of similar skill to the player. This ensures that a diverse set of teammates and opponents participate in training, and helps to promote robustness in the learned policies. A population-based training method is implemented to enhance the performance of weaker players and improve the overall ability of all players.

PP can accommodate a wide range of agents, making it also suitable for deployment in cooperative settings. However, Siu et al. [51] observed that in such scenarios, human players tended to favor rule-based agents over RL-based ones. This finding highlights the need to take into account human perceptions of AI when designing and developing systems intended for real-world adoption.

To address this issue, fictitious co-play (FCP) [20] aims to produce robust partners that can assist humans with different styles and skill levels without relying on human-generated data (i.e., zero-shot coordination with humans). FCP is a two-stage approach. In the first stage, N partner agents are trained independently in self-play to create a diverse pool of partners. In the second stage, FCP trains a best-response agent against the diverse pool to achieve robustness. Hidden-utility self-play [36] follows the FCP framework and uses a hidden reward function to model human bias with domain knowledge to solve the human–AI cooperation problem. A similar work for assistive robots learns a good latent representation for human policies [52].

3.4. Evolution-Based Training Methods

Evolutionary algorithms are a family of optimization algorithms inspired by the process of natural selection. They involve generating a population of candidate solutions and iteratively improving them by applying operators such as mutation, crossover, and selection, which mimic the processes of variation, reproduction, and selection in biological evolution. These algorithms are widely used in solving complex optimization problems in various fields, including engineering, finance, and computer science. Evolutionary-based DRL is a type of PB-DRL that approaches training from an evolutionary perspective and often incorporates swarm intelligence techniques, particularly evolution algorithms. In this subsection, we will focus on recent hybrid DRL algorithms that combine evolutionary

approaches with deep reinforcement learning to accelerate the training phase. These algorithms can be used alongside SP or PP algorithms [35].

Population-based training (PBT) introduced in [37] is an online evolutionary process that adapts internal rewards and hyperparameters while performing model selection by replacing underperforming agents with mutated versions of better agents. Multiple agents are trained in parallel, and they periodically exchange information by copying weights and hyperparameters. The agents evaluate their performance, and underperforming agents are replaced by mutated versions of better-performing agents. This process continues until a satisfactory performance is achieved, or a maximum budget is reached.

Majumdar et al. [38] propose multi-agent evolutionary reinforcement learning (MERL) as a solution for the sample inefficiency problem of PBT in cooperative MARL environments where the team reward is sparse and agent-specific reward is dense. MERL is a split-level training platform that combines both gradient-based and gradient-free optimization methods, without requiring domain-specific reward shaping. The gradient-free optimizer is used to maximize the team objective by employing an evolutionary algorithm. Specifically, the evolutionary population maintains a variety of teams and uses evolutionary algorithms to maximize team rewards (fitness). The gradient-based optimizer maximizes the local reward of each agent by using a common replay buffer with other team members in the evolutionary population. Collaborative evolutionary reinforcement learning (CERL) [39] is a similar work which addresses the sample inefficiency problem of PBT. It uses a collective replay buffer to share all information across the population.

Deep evolutionary reinforcement learning (DERL) [40] is a framework for creating embodied agents that combines evolutionary algorithms with DRL, which aims to find a diverse solutions. DERL decouples the processes of learning and evolution in a distributed asynchronous manner, using tournament-based steady-state evolution. Similar to PBT [37], DERL maintains a population to encourage diverse solutions. The average final reward is used as a fitness function, and a tournament-based selection method is used to choose the parents for generating children via mutation operations. Liu et al. [53] demonstrated that end-to-end PBT can lead to emergent cooperative behaviors in the soccer domain. They also applied an evaluation scheme based on Nash averaging to address the diversity and exploitability problem.

3.5. General Framework

The policy-space response oracles (PSRO) framework is currently the most widely used general framework for PB-DRL. It unifies various population-based methods, such as SP and PP, with empirical game theory to effectively solve games [25]. As shown in Figure 4, PSRO divides these algorithms into three modules: meta strategy, best-response solution, and policy zoo expansion. The first module, meta strategy, involves solving the meta-game using a meta-solver to obtain the meta strategy (policy distribution) of each policy zoo. The second module, best-response solution, involves each agent sampling policies of other agents π_{-i} and computing its best response π_i with fixed π_{-i}. The third module, policy zoo expansion, involves adding the best response to the corresponding policy zoo. The process starts with a single policy. In each episode, one player trains its policy π_i using a fixed policy set, which is sampled from the meta-strategies of its opponents ($\boldsymbol{\pi'_{-i}} \sim \boldsymbol{\pi^{zoo}_{-i}}$). At the end of every epoch, each policy zoo expands by adding the approximate best response to the meta-strategy of the other players, and the expected utilities for new policy combinations computed via simulation are added to the payoff matrix.

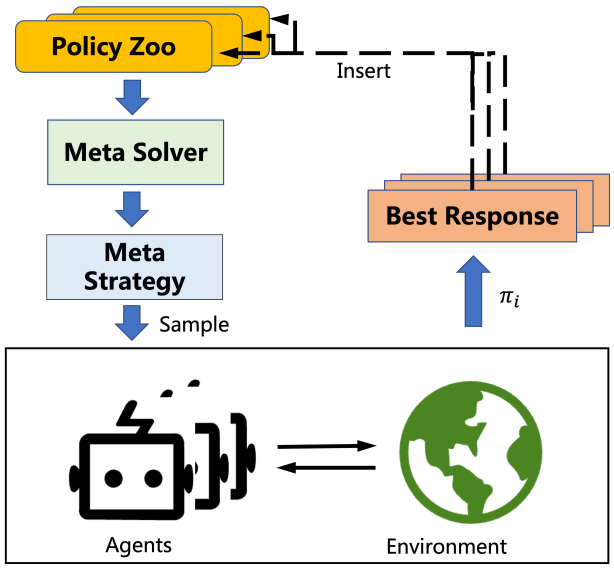

Figure 4. Overview of PSRO.

Although PSRO has demonstrated its performance, several drawbacks have been identified and addressed by recent research. One such extension is Rectified Nash response (PSRO_{rN}) [41], which addresses the diversity issue and introduces adaptive sequences of objectives that facilitate open-ended learning. The effective diversity of the population is defined as:

$$d(\boldsymbol{\pi}^{zoo}) = \sum_{i,j=1}^{n} \lfloor \phi(w_i, w_j) \rfloor_+ \cdot p_i \cdot p_j \tag{7}$$

where $n = |\boldsymbol{\pi}^{zoo}|$, $\phi(x, y)$ is the payoff function, $\boldsymbol{p}$ is the Nash equilibrium on $\boldsymbol{\pi}_{zoo}$, $\lfloor x \rfloor_+$ is the rectifier, denoted by $\lfloor x \rfloor_+ = x$ if $x \leq 0$ and $\lfloor x \rfloor_+ = 0$ otherwise. Equation (7) encourages agents to play against opponents who they can beat. Perhaps surprisingly, the authors found that building objectives around the weaknesses of agents does not actually encourage diverse skills. To elaborate, when the weaknesses of an agent are emphasized during training, the gradients that guide its policy updates will be biased towards improving those weaknesses, potentially leading to overfitting to a narrow subset of the state space. This can result in a lack of diversity in the learned policies and a failure to generalize to novel situations. Several other works have also focused on the diversity aspect of PSRO frameworks. In [42], the authors propose a geometric interpretation of behavioral diversity in games (Diverse PSRO) and introduce a novel diversity metric that uses determinantal point process (DPP). The diversity metric is based on the expected cardinality of random samples from a DPP in which the ground set is the strategy population. It is denoted as:

$$Diversity(\boldsymbol{\pi}^{zoo}) = E_{\boldsymbol{\pi}' \sim \mathbb{P}_{L_{\boldsymbol{\pi}^{zoo}}}} \left[|\boldsymbol{\pi}'| \right] = Tr(\boldsymbol{I} - (L_{\boldsymbol{\pi}^{zoo}} + \boldsymbol{I})^{-1}), \tag{8}$$

where a DPP defines a probability $\mathbb{P}$, $\boldsymbol{\pi}'$ is a random subset drawn from the DPP, and $L_{\boldsymbol{\pi}^{zoo}}$ is the DPP kernel. They incorporate this diversity metric into best-response dynamics to improve overall diversity. Similarly, [54] notes the absence of widely accepted definitions for diversity and offers a redefined behavioral diversity measure. The authors propose response diversity as another way to characterize diversity through the response of policies when facing different opponents.

Pipeline PSRO [43] is a scalable method that aims to improve the efficiency of PSRO, which is a common problem of most of PSRO-related frameworks, in finding approximate Nash equilibrium. It achieves this by maintaining a hierarchical pipeline of reinforcement learning workers, allowing it to parallelize PSRO while ensuring convergence. The method includes two classes of policies: fixed and active. Active policies are trained in a hierarchical pipeline, while fixed policies are not trained further. When the performance improvement

of the lowest-level active worker in the pipeline does not meet a given threshold within a certain time period, the policy becomes fixed, and a new active policy is added to the pipeline. Another work has improved the computation efficiency and exploration efficiency by introducing a new subroutine of no-regret optimization [55].

PSRO framework has another branch which optimizes the meta-solver concept. Alpha-PSRO [44] extends the original PSRO paper to apply readily to general-sum, many-player settings, using an α-Rank [56], a ranking method that considers all pairwise comparisons between policies, as the meta-solver. Alpha-PSRO defines preference-based best response (PBR), an oracle that finds policies that maximize their rank against the population. Alpha-PSRO works by expanding the strategy pool through constructing a meta-game and calculating a payoff matrix. The meta-game is then solved to obtain a meta-strategy, and finally, a best response is calculated to find an approximate optimal response. Joint PSRO [57] uses correlated equilibrium as the meta-solver, and Mean-Field PSRO [58] proposes newly defined mean-field no-adversarial-regret learners as the meta-solver.

4. Challenges and Hot Topics

In the previous section, we discussed several PB-DRL algorithms that have shown significant improvements in real-life game scenarios. However, the application of these algorithms also faces several challenges that need to be addressed to further advance the field of PB-DRL.

4.1. Challenges

One of the most significant challenges in PB-DRL is the need for increased diversity within the population. Promoting diversity not only helps AI agents avoid checking the same policies repeatedly, but also enables them to discover niche skills, avoid being exploited, and maintain robust performance when encountering unfamiliar types of opponents [22]. As the population grows, it becomes more challenging to maintain diversity and ensure efficient exploration of the search space. Without adequate diversity, the population may converge prematurely to suboptimal solutions, leading to the stagnation of the learning process. Overfitting to policies in the policy zoo is a significant challenge to generalization [25,59]. Although the diversity of a population has been widely discussed in the evolutionary algorithm community at the genotype level, phenotype level, and the combination of the previous two cases [60], which typically operate on a fixed set of candidate solutions, PB-DRL is often used in dynamic and uncertain environments where the population size and diversity can change over time. Additionally, since policies are always represented as neural networks, using difference-based or distance-based methods directly, which are widely used in evolutionary computations, are not suitable choices. Some heuristic algorithms have been proposed. Balduzzi et al. [41] design an opponents selection method to expand the policy game space to improve diversity. Another approach is to incorporate different levels of hierarchy within the population to maintain diversity [44]. An interesting work [42] models behavioral diversity for learning in games by using a determinantal point process as the diversity metric. Other techniques that improve the diversity of the policy pool can be found in [61–63].

The need for increased efficiency is a significant challenge in PB-DRL, as evaluating each individual within a growing population becomes computationally expensive, resulting in a reduced learning rate. PB-DRL is often applied to large-scale environments with high-dimensional state and action spaces, making the evaluation of each individual within a population even more computationally expensive. For instance, AlphaStar trained the league over a period of 44 days using 192 8-core TPUs, 12 128-core TPUs, and 1800 CPUs, which potentially cost more than 120 billion dollars in renting cloud computing services for training [5]. One promising approach to improving efficiency in PB-DRL is to develop more sample-efficient algorithms. This can be achieved through various means, such as monotonic improvement in exploitability [55], regret bound [64]. Another approach to improving efficiency in PB-DRL is to use distributed computing techniques [43,65]. These

techniques can enable faster evaluation of individuals within a population, as well as better parallelization of the learning process. For example, some recent works named distributed deep reinforcement learning [66] are often used to accelerate the training process in PB-DRL, such as SEED RL [67], Gorila [68], and IMPALA [69]. In addition to the technical challenges of improving efficiency in PB-DRL, there are also practical challenges related to the cost and availability of computing resources. One possible solution to this challenge is to develop more energy-efficient algorithms that can run on low-power devices or take advantage of specialized hardware, such as GPUs or TPUs. Flajolet et al. [70] indicate that the judicious use of compilation and vectorization allows population-based training to be performed on a single machine with one accelerator with minimal overhead compared to training a single agent.

4.2. Hot Topics

Despite these challenges, it is essential to note that this field is rapidly evolving. Currently, there are several hot topics and future directions in PB-DRL worth exploring, and researchers are actively engaged in these endeavors.

Games: PB-DRL has demonstrated outstanding performance in many games, including board games [47], online games [5,11], and more. As a result, game manufacturers have become interested in exploring several directions. These include:

1. AI bots that can learn to make decisions like humans, making them suitable for use in tutorials, hosting games, computer opponents, and more.
2. AI non-player characters that train agents to interact with players according to their own character settings, which can be used for virtual hosts, open-world RPG games, and other applications.
3. AI teammates that are designed to help and support human players in cooperative games or simulations. AI teammates can provide assistance, such as cover fire, healing, or completing objectives, to human players in cooperative games or simulations.

Zero-shot coordination: The zero-shot coordination (ZSC) problem refers to the situation where agents must independently produce strategies for a collaborative game that are compatible with novel partners not seen during training [63]. Population-based reinforcement learning has been used for this problem, starting with FCP [20], and there is ongoing research using the keywords "zero-shot human-AI coordination." Researchers aim to identify a sufficiently robust agent capable of effectively generalizing human policies. Many methods have been used in this problem, such as lifetime learning [71], population diversity [62,63], and model human bias [36].

Robotics: Reinforcement learning has become increasingly prevalent in the robotics field [72–74]. The use of PB-DRL has also expanded to robots, including robotic manipulation [75], assistance with robots [52], multi-robot planning [76], and robot table tennis [77]. In a recent study, it was shown that PB-DRL could generate varied environments [78], which is advantageous for developing robust robotics solutions.

Financial markets: Population-based algorithms and concepts have immense potential for use in financial markets and economic forecasting [79]. Despite the widespread use of MARL in financial trading, the application of PB-DRL to financial markets appears to be underutilized in both academic and industry-related research. This is partly due to the high demands placed on simulation environments when working with PB-DRL. Once an environment that meets the requirements is created, PB-DRL will show its power.

5. Conclusions

In this paper, we have provided a comprehensive survey of representative population-based deep reinforcement learning (PB-DRL) algorithms, applications, and general frameworks. We categorize PB-DRL research into the following areas: naive self-play, fictitious self-play, population-play, evolution-based training methods, and general framework. We compare the main ideas of different types of algorithms by summarizing the various types of PB-DRL algorithms and describing how they have been used in real-life applications.

Furthermore, we introduce evolution-based training methods to expound on common ways to adjust hyperparameters or accelerate training. General frameworks for PB-DRL are also introduced for different game settings, providing a general training process and theoretical proofs. Finally, we discuss the challenges and opportunities of this exciting field. We aim to provide a valuable reference for researchers and engineers working on practical problems.

Author Contributions: Conceptualization, W.L. and P.Z.; methodology, W.L. and P.Z.; software, T.H.; validation, X.W. and S.Y.; formal analysis, X.W.; investigation, W.L. and X.W.; resources, W.L.; writing—original draft preparation, W.L.; writing—review and editing, T.H., X.W. and P.Z; visualization, S.Y.; supervision, L.Z.; project administration, P.Z. and L.Z.; funding acquisition, L.Z. All authors have read and agreed to the published version of the manuscript.

Funding: The work reported in this paper was supported by the National Key R&D Program of China (Grant Number: 2021ZD0113502, 2021ZD0113503), Shanghai Municipality Science and Technology Major Project (Grant Number: 2021SHZDZX0103), and China Postdoctoral Science Foundation (Grant Number: BX20220071, 2022M720769), and Research on Basic and Key Technologies of Intelligent Robots (Grant Number: KEH2310017).

Data Availability Statement: Not applicable.

Acknowledgments: Many thanks to FDU IPASS Group for taking the time to proofread this article.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
2. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Graves, A.; Antonoglou, I.; Wierstra, D.; Riedmiller, M. Playing Atari with Deep Reinforcement Learning. *arXiv* **2013**, arXiv:1312.5602.
3. Silver, D.; Huang, A.; Maddison, C.J.; Guez, A.; Sifre, L.; Van Den Driessche, G.; Schrittwieser, J.; Antonoglou, I.; Panneershelvam, V.; Lanctot, M.; et al. Mastering the game of Go with deep neural networks and tree search. *Nature* **2016**, *529*, 484–489. [CrossRef] [PubMed]
4. Silver, D.; Hubert, T.; Schrittwieser, J.; Antonoglou, I.; Lai, M.; Guez, A.; Lanctot, M.; Sifre, L.; Kumaran, D.; Graepel, T.; et al. A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play. *Science* **2018**, *362*, 1140–1144. [CrossRef]
5. Vinyals, O.; Babuschkin, I.; Czarnecki, W.M.; Mathieu, M.; Dudzik, A.; Chung, J.; Choi, D.H.; Powell, R.; Ewalds, T.; Georgiev, P.; et al. Grandmaster level in StarCraft II using multi-agent reinforcement learning. *Nature* **2019**, *575*, 350–354. [CrossRef] [PubMed]
6. Degrave, J.; Felici, F.; Buchli, J.; Neunert, M.; Tracey, B.; Carpanese, F.; Ewalds, T.; Hafner, R.; Abdolmaleki, A.; de Las Casas, D.; et al. Magnetic control of tokamak plasmas through deep reinforcement learning. *Nature* **2022**, *602*, 414–419. [CrossRef] [PubMed]
7. Fawzi, A.; Balog, M.; Huang, A.; Hubert, T.; Romera-Paredes, B.; Barekatain, M.; Novikov, A.; R Ruiz, F.J.; Schrittwieser, J.; Swirszcz, G.; et al. Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature* **2022**, *610*, 47–53. [CrossRef]
8. Hernandez-Leal, P.; Kartal, B.; Taylor, M.E. A survey and critique of multiagent deep reinforcement learning. *Auton. Agents Multi-Agent Syst.* **2019**, *33*, 750–797. [CrossRef]
9. Buşoniu, L.; Babuška, R.; De Schutter, B. Multi-agent reinforcement learning: An overview. In *Innovations in Multi-Agent Systems and Applications-1*; Springer: Berlin/Heidelberg, Germany, 2010; pp. 183–221.
10. Brown, N.; Sandholm, T. Superhuman AI for multiplayer poker. *Science* **2019**, *365*, 885–890. [CrossRef]
11. Berner, C.; Brockman, G.; Chan, B.; Cheung, V.; Debiak, P.; Dennison, C.; Farhi, D.; Fischer, Q.; Hashme, S.; Hesse, C.; et al. Dota 2 with Large Scale Deep Reinforcement Learning. *arXiv* **2019**, arXiv:1912.06680.
12. Czarnecki, W.M.; Gidel, G.; Tracey, B.; Tuyls, K.; Omidshafiei, S.; Balduzzi, D.; Jaderberg, M. Real world games look like spinning tops. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 17443–17454.
13. Kitchenham, B.; Charters, S. *Guidelines for Performing Systematic Literature Reviews in Software Engineering*; Elsevier: Amsterdam, The Netherlands, 2007.
14. Kuhn, H. Extensive games and the problem of information. *Contributions to the Theory of Games*; Princeton University Press: Princeton, NJ, USA, 1953; p. 193.
15. LeCun, Y.; Bengio, Y.; Hinton, G. Deep learning. *Nature* **2015**, *521*, 436–444. [CrossRef] [PubMed]
16. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef] [PubMed]

17. Mnih, V.; Kavukcuoglu, K.; Silver, D.; Rusu, A.A.; Veness, J.; Bellemare, M.G.; Graves, A.; Riedmiller, M.; Fidjeland, A.K.; Ostrovski, G.; et al. Human-level control through deep reinforcement learning. *Nature* **2015**, *518*, 529–533. [CrossRef] [PubMed]
18. Fujimoto, S.; van Hoof, H.; Meger, D. Addressing Function Approximation Error in Actor-Critic Methods. In *35th International Conference on Machine Learning*; Dy, J., Krause, A., Eds.; PMLR: Cambridge, MA, USA, 2018; Volume 80, pp. 1587–1596.
19. Schulman, J.; Wolski, F.; Dhariwal, P.; Radford, A.; Klimov, O. Proximal policy optimization algorithms. *arXiv* **2017**, arXiv:1707.06347.
20. Strouse, D.; McKee, K.; Botvinick, M.; Hughes, E.; Everett, R. Collaborating with humans without human data. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 14502–14515.
21. Lin, F.; Huang, S.; Pearce, T.; Chen, W.; Tu, W.W. TiZero: Mastering Multi-Agent Football with Curriculum Learning and Self-Play. *arXiv* **2023**, arXiv:2302.07515.
22. Yang, Y.; Wang, J. An overview of multi-agent reinforcement learning from game theoretical perspective. *arXiv* **2020**, arXiv:2011.00583.
23. de Witt, C.S.; Gupta, T.; Makoviichuk, D.; Makoviychuk, V.; Torr, P.H.; Sun, M.; Whiteson, S. Is independent learning all you need in the starcraft multi-agent challenge? *arXiv* **2020**, arXiv:2011.09533.
24. Yu, C.; Velu, A.; Vinitsky, E.; Gao, J.; Wang, Y.; Bayen, A.; Wu, Y. The surprising effectiveness of ppo in cooperative multi-agent games. *Adv. Neural Inf. Process. Syst.* **2022**, *35*, 24611–24624.
25. Lanctot, M.; Zambaldi, V.F.; Gruslys, A.; Lazaridou, A.; Tuyls, K.; Pérolat, J.; Silver, D.; Graepel, T. A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R., Eds.; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 4190–4203.
26. Lowe, R.; Wu, Y.; Tamar, A.; Harb, J.; Abbeel, P.; Mordatch, I. Multi-Agent Actor-Critic for Mixed Cooperative-Competitive Environments. In Proceedings of the Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, Long Beach, CA, USA, 4–9 December 2017; Guyon, I., von Luxburg, U., Bengio, S., Wallach, H.M., Fergus, R., Vishwanathan, S.V.N., Garnett, R., Eds.; Curran Associates Inc.: Red Hook, NY, USA, 2017; pp. 6379–6390.
27. Rashid, T.; Samvelyan, M.; Schroeder, C.; Farquhar, G.; Foerster, J.; Whiteson, S. QMIX: Monotonic Value Function Factorisation for Deep Multi-Agent Reinforcement Learning. In *35th International Conference on Machine Learning*; Dy, J., Krause, A., Eds.; PMLR: Cambridge, MA, USA, 2018; Volume 80, pp. 4295–4304.
28. Sukhbaatar, S.; Szlam, A.; Fergus, R. Learning Multiagent Communication with Backpropagation. In Proceedings of the Advances in Neural Information Processing Systems 29: Annual Conference on Neural Information Processing Systems 2016, Barcelona, Spain, 5–10 December 2016; Lee, D.D., Sugiyama, M., von Luxburg, U., Guyon, I., Garnett, R., Eds.; Curran Associates Inc.: Red Hook, NY, USA, 2016; pp. 2244–2252.
29. Al, S. Some studies in machine learning using the game of checkers. *IBM J. Res. Dev.* **1959**, *3*, 210–229.
30. Brown, G.W. Iterative solution of games by fictitious play. *Act. Anal. Prod. Alloc.* **1951**, *13*, 374.
31. Heinrich, J.; Lanctot, M.; Silver, D. Fictitious self-play in extensive-form games. In *International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2015; pp. 805–813.
32. Heinrich, J.; Silver, D. Deep reinforcement learning from self-play in imperfect-information games. *arXiv* **2016**, arXiv:1603.01121.
33. Lockhart, E.; Lanctot, M.; Pérolat, J.; Lespiau, J.; Morrill, D.; Timbers, F.; Tuyls, K. Computing Approximate Equilibria in Sequential Adversarial Games by Exploitability Descent. In Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, 10–16 August 2019; Kraus, S., Ed.; pp. 464–470. [CrossRef]
34. Bansal, T.; Pachocki, J.; Sidor, S.; Sutskever, I.; Mordatch, I. Emergent Complexity via Multi-Agent Competition. In Proceedings of the International Conference on Learning Representations, Vancouver, BC, Canada, 30 April–3 May 2018.
35. Jaderberg, M.; Czarnecki, W.M.; Dunning, I.; Marris, L.; Lever, G.; Castaneda, A.G.; Beattie, C.; Rabinowitz, N.C.; Morcos, A.S.; Ruderman, A.; et al. Human-level performance in 3D multiplayer games with population-based reinforcement learning. *Science* **2019**, *364*, 859–865. [CrossRef]
36. Yu, C.; Gao, J.; Liu, W.; Xu, B.; Tang, H.; Yang, J.; Wang, Y.; Wu, Y. Learning Zero-Shot Cooperation with Humans, Assuming Humans Are Biased. *arXiv* **2023**, arXiv:2302.01605.
37. Jaderberg, M.; Dalibard, V.; Osindero, S.; Czarnecki, W.M.; Donahue, J.; Razavi, A.; Vinyals, O.; Green, T.; Dunning, I.; Simonyan, K.; et al. Population Based Training of Neural Networks. *arXiv* **2017**, arXiv:1711.09846v2.
38. Majumdar, S.; Khadka, S.; Miret, S.; Mcaleer, S.; Tumer, K. Evolutionary Reinforcement Learning for Sample-Efficient Multiagent Coordination. In *37th International Conference on Machine Learning*; Daumé, H., Singh, A., Eds.; PMLR: Cambridge, MA, USA, 2020; Volume 119, pp. 6651–6660.
39. Khadka, S.; Majumdar, S.; Nassar, T.; Dwiel, Z.; Tumer, E.; Miret, S.; Liu, Y.; Tumer, K. Collaborative Evolutionary Reinforcement Learning. In *36th International Conference on Machine Learning*; Chaudhuri, K., Salakhutdinov, R., Eds. PMLR: Cambridge, MA, USA, 2019; Volume 97, pp. 3341–3350.
40. Gupta, A.; Savarese, S.; Ganguli, S.; Fei-Fei, L. Embodied intelligence via learning and evolution. *Nat. Commun.* **2021**, *12*, 5721. [CrossRef]
41. Balduzzi, D.; Garnelo, M.; Bachrach, Y.; Czarnecki, W.; Perolat, J.; Jaderberg, M.; Graepel, T. Open-ended learning in symmetric zero-sum games. In *International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2019; pp. 434–443.

42. Perez-Nieves, N.; Yang, Y.; Slumbers, O.; Mguni, D.H.; Wen, Y.; Wang, J. Modelling behavioural diversity for learning in open-ended games. In *International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2021; pp. 8514–8524.
43. McAleer, S.; Lanier, J.B.; Fox, R.; Baldi, P. Pipeline psro: A scalable approach for finding approximate nash equilibria in large games. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 20238–20248.
44. Muller, P.; Omidshafiei, S.; Rowland, M.; Tuyls, K.; Perolat, J.; Liu, S.; Hennes, D.; Marris, L.; Lanctot, M.; Hughes, E.; et al. A Generalized Training Approach for Multiagent Learning. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
45. Hernandez, D.; Denamganai, K.; Devlin, S.; Samothrakis, S.; Walker, J.A. A comparison of self-play algorithms under a generalized framework. *IEEE Trans. Games* **2021**, *14*, 221–231. [CrossRef]
46. Tesauro, G. TD-Gammon, a self-teaching backgammon program, achieves master-level play. *Neural Comput.* **1994**, *6*, 215–219. [CrossRef]
47. Silver, D.; Schrittwieser, J.; Simonyan, K.; Antonoglou, I.; Huang, A.; Guez, A.; Hubert, T.; Baker, L.; Lai, M.; Bolton, A.; et al. Mastering the game of go without human knowledge. *Nature* **2017**, *550*, 354–359. [CrossRef]
48. Ye, D.; Chen, G.; Zhao, P.; Qiu, F.; Yuan, B.; Zhang, W.; Chen, S.; Sun, M.; Li, X.; Li, S.; et al. Supervised learning achieves human-level performance in moba games: A case study of honor of kings. *IEEE Trans. Neural Netw. Learn. Syst.* **2020**, *33*, 908–918. [CrossRef] [PubMed]
49. Baker, B.; Kanitscheider, I.; Markov, T.; Wu, Y.; Powell, G.; McGrew, B.; Mordatch, I. Emergent Tool Use From Multi-Agent Autocurricula. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
50. Shamma, J.S.; Arslan, G. Dynamic fictitious play, dynamic gradient play, and distributed convergence to Nash equilibria. *IEEE Trans. Autom. Control* **2005**, *50*, 312–327. [CrossRef]
51. Siu, H.C.; Peña, J.; Chen, E.; Zhou, Y.; Lopez, V.; Palko, K.; Chang, K.; Allen, R. Evaluation of Human-AI Teams for Learned and Rule-Based Agents in Hanabi. In *Advances in Neural Information Processing Systems*; Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., Vaughan, J.W., Eds.; Curran Associates, Inc.: Red Hook, NY, USA, 2021; Volume 34, pp. 16183–16195.
52. He, J.Z.Y.; Erickson, Z.; Brown, D.S.; Raghunathan, A.; Dragan, A. Learning Representations that Enable Generalization in Assistive Tasks. In Proceedings of the 6th Annual Conference on Robot Learning, Auckland, New Zealand, 14–18 December 2022.
53. Liu, S.; Lever, G.; Merel, J.; Tunyasuvunakool, S.; Heess, N.; Graepel, T. Emergent Coordination Through Competition. In Proceedings of the 7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, 6–9 May 2019.
54. Liu, X.; Jia, H.; Wen, Y.; Hu, Y.; Chen, Y.; Fan, C.; Hu, Z.; Yang, Y. Towards unifying behavioral and response diversity for open-ended learning in zero-sum games. *Adv. Neural Inf. Process. Syst.* **2021**, *34*, 941–952.
55. Zhou, M.; Chen, J.; Wen, Y.; Zhang, W.; Yang, Y.; Yu, Y. Efficient Policy Space Response Oracles. *arXiv* **2022**, arXiv:2202.0063v4.
56. Omidshafiei, S.; Papadimitriou, C.; Piliouras, G.; Tuyls, K.; Rowland, M.; Lespiau, J.B.; Czarnecki, W.M.; Lanctot, M.; Perolat, J.; Munos, R. α-rank: Multi-agent evaluation by evolution. *Sci. Rep.* **2019**, *9*, 9937. [CrossRef]
57. Marris, L.; Muller, P.; Lanctot, M.; Tuyls, K.; Graepel, T. Multi-agent training beyond zero-sum with correlated equilibrium meta-solvers. In *International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2021; pp. 7480–7491.
58. Muller, P.; Rowland, M.; Elie, R.; Piliouras, G.; Pérolat, J.; Laurière, M.; Marinier, R.; Pietquin, O.; Tuyls, K. Learning Equilibria in Mean-Field Games: Introducing Mean-Field PSRO. In Proceedings of the 21st International Conference on Autonomous Agents and Multiagent Systems, AAMAS 2022, Auckland, New Zealand, 9–13 May 2022; Faliszewski, P., Mascardi, V., Pelachaud, C., Taylor, M.E., Eds.; International Foundation for Autonomous Agents and Multiagent Systems (IFAAMAS), 2022; pp. 926–934. [CrossRef]
59. McKee, K.R.; Leibo, J.Z.; Beattie, C.; Everett, R. Quantifying the effects of environment and population diversity in multi-agent reinforcement learning. *Auton. Agents Multi-Agent Syst.* **2022**, *36*, 21. [CrossRef]
60. Črepinšek, M.; Liu, S.H.; Mernik, M. Exploration and exploitation in evolutionary algorithms: A survey. *ACM Comput. Surv. (CSUR)* **2013**, *45*, 1–33. [CrossRef]
61. Garnelo, M.; Czarnecki, W.M.; Liu, S.; Tirumala, D.; Oh, J.; Gidel, G.; van Hasselt, H.; Balduzzi, D. Pick Your Battles: Interaction Graphs as Population-Level Objectives for Strategic Diversity. In Proceedings of the AAMAS'21: 20th International Conference on Autonomous Agents and Multiagent Systems, Virtual Event, UK, 3–7 May 2021; Dignum, F., Lomuscio, A., Endriss, U., Nowé, A., Eds.; ACM: New York, NY, USA, 2021; pp. 1501–1503. [CrossRef]
62. Zhao, R.; Song, J.; Hu, H.; Gao, Y.; Wu, Y.; Sun, Z.; Wei, Y. Maximum Entropy Population Based Training for Zero-Shot Human-AI Coordination. *arXiv* **2021**, arXiv:2112.11701v3.
63. Lupu, A.; Cui, B.; Hu, H.; Foerster, J. Trajectory diversity for zero-shot coordination. In *International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2021; pp. 7204–7213.
64. Bai, Y.; Jin, C. Provable self-play algorithms for competitive reinforcement learning. In *International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2020; pp. 551–560.
65. Dinh, L.C.; McAleer, S.M.; Tian, Z.; Perez-Nieves, N.; Slumbers, O.; Mguni, D.H.; Wang, J.; Ammar, H.B.; Yang, Y. Online Double Oracle. *arXiv* **2021**, arXiv:2103.07780v5.
66. Yin, Q.; Yu, T.; Shen, S.; Yang, J.; Zhao, M.; Huang, K.; Liang, B.; Wang, L. Distributed Deep Reinforcement Learning: A Survey and A Multi-Player Multi-Agent Learning Toolbox. *arXiv* **2022**, arXiv:2212.00253.

67. Espeholt, L.; Marinier, R.; Stanczyk, P.; Wang, K.; Michalski, M. SEED RL: Scalable and Efficient Deep-RL with Accelerated Central Inference. In Proceedings of the International Conference on Learning Representations, Addis Ababa, Ethiopia, 26–30 April 2020.
68. Nair, A.; Srinivasan, P.; Blackwell, S.; Alcicek, C.; Fearon, R.; De Maria, A.; Panneershelvam, V.; Suleyman, M.; Beattie, C.; Petersen, S.; et al. Massively parallel methods for deep reinforcement learning. *arXiv* **2015**, arXiv:1507.04296.
69. Espeholt, L.; Soyer, H.; Munos, R.; Simonyan, K.; Mnih, V.; Ward, T.; Doron, Y.; Firoiu, V.; Harley, T.; Dunning, I.; et al. Impala: Scalable distributed deep-rl with importance weighted actor-learner architectures. In *International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2018; pp. 1407–1416.
70. Flajolet, A.; Monroc, C.B.; Beguir, K.; Pierrot, T. Fast population-based reinforcement learning on a single machine. In *International Conference on Machine Learning*; PMLR: Cambridge, MA, USA, 2022; pp. 6533–6547.
71. Shih, A.; Sawhney, A.; Kondic, J.; Ermon, S.; Sadigh, D. On the Critical Role of Conventions in Adaptive Human-AI Collaboration. In Proceedings of the 9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, 3–7 May 2021.
72. Hwangbo, J.; Lee, J.; Dosovitskiy, A.; Bellicoso, D.; Tsounis, V.; Koltun, V.; Hutter, M. Learning agile and dynamic motor skills for legged robots. *Sci. Robot.* **2019**, *4*, eaau5872. [CrossRef] [PubMed]
73. Lee, J.; Hwangbo, J.; Wellhausen, L.; Koltun, V.; Hutter, M. Learning quadrupedal locomotion over challenging terrain. *Sci. Robot.* **2020**, *5*, eabc5986. [CrossRef] [PubMed]
74. Miki, T.; Lee, J.; Hwangbo, J.; Wellhausen, L.; Koltun, V.; Hutter, M. Learning robust perceptive locomotion for quadrupedal robots in the wild. *Sci. Robot.* **2022**, *7*, eabk2822. [CrossRef] [PubMed]
75. OpenAI, O.; Plappert, M.; Sampedro, R.; Xu, T.; Akkaya, I.; Kosaraju, V.; Welinder, P.; D'Sa, R.; Petron, A.; Pinto, H.P.d.O.; et al. Asymmetric self-play for automatic goal discovery in robotic manipulation. *arXiv* **2021**, arXiv:2101.04882.
76. Riviere, B.; Hönig, W.; Anderson, M.; Chung, S.J. Neural tree expansion for multi-robot planning in non-cooperative environments. *IEEE Robot. Autom. Lett.* **2021**, *6*, 6868–6875. [CrossRef]
77. Mahjourian, R.; Miikkulainen, R.; Lazic, N.; Levine, S.; Jaitly, N. Hierarchical policy design for sample-efficient learning of robot table tennis through self-play. *arXiv* **2018**, arXiv:1811.12927.
78. Li, D.; Li, W.; Varakantham, P. Diversity Induced Environment Design via Self-Play. *arXiv* **2023**, arXiv:2302.02119.
79. Posth, J.A.; Kotlarz, P.; Misheva, B.H.; Osterrieder, J.; Schwendner, P. The applicability of self-play algorithms to trading and forecasting financial markets. *Front. Artif. Intell.* **2021**, *4*, 668465. [CrossRef]