electronics

MDPI

Article

NFSP-PLT: Solving Games with a Weighted NFSP-PER-Based Method

Huale Li [1,2,3,4], Shuhan Qi [3,4,*], Jiajia Zhang [3], Dandan Zhang [3], Lin Yao [3], Xuan Wang [3], Qi Li [5] and Jing Xiao [6]

[1] School of Software, Northwestern Polytechnical University, Xi'an 710072, China
[2] Yangtze River Delta Research Institute of NPU, Taicang 215400, China
[3] School of Computer Science and Technology, Harbin Institute of Technology (Shenzhen), Shenzhen 518055, China
[4] Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies, Shenzhen 518000, China
[5] China Merchants Group Digital Transformation Center, Shenzhen 518000, China
[6] Ping An Insurance (Group) Company, Shenzhen 518000, China
* Correspondence: shuhanqi@cs.hitsz.edu.cn

Abstract: Nash equilibrium strategy is a typical goal when solving two-player imperfect-information games (IIGs). Neural fictitious self-play (NFSP) is a popular method to find the Nash equilibrium in IIGs, which is the first end-to-end method used to compute the Nash equilibrium strategy. However, the training of NFSP requires a large number of sample data and the interactive cost of obtaining such data is often very high. Realizing the efficient training of network under limited samples is an urgent problem. In this paper, we first proposed a new NFSP-based method, NFSP with prioritized experience replay (NFSP-PER), to improve the sample training efficiency. Then, a weighted NFSP-PER with learning time (NFSP-PLT) was proposed to control the utilization degree of priority-weighted samples. Furthermore, based on the NFSP-PLT, an adaptive upper-confidence-bound applied to tree (UCT) is used to solve the optimal response strategy, which makes the solving strategy more accurate. Extensive experimental results show that the proposed NFSP-PLT effectively improves the sample learning efficiency compared with the existing works.

Keywords: game theory; imperfect information; neural fictitious self-play; deep reinforcement learning

Citation: Li, H.; Qi, S.; Zhang, J.; Zhang, D.; Yao, L.; Wang, X.; Li, Q.; Xiao, J. NFSP-PLT: Solving Games with a Weighted NFSP-PER-Based Method. *Electronics* **2023**, *12*, 2396. https://doi.org/10.3390/electronics12112396

Academic Editor: Yoichi Hayashi

Received: 17 April 2023
Revised: 12 May 2023
Accepted: 22 May 2023
Published: 25 May 2023

1. Introduction

With the rapid development of artificial intelligence, computer games, an artificial intelligence research direction, have attracted much attention in recent years. Compared with the perfect-information game with completely observable information, the imperfect-information game with unobservable information is closer to the real application scenario, and its solution is also more challenging. Due to the existence of this hidden information, it is more difficult to make decisions than it is in perfect-information games. In recent years, the study on two-player IIGs has achieved great success. Among them, Nash equilibrium, an important concept in the game theory, has attracted the attention of researchers. Nash equilibrium [1] is an optimal solution to the game, which is often used as the solution goal of imperfect-information games. At present, the main methods to compute the Nash equilibrium strategy are counterfactual regret minimization (CFR) [2] and fictitious play (FP) [3].

The CFR is an iterative method to solve the game strategy, and the final average strategy solved by CFR is a Nash equilibrium strategy or approximate Nash equilibrium strategy. The CFR expands a game problem through the game tree, and iteratively traverses each node of this game tree. The CFR has achieved great success in recent years, especially in poker games [4–8]. For example, DeepStack [4] is the first computer program in the world to defeat professional poker players in heads-up no-limit Texas Hold'em, using neural

networks combined with CFR to construct a deep counterfactual value network to fit regret value estimation. Libratus [5] also defeated top human specialist professionals in heads-up no-limit Texas hold'em, using a blueprint for the overall strategy and a self-improver algorithm for fixing potential weaknesses. Brown et al. introduce deep CFR, a form of CFR that obviates the need for abstraction by instead using deep neural networks to approximate the behavior of CFR in the full game [6]. Schmid et al. introduce a variance-reduction technique that applies to any sampling variant of MCCFR (Monte Carlo counterfactual regret minimization), which brings an order-of-magnitude speedup and the empirical variance decreases by three orders of magnitude [7]. Liu et al. introduce a new CFR variant recursive CFR (ReCFR) in which recursive substitute values are learned and used to replace cumulative regrets [9]. However, limited by computing resources and storage resources, the solution scale of CFR is not enough for large-scale games. When solving large-scale game problems, the large-scale game problems must be abstracted and expert knowledge is needed for the detailed design, which greatly limits its further application [8,10–12].

The FP [3] is also an iterative method that can be used to find the Nash equilibrium or approximate Nash equilibrium in two-player IIGs. A game-player in the FP develops their best response strategy by competing against opponents in the timesteps of simulated games. After a sufficient number of repetitions is obtained, the average strategy of the historical best responses will converge to the Nash equilibrium. However, the FP only provides a framework with a theoretical guarantee, and the practical solution to the problem still needs further research. To this end, fictitious self-play (FSP) is proposed by Heinrich et al. [13], which extends the FP's scope of application to extensive-form game problems. The FSP solves the game strategy through machine learning-based methods, which greatly improves the practicality of the original FP. Furthermore, Heinrich et al. improve the FSP with a deep neural network and propose a new variant NFSP [14]. The NFSP can be applied to deep reinforcement learning [15,16] (such as deep Q-learning, DQN [17,18]) to learn the best response, and uses the supervised learning method (such as deep neural network [19]) to fit the average strategy. In the NFSP, players play the game with a mixed strategy composed of their own best response and average strategy, while opponents play the game using their average strategy. It is worth noting that, compared with the previous methods, such as CFR-based methods, the NFSP is an end-to-end method and does not require any prior knowledge. As a result, NFSP is worthy of more attention and research in the field of IIGs.

There are numerous enhanced NFSP-based techniques available at present [20–27]. For example, Xue et al. [20] proposed a novel learning paradigm to solve large-scale extensive-form NSGs (Network Security Games). The NSG-NFSP reform the best response policy network in NFSP to be a mapping from action–state pair to action–value, to make the calculation of best response possible in NSGs. Chen et al. [21] proposed an NFSP-based method, replacing the computation of best response in NFSP with regret-matching. Zhang et al. [22] proposed Monte Carlo-NFSP (MC-NFSP), which combines a Monte Carlo tree search with NFSP to improve the performance in real-time zero-sum IIGs. Ganzfried [23] compared the performance of two popular algorithms, FP and CFR, and showed that FP leads to improved Nash equilibrium approximation over a variety of game classes and sizes. Kamra et al. [24] developed an approximate extension of FP to two-player games with high-dimensional continuous action spaces. Perolat et al. [25] built a stochastic approximation of the FP process by using an architecture inspired by actor-critic algorithms. He et al. [26] proposed local-regret-minimization-based FP (LRM-FP) for computing approximate Nash equilibrium. The LRM-FP obtains a behavioral strategy by computing the counterfactual regret from the local state-action value. Han et al. [27] presented a new NFSP-based method combining NFSP and Kernel Regression UCT (KR-UCT) [28] for digital curling games (DCGs), where NFSP uses two adversary learning networks and can automatically produce supervised data. Table 1 provides a brief description of CFR- and NFSP-based methods.

Table 1. A brief description of CFR- and NFSP-based methods.

Method	Game Types	Contribution
DeepStack [4]	two-player IIGs	combines deep neural network valuation to conduct a finite depth search on the game tree, the first to defeat professional poker players in heads-up no-limit Texas Hold'em
Libratus [5]	two-player IIGs	defeated top human specialist professionals in heads-up no-limit Texas hold'em, using a blueprint for the overall strategy and a self-improver algorithm for fixing potential weaknesses
deep CFR [6]	two-player IIGs	obviates the need for abstraction by instead using deep neural networks to approximate the behavior of CFR in the full game
MCCFR [7]	two-player IIGs	introduces a variance-reduction technique, brings an order of magnitude speedup and the empirical variance decreases
FSP [13]	two-player IIGs	solves the game strategy through machine learning-based methods, which greatly improves the practicality of the FP
NFSP [14]	two-player IIGs	a first end-to-end solving method for FSP, learns the best response with DRL and uses the supervised learning to fit the average strategy
[25]	two-player IIGs	uses an architecture inspired by actor-critic algorithms, builds a stochastic approximation of the FP process
[24]	two-player IIGs	uses generative neural networks to approximate players' best responses while also learning a differentiable approximate model to the players' rewards
NSG-NFSP [20]	NSGs	reforms the best response policy network in NFSP to be a mapping from action-state pair to action-value
[21]	two-player IIGs	replaces the computation of best response in NFSP with regret -matching, make the optimality gap converge to zero as it iterates
MC-NFSP [22]	two-player IIGs	uses MC to NFSP, improves the performance in real-time zero-sum IIGs
LRM-FP [26]	two-player IIGs	obtains a behavioral strategy by computing counterfactual regret from the local state-action value
[27]	DCGs	combines NFSP and KR-UCT, uses two adversary learning networks and can automatically produce supervised data

The NFSP still requires a large number of training samples to make the network performance good enough, although these methods improve the performance of NFSP to some extent. In previous NFSP-based methods, the training samples are randomly sampled. However, the value of different samples is different, so it is efficient to select more valuable samples for training. In terms of sample utilization efficiency, the NFSP still has great room for improvement. Therefore, this paper studies the problem of sample utilization efficiency in the training of NFSP. In vanila DQN [18], network training is carried out through uniform sampling. Prioritized experience replay [29] increases sample utilization efficiency by changing the priority of sample sampling, which further improves the performance of vanilla DQN. In view of this, in this paper, we propose an NFSP-based method, NFSP with Prioritized Experience Replay (NFSP-PER), which enhances the sample learning efficiency by optimizing the empirical learning method. Specifically, the priority experience replay mechanism is introduced to preferentially sample samples with higher value. Through the more efficient use of samples with higher value, the priority-weighted sample utilization degree control is realized, and the efficient sample learning of NFSP is realized. In the NFSP-PER, considering the instability of the traditional deep reinforcement learning (DRL) method [16] when solving the optimal response strategy, an adaptive UCT is used, combined with the traditional DRL method. Extended experiments show that the proposed method improves the sample learning efficiency compared with the comparison methods. We summarize our contributions as follows:

(1) We propose an NFSP-based method, studying the strategy of IIGs. Not only can the proposed method solve the strategy of two-player IIGs, but can also solve the strategy of multiplayer IIGs.

(2) We propose an NFSP-PER combining NFSP with Prioritized Experience Replay, which enhances the sample learning efficiency by optimizing the empirical learning method. In addition, NFSP-PLT is proposed to realize the priority-weighted sample utilization degree control. Furthermore, in the NFSP-PER, an adaptive UCT is used, combined with the traditional DRL method, to solve the optimal response strategy.

(3) Extensive experimental results showed that the proposed NFSP-PLT efficiently improves the sample learning efficiency and can solve the strategy of multiplayer IIGs compared with the comparison methods.

This paper is organized as follows: Section 2 introduces related works. Section 3 introduces the system model and problem formulation. Section 4 introduces an overview of the proposed method in detail. Section 5 shows the experimental results, including comparison experiments and ablation studies. Section 6 presents the conclusion of this paper.

2. Related Works

In this section, the concepts of Nash equilibrium, NFSP and UCT will be introduced. The Table 2 provides a detailed description of symbols in the paper.

Table 2. Detailed descriptions of symbols in the paper.

Symbol	Detailed Description
σ_i	The strategy of game-player i, a probability vector over actions for the player i on the current state of the game. σ_{-i} is the strategy of game-players except player i.
u_i	The payoff of game-player i. $u_i(\sigma_i, \sigma_{-i})$ is the payoff of game-player i when the strategies σ_i and σ_{-i} are applied by players i and $-i$, respectively.
$BR(\sigma_{-i})$	The game-player i's best response to the opponent $-i$'s strategy. This means the strategy obtaining the most payoff for player i when the opponent $-i$ adopts the strategy σ_{-i}.
σ_i^*	The Nash equilibrium strategy, which means player i chose the best response for the opponent $-i$.
n_j	The number of visits to child node j. n represents the number of visits to the current node. $\bar{X}_j$ represents the mean value of selecting the child node.
$\lvert\delta_i\rvert$	The temporal difference error [16] (TD-error) of the experience sample e_i.
α	This controls the effect degree of the TD-error, $\alpha \in [0, 1]$.

2.1. Nash Equilibrium

Nash equilibrium is a classical concept in the field of game theory [1]. Nash equilibrium is a strategy profile in which a game player cannot obtain more benefits by deviating from Nash equilibrium. Here, we provide some necessary descriptions to better understand the Nash equilibrium.

In the field of IIGs, a strategy refers to the probability distribution of all legal actions on an information set. The information set is a unique concept in IIGs. The game state in the information set is indistinguishable for game-players. For a specific game problem, the best response of the game-player is the best response to the opponent's strategy. When the game-player chooses the best response in the game process, the game-player can obtain the maximum benefit in this game. The formal definition of the best response $BR(\sigma_{-i})$ is [1]:

$$u_i(BR(\sigma_{-i}), \sigma_{-i}) = \max_{\sigma_i'} u_i(\sigma_i', \sigma_{-i}) \tag{1}$$

where $BR(\sigma_{-i})$ is the game-player's best response to opponents' strategy σ_{-i}; σ_i is the strategy of game-player i; σ_{-i} is the strategy of game-players except player i; u_i is the payoff of game-player i. For a two-player zero-sum game, $u_1 + u_2 = 0$.

Nash equilibrium strategy refers to when game-players choose the best response strategy. A Nash equilibrium σ^* can be defined as [1]:

$$u_i(\sigma_i^*, \sigma_{-i}^*) = \max_{\sigma_i'} u_i(\sigma_i', \sigma_{-i}^*) \tag{2}$$

It should be noted that, in most cases, it is very difficult to directly calculate the Nash equilibrium. Thus, in many cases, we can calculate an approximate Nash equilibrium: ϵ-Nash equilibrium. In an ϵ-Nash equilibrium, no game-player can improve their utility more than ϵ by unilaterally changing their acting strategy. An ϵ-Nash equilibrium can be defined as [1]:

$$u_i(\sigma_i^*, \sigma_{-i}^*) + \epsilon = \max_{\sigma_i'} u_i(\sigma_i', \sigma_{-i}^*) \tag{3}$$

2.2. Neural Fictitious Self-Play

NFSP [14] is a method of using a machine learning algorithm to solve the game strategy under the framework of game theory. The average strategy could theoretically be proved to be the Nash equilibrium strategy or approximate Nash equilibrium strategy.

Specifically, there are best responses and average strategies in NFSP [14], in which the best response is fitted by the DRL [16] method and the average strategy is fitted by a deep neural network. In the game, the game-player chooses the mixed strategy, composed of the best response and the average strategy, to play the game, while the opponent adopts the average strategy for the game-player. In the process of network training, NFSP [14] contains two datasets for sample storage. One is the best-response sample dataset, which is used to store the samples generated when the game-player makes decisions according to the mixed strategy in the game process; the other is the average strategy sample dataset, which is used to store the samples generated when the game-player makes decisions according to the best response in the game process.

In addition, in order to assure the stability of the resulting algorithm and enable simultaneous self-play learning, the NFSP employs two technical advances. To eliminate the windowing artefacts caused by sampling from a finite data memory, it firstly employs reservoir sampling [30]. Secondly, it employs anticipatory dynamics [31] to allow each agent to sample its own best reaction behaviour while also tracking changes in opponents' behaviour more effectively.

2.3. Upper Confidence Bound Applied to Tree

UCT is the most widely used MCTS algorithm in recent years [32]. The UCT algorithm can be used to solve the problem of "exploration and utilization" in MCTS. Exploration refers to the selection of unreached nodes, and utilization refers to the utilization of known historical information in the iteration process. UCT algorithm adopts upper confidence bounds (UCB) in the sub-node selection stage of the Monte Carlo tree search. Kocsis and Szepesvári [33] proposed UCB to realize the tree search. In the selection of child nodes in the dobby gambling machine problem, the value of child nodes is approximated by Monte Carlo simulation to approximate the expected reward. UCB has the advantages of simplicity and efficiency, and can ensure that the optimal limit is kept in a constant range in the case of regret growth [34]. Whenever a node (or an action) is to be selected in the current game tree, the selection can be modeled as an independent multi-armed bandit machine problem, with the parent node as a bandit and the action of selecting a child node as a rocker. The child node j selected in each iteration needs to meet the following equation to be maximized:

$$UCT = \bar{X}_j + 2C_p\sqrt{\frac{2\ln n}{n_j}} \tag{4}$$

where n represents the number of visits to the current node, n_j the number of visits to child node j. $C_p > 0$ and C_p is a constant. $\bar{X}_j$ represents the mean value of selecting the child node.

The first term of Equation (4) is equivalent to the confidence interval of utilization, and the second term is equivalent to the degree of trust in exploration. The essence of Equation (4) is to achieve a balance between exploration and utilization in MCTS. The exploration term can be understood as being more willing to explore better strategies,

while the utilization term indicates that it is more inclined to use the existing optimal strategies. The global optimal strategy solution can be solved through a balance between exploration and utilization. On the one hand, with the access of each node, the denominator of the exploration item increases, reducing its proportion. On the other hand, if another child node of the parent node is accessed, the numerator increases. Thus, the value of the exploration item of the brother node that is not accessed increases. The exploration item can ensure that the selection probability of each child node is non-zero, which is very important for the randomness of the strategy. Therefore, even the child nodes with low rewards are guaranteed to be finally selected (in the case of sufficient time), and different game tree paths can be explored. This can be determined by adjusting the constant C_p of the exploration item that determines the importance of strategic exploration. As the UCT iterations grow to infinity, the probability of choosing a suboptimal action at the root node converges to zero at a polynomial rate [35–38].

3. System Model and Problem Formulation

Extended game [39] is a classic game model that represents sequential decision games. Especially in the field of IIGs, extended games are often used to model corresponding problems. Extended game describes the game process through a game tree, where nodes (also known as decision points) represent the game state. A game process is a path in which the starting node is the root node and the ending node is the leaf node. The direction of this path is determined by the game actions chosen by game players. In the field of incomplete information games, a finite extended game usually uses a six-tuple $< N, H, P, f, I, u >$ for formal description [39]. N is the set of game players in the game. H is the set of game states, and the action sequence constitutes each game state h, $h \in H$. P is the player function and $P(h)$ is the player who will take an action after the state h. f is a function that associates with every state h. I is the information set of the game. u is the payoff function for game players.

We conducted research on strategy solving problems in the field of IIGs. From the previous introduction, it can be found that the main goal is solving its Nash equilibrium strategies or approximate Nash equilibrium strategies. In the paper, we first model the poker problem as an extended game model for representation. The vanilla NFSP [14] can solve game strategies with an end-to-end way and avoid the need for a large amount of prior knowledge. NFSP uses DRL to solve the optimal strategy and supervised learning to solve the average strategy. Finally, it can theoretically ensure that its average strategy is an approximate Nash equilibrium strategy. However, NFSP has a low efficiency in sample utilization, which hinders its further expansion and application. We conducted research on this issue in the paper. The goal is to improve the efficiency of training samples while maintaining game performance.

4. Our Method

In this section, the proposed methods NFSP-PER and NFSP-PLT are introduced and elaborated in detail.

4.1. An Overview of NFSP-PER and NFSP-PLT

To improve the sample efficiency in the training process of NFSP, we applied a priority experience replay mechanism [29] to the NFSP [14] and proposed an NFSP-PER. In our NFSP-PER, samples with higher value will be given a higher sampling priority during network training. In addition, for samples with a higher value, our method will further improve its utilization to better mine the sample value. Finally, the efficient network training is realized under a limited number of samples.

The framework of NFSP-PER is shown in Figure 1. There are two game-players, player and opponent, in the figure. The player acts according to the mixed strategy while the opponent acts according to the average strategy. M_{DRL} is used to train the DRL method and M_{SL} is used to train the supervised learning method. S_t is the game state at the t-th

timestep. a_t is the action in the t-th timestep. r_{t+1} is the reward when the player takes the action a_t at the game state S_t.

Step 1: The game-players both make decisions in the game, in which the player acts with a mixed strategy and the opponent acts with an average strategy. Here, the mixed strategy consists of the best response and the average strategy.

Step 2: Storing samples. There are two sample datasets, M_{DRL} $(s_t, a_t, r_{t+1}, s_{t+1})$ and M_{SL} (s_t, a_t). M_{DRL} stores samples when the player acts in the game, while M_{SL} stores samples when the player acts with best response.

Step 3: Training networks. This step is mainly used to train the neural networks with a priority experience replay mechanism [29]. In our NFSP-PER, we take the DQN (deep Q-learning) [18] as the DRL method and deep neural network as the supervised method.

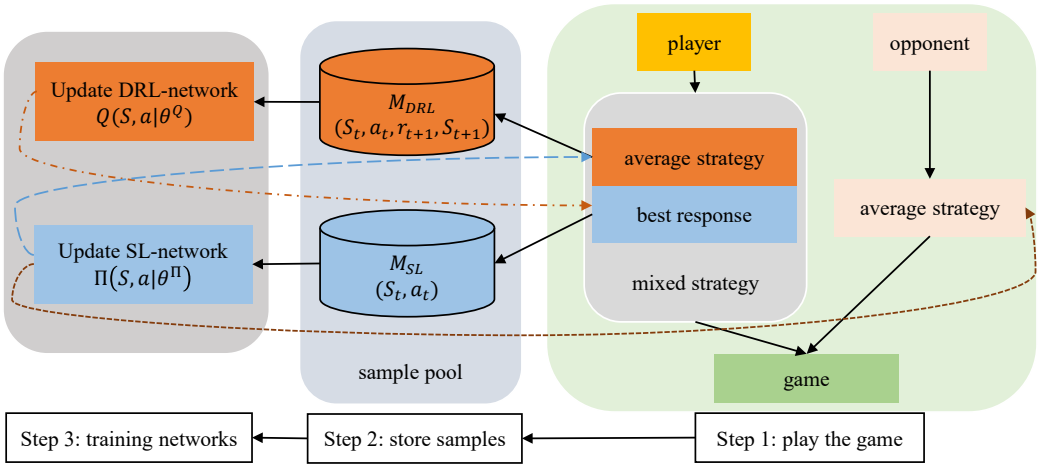


Figure 1. The framework of the proposed NFSP-PER.

After the optimization of the learning sequence of experience segments is completed in the NFSP-PER, the optimization of learning degree of experience segments is further considered [40]. Based on this, NFSP-PLT is proposed in the paper. The core idea of the NFSP-PLT is that the experience fragments with a higher learning value should be given more learning times. This paper adjusts the learning times of experience segments in training to control the learning degree of experience segments with different values. The framework of the NFSP-PLT is the same as the NFSP-PER, as shown in Figure 1.

4.2. Non-Fictitious Self Play-Priority Experience Replay

Compared with the vanilla NFSP [14], our NFSP-PER mainly improves the efficiency of training samples. Then, we mainly focus on step 3 in Figure 1; that is, describing the use of samples for network training in detail.

In the NFSP, DQN [18] adopts the experience replay mechanism, and the learning order of experience samples is adjusted to a certain extent. It constantly updates to obtain the optimal Q value, $Q^*(s,a) \leftarrow Q(s,a) + \alpha(r + \gamma max Q(s',a') - Q(s,a))$. $Q(s,a)$ is the Q value of state-action value when taking the action a on the state s. α is the learning rate, γ is the discount factor and r is the reward when taking the action a on the state s. DQN saves all the previous experience samples and then simply samples them randomly [18]. It learns the experience samples of the later experience first, or learns the experience samples of the previous experience later. However, there is a lack of refined management for the learning sequence of experience samples with different values in the sample pool when using simple random sampling. As long as random sampling is used, no matter whether the learning value of the experience sample is high or low and whether the experience sample contains any learning value, priority will be given to learning. In view of this, we can provide a better organizational arrangement to the learning order.

Based on the experience replay mechanism in the DQN [18], our NFSP-PER introduces the priority experience sampling mechanism and sets the priority according to the learning

value of the experience samples to filter the experience in the sample pool. For the current agent, the experience samples with higher priority will be sampled and learned first to optimize the learning order. Taking the size of TD-error [16] as a measure of priority, the TD-error can reflect how unexpected the experience sample is to the current agent; that is, it reflects the extent to which the newly generated experience exceeds the knowledge and experience of the intelligence. Therefore, the larger the TD-error, the more valuable the experience sample, and the higher the priority should be. The priority of samples $p(e_i)$ can be defined as:

$$p(e_i) = \frac{(|\delta_i| + \varepsilon)^{\alpha}}{\sum_k (|\delta_i| + \varepsilon)^{\alpha}} \tag{5}$$

where e_i is the experience sample; $|\delta_i|$ is the TD-error of e_i and $\alpha \in [0, 1]$, which controls the effect degree of the TD-error. When $\alpha = 0$, it degenerates to simple random sampling. ε is a positive value with small value, which is used to avoid zero-priority experience samples and ensure that all experience samples can be sampled. This representation balances simple random sampling with greedy priority sampling (i.e., sampling in descending order of TD-error [16] absolute value). Sampling according to the normalized probability ensures that the sampling probability is proportional to the priority and contains monotonic characteristics. It also improves the exploration of sampling and the diversity of experience samples.

It should be noted that the introduction of priority changes the probability distribution of the original experience samples and introduces deviation, which needs to be corrected. The method adopts weighted importance sampling for deviation annealing, which has a small variance compared with general importance sampling. When correcting, the importance sampling weight is added to the loss function as the weight coefficient, and then the updated gradient is used to complete the update.

In the NFSP-PER, with the gradual convergence of training, the weight can be elastically adjusted by the annealing method, and the annealing coefficient β can be introduced. Starting from an initial value β_0, the influence of weight will gradually decrease as the training gradually linearly increases to 1 to determine the final weight. Thus, the final weight can be defined as:

$$\begin{aligned} \omega_i &= \frac{(N \cdot p(e_i))^{-\beta}}{\max_j \omega_j} \\ &= \frac{(N \cdot p(e_i))^{-\beta}}{\max_j (N \cdot p(e_j))^{-\beta}} \\ &= \left(\frac{p(e_i)}{\min_j p(e_j)} \right)^{-\beta} \end{aligned} \tag{6}$$

where $\omega_i = \frac{1}{N} \frac{1}{p(e_i)}$.

4.3. NFSP-PLT

After the optimization of the learning sequence of experience samples is completed, the optimization of the learning degree of experience samples is further considered. The higher the learning value, the more in-depth learning should be provided. This paper adjusts the learning times of experience samples in training to control the learning degree of experience samples with different values. Based on the NFSP-PER, we further proposed NFSP-PLT. Specifically, the priority of the previous part is used as the weight coefficient of the learning times. The learning times $LT(e_i)$ of experience segments in one instance of training can be defined:

$$LT(e_i) = clip[p(e_i) N_{LTmax}, N_{LTmin}, N_{LTmax}] \tag{7}$$

where N_{LTmin} and N_{LTmax} are the upper and lower bounds of the learning times LT of experience segments, respectively. $clip$ limits the learning times LT after rounding to the range $[N_{LTmin}, N_{LTmax}]$. In this process, attention should be paid to ensuring a moderate degree of learning and avoid over-fitting problems.

In addition, the best response of NFSP-PER uses batch learning, and the priorities of multiple experience segments obtained by batch sampling may be different, so the priority weight is updated to the average of the priorities of all experience segments. The learning times LT of experience segments in a batch training can be redefined as follows:

$$LT(e) = \text{clip}\left[\frac{1}{k}\sum_{i=0}^{k-1} p(e_i) N_{lt\,\max}, N_{lt\,\min}, N_{lt\,\max}\right] \tag{8}$$

where k is the batch size and $e = e_0, e_1, L, e_k$ is the sampled batch experience.

A detailed algorithm of the proposed NFSP-LT is described in Algorithm 1. In the Algorithm 1, η is the dynamic anticipation parameter, $\eta \in [0,1]$, β is the best-response strategy ($\beta = \epsilon - greedy(Q)$, which selects a random action with probability ϵ and otherwise chooses the action that maximizes the predicted action values, Q represents the action-value network of DQN), π is the average strategy, M_{DRL} ($s_t, a_t, r_{t+1}, s_{t+1}$) and M_{SL} (s_t, a_t) are sample datasets, M_{DRL} stores samples when the player acts in the game, while M_{SL} stores samples when the player acts with the best response and T is the upper bound of episodes.

Algorithm 1 The algorithm of the proposed NFSP-PLT

Require: Initialize the game environment Γ, game player, fictitious player
Output: The strategy π^*

```
Initialize σ = (1 − η)β + ηπ, θ, θ′, M_DRL, M_SL, T
for episode 1 → T do
    while the episode is not the end do          ▷ Sample and collect the experience data
        select the action according to the σ_i playing with the average strategy π_{−i} of the fictitious player
        generate the experience fragments e
        determine the priority of e and save to the M_SL
        if σ = β then                            ▷ Save the data with best response
            save e to the M_DRL
        end if
    end while
    priority sampling experience fragments in the M_SL and update the DRL method with θ
    periodically update θ′
    update the priority of the M_SL with the learned TD-error
    update the π with M_DRL
end for
π = π* return π*
```

4.4. Applying Adaptive UCT in the NFSP-PER

We used the experience playback mechanism to improve the sample learning efficiency of traditional NFSP in NFSP-PER and NFSP-PLT. Similar to the traditional NFSP method, the DRL method is also used in the previous proposed method to solve the optimal response strategy, which will lead to a difficult convergence and unstable convergence in the training process. To solve this problem, we further propose a strategy solution method based on a combination of adaptive UCT [32] and DRL to solve the optimal response strategy. It should be noted that the difference between this method and the previously proposed method is the introduction of adaptive UCT when solving the optimal strategy. Therefore, unless otherwise specified in the following statements, the sampling forms used are experience playback mechanisms. Considering that an adaptive UCT algorithm based

on fittest survival MCTS (FS-MCTS) [41] is used to solve the optimal response strategy, we first introduce the adaptive UCT algorithm.

The calculation process of the traditional UCT algorithm has to deal with millions of game tree nodes in each iteration [32]. This will simulate until a fixed number of iteration rounds and traverse all game states. Adaptive UCT combines the simulated adaptive mechanism with the iteration of UCT. First, we will clarify several key concepts, including the candidate set, current optimal candidate, and adaptive threshold. Among them, the candidate set represents the optional actions under the current information set. The current optimal action represents the optimal action in the candidate set. The optimal judgment standard is determined according to the selection node strategy of UCT [32]. The adaptive threshold represents the number of iteration rounds to start the significance detection. The core idea of the adaptive mechanism is that, under the current number of iteration rounds, the optimal action of the current information set can be significantly distinguished by historical statistical data, and then the iteration process of UCT can be terminated in advance. The detailed algorithm of using adaptive UCT to solve optimal response strategy is described in the Algorithm 2. Then, the adaptive UCT process is described as follows:

Algorithm 2 The algorithm of solving optimal response strategy based on adaptive UCT

Require: The current game state S_0, iteration time T, adaptive threshold $T\_Threshold$, sample significance level α
Output: The optimal action A_best in current state

for *iteration* $1 \rightarrow T$ **do**
  create the node of game tree: $Node\_current \leftarrow S_0$
  while $Node\_current$ is non-terminal **do** ▷ Expand the game tree
    $U(s,a) \propto P(s,a)/(1+N(s,a))$
    Select($Node\_current$): $Node\_current = \text{argmax}_{node}(Q(s,a)+U(s,a))$
  end while
  if $Node\_current$ is not fully expanded **then**
    $Node\_current \leftarrow Expand(Node\_current)$
  end if
  simulate with value and policy network: $(P(s',\cdot),V(s')) = f_\theta(s') \leftarrow Simulation(Node\_current)$
  backup from current node: $Q(s,a) = 1/N(s,a)\sum_{s'|s,a\rightarrow s'} V(s') \leftarrow Backup((Node\_current)$
  if $iteration \leqslant T\_Threshold$ **then**
    if $Z(A_{current}, A_{others}) \leqslant Z_{\alpha/2}$ **then**
      return $A_{current}$
    end if
  end if
end for
return $argmax_A(N(S))$

Step 1: Set the initial candidate set as all the optional actions of the current information set. Use the UCT algorithm to iteratively simulate the current information set. Use the memory module to count the historical simulation data until the number of iterations reaches the adaptive threshold.

Step 2: Obtain the current best candidate according to the UCT, and start to perform significance detection. Compare the candidate set elements with the best candidate. If the significance of the best action is better than a candidate and the candidate will be deleted from the candidate set and if there is only one element in the candidate set, execute step 4; otherwise, execute step 3.

Step 3: Continue to perform UCT simulation, iterating for a fixed number of rounds, and then return to step 2.

Step 4: Terminate the iteration in advance and take the current candidate as the optimal action of the information set to return the result.

The strategy solution of traditional UCT algorithm [32] usually depends on the number of iteration rounds, and the main improvement in the adaptive mechanism is that it can provide a pruning and early termination idea for UCT. In the adaptive UCT iteration process, when there is only one candidate action in the candidate set, the iteration can be terminated in advance; that is, when one action is significantly better than other candidates, the UCT result can be returned. This adaptive mechanism can be applied to the whole UCT iteration game tree, and can be directly applied to the subtree to prune UCT in the game tree simulation process.

5. Experiments

We conducted experiments to evaluate the performance of our proposed in terms of the reward compared with the performances of the comparison methods. In addition, we conducted ablation studies to verify the effectiveness of the proposed method.

5.1. Experimental Setup

In the field of IIGs, the poker game was regarded as a test platform for many years. Successful poker agents, such as DeepStack [4] and Libratus [5], both take the poker game Heads-up No-limit hold'em (HUNL) as the test platform to verify their effectiveness. The poker game contains all necessary elements of IIGs, such as game players, utility function and imperfect information. Thus, we also used the poker game as the test platform in this paper.

Leduc poker and the HUNL are used to test the effectiveness of the proposed methods. Leduc poker contains six cards, with two rounds in a game. The two game-players both have a private card in the first round. Here, the private card is unobservable to each other player. In the second round, a public card is dealt. In the process of the game, the action could be call, fold and raise. For the HUNL, there are 52 cards and four rounds. In the first round, the two private cards are dealt to the game-player. Three public cards are dealt in the second round. In the third round and fourth round, one public card is dealt, respectively.

In addition, game performance is used to verify NFSP-PER compared with other methods. This metric is a classical metric in the field of IIGs. In our NFSP-PER, DQN is used to fit the best response. The sample datasets of M_{DRL} and M_{SL} are 30,000 and 1,000,000, respectively. In the game, reservoir sampling is used to store $< S, a, r >$ of the best response. The learning rate in the DRL and SL are 0.1 and 0.005, respectively. The Programming language is Python and all experiments are conducted on four Xeon(R) CPUs of E5-2640 with 10 cores @2.40 GHz, and one Tesla P100 GPU with 16G memory. The discount parameter γ is 0.99. The loss function used in DQN is described as:

$$\mathcal{L}\left(\theta^{Q}\right) = \mathbb{E}_{(s,a,r,s')\sim\mathcal{M}_{DRL}}\left[\left(r + \max_{a'} Q\left(s', a' \mid \theta^{Q'}\right) - Q\left(s, a \mid \theta^{Q}\right)\right)^{2}\right], \tag{9}$$

and the loss function used in the SL method is described as:

$$\mathcal{L}\left(\theta^{\Pi}\right) = \mathbb{E}_{(s,a)\sim\mathcal{M}_{SL}}\left[-\log\Pi\left(s, a \mid \theta^{\Pi}\right)\right], \tag{10}$$

the mixed strategy in the game is $\sigma = \Pi(s, a|\theta^{\Pi})$, with the probability $1 - \eta$, and $\sigma = \epsilon - greedy$ with the probability η.

5.2. Comparison Experiment Results

In the paper, two improvements are conducted based on the NFSP, which are NFSP-PER and NFSP-PLT. In order to verify the effectiveness of these two proposed methods, three comparison methods are used to conduct experiments, which are DeepCFR [6], random and NFSP-UCT. The agent NFSP-UCT is provided by the Annual Computer Poker Competition (ACPC). A total of 20,000 games are conducted when played using the

comparison method, respectively. An average return with 95% confidence is used as the metric. Average return is measured by bb/h, which is a standard measure of the win rate in poker games [4,5]. bb/h represents how many big blinds won in each hand (in poker games, one hand stands for playing one game).

Two groups of experiments were conducted in the paper. Firstly, the comparison experiment was conducted on the Leduc and HUNL; the results are shown in Tables 3 and 4. Secondly, to further verify the performance of our method, we conducted a test experiment in multiplayer poker, which are three-player and six-player No-limit hold'em games, represented with 3-NH and 6-NH, respectively. In Table 3, 'NFSP-PER vs. Random' represents that, in a two-player game, the agents training with the two methods (NFSP-PER and Random) are used for gameplay. When the return is greater than zero, the NFSP-PER method is superior. The higher the return, the more obvious the advantage of the NFSP-PER. The other two (NFSP-PER vs. NFSP-UCT and NFSP-PER vs. DeepCFR) are similar to 'NFSP-PER vs. Random'.

Table 3. Comparison results of NFSP-PER.

Game	NFSP-PER vs. Random	NFSP-PER vs. NFSP-UCT	NFSP-PER vs. DeepCFR
Leduc	0.9731 ± 0.05	0.0168 ± 0.09	0.2162 ± 0.07
HUNL	34.0647 ± 0.48	36.7131 ± 0.66	27.9865 ± 0.63

Table 3 shows the comparison experiment results of NFSP-PER vs. comparison methods. The game performance of NFSP-PER is significantly better than the comparison methods: Random, NFSP-UCT and DeepCFR. The average return won in 20,000 games on the Leduc are 0.9731 ± 0.05 bb/h, 0.0168 ± 0.09 bb/h and 0.2162 ± 0.07 bb/h, respectively. The average return won in 20,000 games on the HUNL was 34.0647 ± 0.48 bb/h, 36.7131 ± 0.66 bb/h and 27.9865 ± 0.63 bb/h, respectively. On the Leduc game, the NFSP-UCT loses the least compared with Random and DeepCFR when they are against NFSP-PER. This is because the scale of the Leduc game is relatively small; the UCT in NFSP-UCT can be simulated under limited conditions to obtain a better strategy. However, in the HUNL game, the NFSP-UCT loses the most compared with Random and DeepCFR methods when compared with NFSP-PER. This is because the scale of the HUNL game is very large and contains four rounds; the UCT in NFSP-UCT cannot be simulated under limited conditions, resulting in a poor final strategy.

Table 4. Comparison results of NFSP-PLT. 'NFSP-PLT vs. Random' is similar to 'NFSP-PER vs. Random' in Table 3.

Game	NFSP-PLT vs. Random	NFSP-PLT vs. NFSP-UCT	NFSP-PLT vs. DeepCFR
Leduc	1.0513 ± 0.05	0.0189 ± 0.09	0.2633 ± 0.07
HUNL	34.0445 ± 0.48	47.0840 ± 0.32	28.0418 ± 0.63

Table 4 shows the comparison experiment results of NFSP-PLT vs. comparison methods. The game performance of NFSP-PLT is also better than the comparison methods: Random, NFSP-UCT and DeepCFR. The average return of 20,000 games on the HUNL was 1.0513 ± 0.05 bb/h, 0.0189 ± 0.09 bb/h and 0.2633 ± 0.07 bb/h, respectively. The average return of 20,000 games on the HUNL was 34.0445 ± 0.48 bb/h, 47.0840 ± 0.32 bb/h and 28.0418 ± 0.63 bb/h, respectively. In this comparative experiment, we can draw a similar conclusion to that in the previous Table 3: among the three comparison methods, NFSP-UCT performed best in the Leduc game and worst in the HUNL game. The reason for this is the same as that described in the previous paragraph.

In addition, NFSP-PLT is slightly better than NFSP-PER, which shows that our further improvement in NFSP-PER is also effective. For example, when playing vs. NFSP-UCT on

the HUNL, the average return of NFSP-PLT is 47.0840 $\pm$ 0.32 bb/h, which is more than that of NFSP-PER, 36.7131 $\pm$ 0.66 bb/h.

The second group of experiments was conducted on the 3-NH and 6-NH games. It should be noted that the comparison method DeepCFR was not used in this experiment. This is because DeepCFR only solves two-player IIGs, and is not applicable in multiplayer IIGs. Thus, in this experiment, only the two methods Random and NFSP-UCT were used to test. Table 5 shows the results after testing the proposed methods on two games: 3-NH and 6-NH (3-NH is three-player No-limit hold'em and six-player No-limit hold'em). For example, the number on the Row 'NFSP-PER', '3-NH' and column 'vs. Random' represents that the return of 'NFSP-PER vs. Random' on the game '3-NH'. 'NFSP-PER vs. Random' is similar to 'NFSP-PER vs. Random' in Table 3.

Table 5. Comparison results on the 3-NH and 6-NH.

	Game	vs. Random	vs. NFSP-UCT
NFSP-PER	3-NH	58.3802 $\pm$ 1.12	11.3967 $\pm$ 1.42
	6-NH	84.5855	28.5454 $\pm$ 2.18
NFSP-PLT	3-NH	61.3869 $\pm$ 1.08	11.7747 $\pm$ 1.42
	6-NH	85.4392	47.0840 $\pm$ 0.32

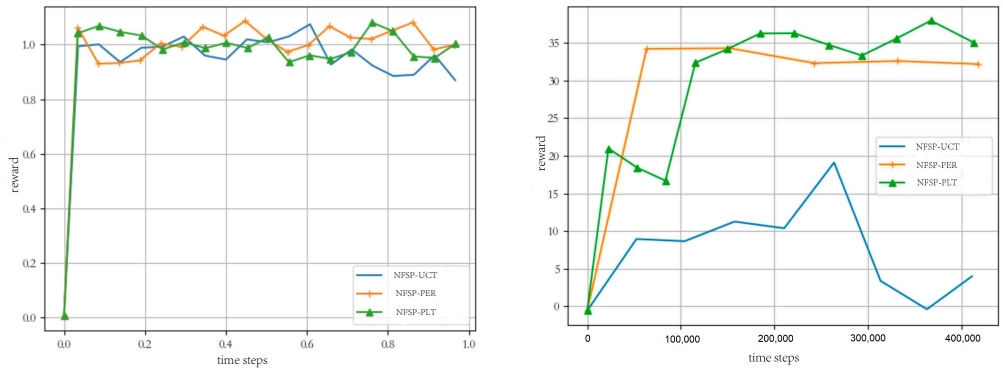
The experimental results are shown in Table 5. Our method performed well in two multiplayer games: 3-NH and 6-NH. For the NFSP-PER, the average return won in 20,000 games on the 3-NH was 58.3802 $\pm$ 1.12 bb/h and 11.3967 $\pm$1.42 bb/h, respectively. The average return won in 20,000 games on the 6-NH was 84.5855 bb/h and 28.5454 $\pm$ 2.18 bb/h, respectively. For the NFSP-PLT, the average return won in 20,000 games on the 3-NH was 61.3869 $\pm$ 1.08 bb/h and 11.7747 $\pm$ 1.42 bb/h, respectively. The average return won in 20,000 games on the 6-NH was 85.4392 bb/h and 47.0840 $\pm$ 0.32 bb/h, respectively. In this experiment, for NFSP-UCT, we found that, as the number of players increased, more was lost. Compared with the three-player game 3-NH, more was lost in the six-player game 6-NH. This further shows that our previous analysis is reasonable. That is, in a larger game, the UCT in NFSP-UCT can not be simulated well under limited conditions, which results in a poor final strategy. In addition, we found that the performance of NFSP-PLT is better than that of NFSP-PER in both the three-player game 3-NH and six-player game 6-NH. The reason for this is that, compared with NFSP-PER, NFSP-PLT wins more returns when playing games with the same comparison method, which further verifies the effectiveness of NFSP-PLT.

After the previous comparative experiments, in general, our proposed method has indeed improved sample utilization efficiency compared to existing methods. Moreover, the introduction of adaptive UCT further improves the performance of the solving strategy. The experimental results also demonstrate that our method can effectively solve two-player and multiplayer strategies in the field of IIGs. Of course, there are still some areas for improvement in our method. Firstly, the computation time still needs to be reduced. Although our strategy performance has improved after the introduction of UCT, it still requires considerable computational resources and time. Secondly, our method lacks theoretical guarantees in multiplayer games. Although NFSP has theoretical guarantees in two-player games that the strategy is an approximate Nash equilibrium strategy, there is no theoretical guarantee in multiplayer games, and the method we proposed is also the same. Finally, the proposed method has only been validated in poker games, especially in Texas Hold'em poker, and its expansion to other types of applications requires further research.

5.3. Experimental Results

In order to verify the effectiveness of the method proposed in the paper, we conducted experiments in this section. First, we verified the effectiveness of the proposed methods NFSP-PER and NFSP-PLT, respectively. The method we proposed improves the efficiency of sample utilization, which is embodied in the training process. To this end, we conducted

a game evaluation on the training process methods of Leduc and HUNL. Specifically, 50,000 rounds of training were conducted on the Leduc and the HUNL platforms for NFSP-PER, NFSP-PLT and NFSP-UCT agents, respectively. During the training process, a conduct evaluation was carried out, with random agents every 1000 rounds. A total of 10,000 games were conducted for each evaluation. The reward curve of the three agents playing with random agents in Leduc and the HUNL is shown in Figure 2.



(**a**) (**b**)

Figure 2. Experimental results results of the proposed method. The Y-axis represents the reward and the X-axis represents the number of timesteps. The higher the reward, the better the method. (**a**) Evaluation in training process on Leduc; (**b**) Evaluation in training process on the HUNL.

As shown in Figure 2a in the Leduc, the three different agents have little difference in game level during training, and NFSP-PER and NFSP-PLT are slightly dominant. As shown in Figure 2b in the HUNL, the game level of NFSP-PER and NFSP-PLT greatly improved compared with NFSP-UCT. At the same time, in the late stage of training, NFSP-PLT won the most returns and the highest level of intelligence. The experimental results fully verify the effectiveness of our proposed method.

Secondly, we conducted experiments on the HUNL to test the effectiveness of the adaptive UCT algorithm. The traditional UCT algorithm usually specifies a fixed number of iteration rounds in the iterative solution strategy. However, the limited iteration time, which is set manually, often cannot meet the actual needs. There is a contradiction between the accuracy of gamestate evaluation and the limited iterative simulation time, and both of them are crucial for solving game strategies. When measuring the influence of adaptive threshold and confidence on the strategy solution, the baseline algorithm selected is UCT_3000; that is, the UCT algorithm with a fixed number of iterations of 3000.

By setting different adaptive thresholds and confidence levels to match with UCT_3000 for comparative experiments, the solution accuracy and time consumption of the adaptive UCT algorithm was obtained with different thresholds and confidence levels, as shown in Table 6. For example, $UCT(2800, 0.98)$ indicates that the adaptive threshold is 2800 and the confidence level is 0.98, where the unit time consumption indicates the number of milliseconds spent in each search and solution action. Experiments show that the adaptive UCT algorithm can better balance its solution speed and accuracy. The online search time is reduced while maintaining the strength of the strategy. The adaptive UCT algorithm not only relies on longer iteration rounds to evaluate the value solution strategy more accurately, but also prunes by eliminating significant inferior nodes to improve the efficiency of the UCT solution. Finally, $UCT(2800, 0.98)$ was used in the paper.

Table 6. Ablation results of adaptive UCT.

Different Settings	Accuracy	Unit Time Consumption
UCT_3000	1	37.22
$UCT(2800, 0.98)$	0.976	21.95
$UCT(2800, 0.95)$	0.971	21.76
$UCT(2800, 0.90)$	0.968	22.08
$UCT(2500, 0.98)$	0.934	20.86
$UCT(2500, 0.95)$	0.936	19.61
$UCT(2500, 0.90)$	0.933	20.29
$UCT(2000, 0.98)$	0.903	16.07
$UCT(2000, 0.95)$	0.901	16.03
$UCT(2000, 0.90)$	0.902	16.02

6. Conclusions

In this paper, we propose a new NFSP-based method NFSP-PER, studying the strategy of IIGs. The proposed NFSP-PER can not only solve the strategy of two-player IIGs, but can also solve the strategy of multiplayer IIGs. NFSP-PER combines NFSP with prioritized experience replay, which enhances the sample learning efficiency by optimizing sampling order of training samples. In addition, NFSP-PLT is proposed to control the utilization degree of priority-weighted samples through the learning times of samples. Furthermore, based on the NFSP-PLT, an adaptive UCT is applied to solve the optimal response that making the strategy more accurate. Extensive experimental results show that the proposed NFSP-PLT effectively improves the sample learning efficiency compared with the existing works.

In the future, we will consider that there are two aspects worthy of further study: First of all, the strategy solution of a large-scale game based on the NFSP method, which mainly aims to achieve an effective strategy solution to large-scale multi-player game problems. Secondly, application research based on the NFSP method, which mainly applies this kind of method to other, related applications.

Author Contributions: Conceptualization, H.L. and S.Q.; methodology, H.L. and D.Z.; software, D.Z. and J.Z.; validation, L.Y., X.W. and Q.L.; formal analysis, J.X.; writing—original draft preparation, H.L. and S.Q.; writing—review and editing, L.Y., X.W. and Q.L.; supervision, L.Y. and J.X. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by key fields R&D project of Guangdong Province (No.2020B0101-380001), Guangdong Provincial Key Laboratory of Novel Security Intelligence Technologies (2022B121-2010005), Shenzhen Foundational Research Funding under Grant (JCYJ20200109113427092, 2020080517-3048001), Basic Research Programs of Taicang, 2022 (TC2022JC14), Fundamental Research Funds for the Central Universities (G2022WD01027, NWPU), PINGAN-HITsz Intelligence Finance Research Center, Ricoh-HITsz Joint Research Center, GBase-HITsz Joint Research Center.

Data Availability Statement: Not applicable.

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

References

1. Nash, J. Non-cooperative games. *Ann. Math.* **1951**, *54*, 286–295. [CrossRef]
2. Zinkevich, M.; Johanson, M.; Bowling, M.; Piccione, C. Regret minimization in games with incomplete information. *Adv. Neural Inf. Process. Syst.* **2007**, *20*, 1729–1736.
3. Brown, G.W. Iterative solution of games by fictitious play. *Act. Anal. Prod. Alloc.* **1951**, *13*, 374.
4. Moravčík, M.; Schmid, M.; Burch, N.; Lisỳ, V.; Morrill, D.; Bard, N.; Davis, T.; Waugh, K.; Johanson, M.; Bowling, M. Deepstack: Expert-level artificial intelligence in heads-up no-limit poker. *Science* **2017**, *356*, 508–513. [CrossRef]
5. Brown, N.; Sandholm, T. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science* **2018**, *359*, 418–424. [CrossRef] [PubMed]
6. Brown, N.; Lerer, A.; Gross, S.; Sandholm, T. Deep counterfactual regret minimization. In Proceedings of the International Conference on Machine Learning, Long Beach, CA, USA, 9–15 June 2019; pp. 793–802.

7. Schmid, M.; Burch, N.; Lanctot, M.; Moravcik, M.; Kadlec, R.; Bowling, M. Variance reduction in monte carlo counterfactual regret minimization (VR-MCCFR) for extensive form games using baselines. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 29–31 January 2019; Volume 33, pp. 2157–2164.
8. Liu, W.; Jiang, H.; Li, B.; Li, H. Equivalence analysis between counterfactual regret minimization and online mirror descent. In Proceedings of the International Conference on Machine Learning, Baltimore, MD, USA, 17–23 July 2022; pp. 13717–13745.
9. Liu, W.; Li, B.; Togelius, J. Model-free neural counterfactual regret minimization with bootstrap learning. *IEEE Trans. Games* **2022**. [CrossRef]
10. Cotae, P.; Reindorf, N.E.A. Using counterfactual regret minimization and Monte Carlo tree search for cybersecurity threats. In Proceedings of the 2021 IEEE International Black Sea Conference on Communications and Networking (BlackSeaCom), Bucharest, Romania, 24–28 May 2021; pp. 1–6.
11. Zhang, H.; Lerer, A.; Brown, N. Equilibrium Finding in Matrix Games via Greedy Regret Minimization. *Proc. Aaai Conf. Artif. Intell.* **2022**, *36*, 9484–9492.
12. Wang, Z.; Mu, C.; Hu, S.; Chu, C.; Li, X. Modelling the dynamics of regret minimization in large agent populations: A master equation approach. In Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, Vienna, Austria, 23–29 July 2022; pp. 534–540.
13. Heinrich, J.; Lanctot, M.; Silver, D. Fictitious self-play in extensive-form games. In Proceedings of the International Conference on Machine Learning, Lille, France, 6–11 July 2015; pp. 805–813.
14. Heinrich, J.; Silver, D. Deep reinforcement learning from self-play in imperfect-information games. *arXiv* **2016**, arXiv:1603.01121.
15. Montague, P.R. Reinforcement learning: An introduction, by Sutton, RS and Barto, AG. *Trends Cogn. Sci.* **1999**, *3*, 360. [CrossRef]
16. Sutton, R.S.; Barto, A.G. *Reinforcement Learning: An Introduction*; MIT Press: Cambridge, MA, USA, 2018.
17. 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.
18. 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]
19. Montufar, G.F.; Pascanu, R.; Cho, K.; Bengio, Y. On the number of linear regions of deep neural networks. *Adv. Neural Inf. Process. Syst.* **2014**, *27*, 2924–2932.
20. Xue, W.; Zhang, Y.; Li, S.; Wang, X.; An, B.; Yeo, C.K. Solving large-scale extensive-form network security games via neural fictitious self-play. *arXiv* **2021**, arXiv:2106.00897.
21. Chen, Y.; Zhang, L.; Li, S.; Pan, G. Optimize neural fictitious self-play in regret minimization thinking. *arXiv* **2021**, arXiv:2104.10845.
22. Zhang, L.; Chen, Y.; Wang, W.; Han, Z.; Li, S.; Pan, Z.; Pan, G. A Monte Carlo Neural Fictitious Self-Play approach to approximate Nash Equilibrium in imperfect-information dynamic games. *Front. Comput. Sci.* **2021**, *15*, 155334. [CrossRef]
23. Ganzfried, S. Fictitious play outperforms counterfactual regret minimization. *arXiv* **2020**, arXiv:2001.11165.
24. Kamra, N.; Gupta, U.; Wang, K.; Fang, F.; Liu, Y.; Tambe, M. Deep Fictitious Play for Games with Continuous Action Spaces. In Proceedings of the AAMAS, Montreal, QC, Canada, 13–17 May 2019; pp. 2042–2044.
25. Perolat, J.; Piot, B.; Pietquin, O. Actor-critic fictitious play in simultaneous move multistage games. In Proceedings of the International Conference on Artificial Intelligence and Statistics, Playa Blanca, Spain, 9–11 April 2018; pp. 919–928.
26. He, K.; Wu, H.; Wang, Z.; Li, H. Finding nash equilibrium for imperfect information games via fictitious play based on local regret minimization. *Int. J. Intell. Syst.* **2022**, *37*, 6152–6167. [CrossRef]
27. Han, Y.; Zhou, Q.; Duan, F. A game strategy model in the digital curling system based on NFSP. *Complex Intell. Syst.* **2022**, *8*, 1857–1863. [CrossRef]
28. Yee, T.; Lisý, V.; Bowling, M.H. Monte Carlo Tree Search in Continuous Action Spaces with Execution Uncertainty. In Proceedings of the IJCAI, New York, NY, USA, 9–15 July 2016; pp. 690–697.
29. Schaul, T.; Quan, J.; Antonoglou, I.; Silver, D. Prioritized experience replay. *arXiv* **2015**, arXiv:1511.05952.
30. Vitter, J.S. Random sampling with a reservoir. *ACM Trans. Math. Softw. (TOMS)* **1985**, *11*, 37–57. [CrossRef]
31. 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]
32. Gelly, S.; Wang, Y.; Munos, R.; Teytaud, O. Modification of UCT with Patterns in Monte-Carlo Go. Ph.D. Thesis, INRIA, Paris, France, 2006.
33. Kocsis, L.; Szepesvári, C. Bandit based monte-carlo planning. In *ECML 2006: Machine Learning: ECML 2006, Proceedings of the European Conference on Machine Learning, European Conference on Machine Learning, Berlin, Germany, 18–22 September 2006*; Springer: Berlin/Heidelberg, Germany, 2006; pp. 282–293.
34. Russo, D. A note on the equivalence of upper confidence bounds and gittins indices for patient agents. *Oper. Res.* **2021**, *69*, 273–278. [CrossRef]
35. Couëtoux, A.; Hoock, J.B.; Sokolovska, N.; Teytaud, O.; Bonnard, N. Continuous upper confidence trees. In *LION 2011: Learning and Intelligent Optimization, Proceedings of the International Conference on Learning and Intelligent Optimization, Rome, Italy, 17–21 January 2011*; Springer: Berlin/Heidelberg, Germany, 2011; pp. 433–445.
36. Saffidine, A.; Cazenave, T.; Méhat, J. UCD: Upper Confidence bound for rooted Directed acyclic graphs. *Knowl.-Based Syst.* **2012**, *34*, 26–33. [CrossRef]

37. Lee, K.M.B.; Kong, F.; Cannizzaro, R.; Palmer, J.L.; Johnson, D.; Yoo, C.; Fitch, R. An upper confidence bound for simultaneous exploration and exploitation in heterogeneous multi-robot systems. In Proceedings of the 2021 IEEE International Conference on Robotics and Automation (ICRA), Xi'an, China, 30 May–5 June 2021; pp. 8685–8691.
38. Roy, K.; Zhang, Q.; Gaur, M.; Sheth, A. Knowledge infused policy gradients with upper confidence bound for relational bandits. In *ECML PKDD 2021: Machine Learning and Knowledge Discovery in Databases. Research Track, Proceedings of the Joint European Conference on Machine Learning and Knowledge Discovery in Databases, Bilbao, Spain, 13–17 September 2021*; Springer: Cham, Switzerland, 2021; pp. 35–50.
39. Osborne, M.J.; Rubinstein, A. *A Course in Game Theory*; MIT Press: Cambridge, MA, USA, 1994.
40. Li, H.; Qi, S.; Zhang, J.; Zhang, D.; Yao, L.; Wang, X.; Li, Q.; Xiao, J. NFSP-PER: An efficient sampling NFSP-based method with prioritized experience replay. In Proceedings of the 2022 the 4th International Conference on Data Intelligence and Security (ICDIS), Shenzhen, China, 24–26 August 2022; pp. 388–393.
41. Zhang, J.; Sun, X.; Zhang, D.; Wang, X.; Qi, S.; Qian, T. Fittest survival: An enhancement mechanism for Monte Carlo tree search. *Int. J. Bio Inspired Comput.* **2021**, *18*, 122–130. [CrossRef]