Scalable, Generalizable, and Offline Methods for Imperfect-Information Extensive-Form Games

Shuxin Li

College of Computing and Data Science

A thesis submitted to the Nanyang Technological University
in partial fulfillment of the requirements for the degree of
Doctor of Philosophy

2025

Statement of Originality

I hereby certify that the work embodied in this thesis is the result of original research, is free of plagiarised materials, and has not been submitted for a higher degree to any other University or Institution.

08-Aug-2024

Date

Shuxin Li

Supervisor Declaration Statement

I have reviewed the content and presentation style of this thesis and declare it is free of plagiargism and of sufficient grammatical clarity to be examined. To the best of my knowledge, the research and writing are those of the candidate except as acknowledged in the Author Attribution Statement. I confirm that the investigations were conducted in accord with the ethics policies and integrity standards of Nanyang Technological University and that the research data are presented honestly and without prejudice.

08-Aug-2024

Date

Prof. Bo An

Authorship Attribution Statement

This thesis contains materials from 3 papers published in the following peer-reviewed conferences, and 1 paper under review, in which I am listed as an author.

Chapter 3 is published as **Shuxin Li**, Youzhi Zhang, Xinrun Wang, Wanqi Xue, Bo An. CFR-MIX: Solving Imperfect Information Extensive-Form Games with Combinatorial Action Space. *Proceedings of the 30th International Joint Conference on Artificial Intelligence* (**IJCAI**), 2021.

The contributions of the co-authors are as follows:

- Prof. An proposed the initial research direction.
- Youzhi, Xinrun, Wanqi, and I discussed and designed the methodology.
- I wrote the manuscript drafts and conducted all experiments.
- Prof. An, Youzhi, and Xinrun provided insightful comments on the manuscript and revised the drafts.

Chapter 4 is published as **Shuxin Li**, Xinrun Wang, Youzhi Zhang, Wanqi Xue, Jakub Cerny, Bo An. Solving Large-Scale Pursuit-Evasion Games using Pre-trained Strategies. *Proceedings of the 37th AAAI Conference on Artificial Intelligence* (**AAAI**), 2023.

The contributions of the co-authors are as follows:

- Prof. An and Xinrun proposed the initial research direction.
- Xinrun, Youzhi, and I discussed and designed the methodology.
- I wrote the manuscript drafts and conducted the main experiments.
- Wanqi assisted in conducting experiments.
- Prof. An and Jakub Cerny provided insightful comments on the manuscript and revised the drafts.

Chapter 5 is published as Pengdeng Li*, **Shuxin Li***, Xinrun Wang, Jakub Cerny, Youzhi Zhang, Stephen McAleer, Hau Chan, Bo An. Grasper: A generalist pursuer for pursuit-evasion problems. *Proceedings of the 23rd International Conference on Autonomous Agents and Multi-Agent Systems* (**AAMAS**), 2024. The contributions of the co-authors are as follows:

- Pengdeng, Xinrun, and I proposed the research direction and initial solution.
- Jakub Cerny and Youzhi discussed with me to determine the final solution.
- Pengdeng and I wrote the manuscript drafts and conducted all experiments.

- Prof. An, Hau Chan, and Stephen McAleer provided insightful comments on the manuscript and revised the drafts.

Chapter 6 is under review as **Shuxin Li**, Youzhi Zhang, Jakub Cerny, Pengdeng Li, Chang Yang, Xinrun Wang, Hau Chan, Bo An. Offline Equilibrium Finding in Extensive-form Games: Datasets, Theory, and Methods.

The contributions of the co-authors are as follows:

- Xinrun and I proposed the initial research direction.
- Youzhi, Jakub Cerny, Pengdeng, Xinrun, and I discussed and designed the methodology.
- I wrote the manuscript drafts and conducted all experiments.
- Prof. An, Chang Yang, and Hau Chan provided insightful comments on the manuscript and revised the drafts.

08-Aug-2024

Date

Shuxin Li

Acknowledgements

This doctoral thesis marks the culmination of a long and memorable journey. Along the way, I have received support and assistance from many individuals. I would like to express my heartfelt gratitude to all those who have supported and guided me throughout my academic journey.

First and foremost, I wish to express my deepest appreciation to my supervisor, Prof. Bo An, who provided me with the invaluable opportunity to study under his supervision. During my doctoral journey, he always offered patient and thoughtful guidance whenever I encountered difficulties or made slow progress. His unwavering support and encouragement have been a constant source of strength, enabling me to persevere through challenges. Prof. Bo An's insightful feedback has continually pushed me to refine my ideas and improve my work, while his constant motivation has kept me focused and driven. His dedication to my academic and personal growth has significantly contributed to the successful completion of my research, and I am profoundly grateful for that.

I am also immensely grateful to my Thesis Advisory Committee (TAC) members: Prof. Yi Li and Prof. Zinovi Rabinovich. Their constructive feedback and valuable suggestions have been instrumental in shaping and refining my research.

Throughout my PhD journey, I have had the honor of collaborating with a group of exceptional researchers: Prof. Xinrun Wang, Prof. Youzhi Zhang, Dr. Pengdeng Li, Dr. Yakub Cerny, Dr. Wanqi Xue, Dr. Stephen McAleer, Prof. Hau Chan, and Ms. Chang Yang. Thank you for your insightful discussions, valuable feedback, and unwavering support. Your expertise and guidance have greatly enriched my research, and your encouragement has motivated me to overcome numerous challenges. I am deeply grateful for the opportunity to learn from and work with such distinguished scholars. Your contributions have been instrumental in the successful completion of my doctoral thesis.

I would like to extend my gratitude to the talented members of our AMI group who accompanied me on my doctoral journey: Mengchen Zhao, Jiuchuan Jiang, Xinrun Wang, Youzhi Zhang, Xu He, Lei Feng, Aye Phyu, Yanchen Deng, Hongxin Wei, Rundong Wang, Wei Qiu, Yakub Cerny, Feifei Lin, Runsheng Yu, Pengdeng Li, Wanqi Xue, Shenggong Ji, Wei Tao, Zhuyun Yin, Renchunzi Xie, Shuo Sun, Shufeng Kong, Hao Cheng, Chaojie Wang, Pengjie Gu, Xinyu Cai, Yewen Li, Zhenghai Xue, Molei Qin, Longtao Zheng, Sheng Zang, Haochong Xia, Chuqiao Zong, Ying Zheng, Dong Xing, Shanqi Liu, Wentao Zhang, Ruhao Jiang, Yuzhou Cao, Weihao Tan, Zitao Song, Penghui Yang, Qiwei, Yiwen Zhu, Zhenxing Ge, Xing Chen, Guoxi Chen, Yitian Hong. Your camaraderie, support, and intellectual discussions have enriched my research experience and contributed significantly to my academic growth. The moments we shared, both in and out of the lab, have been invaluable and have provided a strong foundation for my work. Thank you all for your encouragement, assistance, and friendship.

Lastly, I owe my deepest gratitude to my family, including my parents, sister, and my husband. Your unconditional love, patience, and support have been my pillars of strength. Your encouragement has given me the confidence to persevere through challenges and achieve my goals.

Thank you all for being an integral part of this journey. Your support and belief in me have made this accomplishment possible, and I am eternally grateful.

Abstract

Imperfect-information extensive-form games (IIEFGs) offer a versatile framework for modeling interactions between multiple players in stochastic and imperfect-information scenarios. Due to their superior modeling capabilities, IIEFGs can be applied in a wide range of fields, including economics and computer science. Given their broad application, IIEFGs have become a vital area of research in game theory. There are numerous algorithms developed to solve IIEFGs, ranging from traditional programming-based to advanced deep learning-based methods. Although these algorithms have been proven successful, they still suffer from scalability and generalizability issues when solving large-scale games. Furthermore, these algorithms require continuous interaction with the game environment, which impedes the applications in many real-world scenarios where the environment may be impractical. To mitigate these issues, this thesis is devoted to improving the scalability and generalizability and extending offline learning to the IIEFGs setting.

Our first emphasis is on improving the scalability of existing algorithms to solve IIEFGs. We propose two scalable algorithms: CFR-MIX and PSRO with pre-trained strategies which are based on the counterfactual regret minimization (CFR) and policy space response oracle (PSRO) frameworks, respectively. CFR-MIX is designed to address the exponential combinatorial action space problem in a special type of IIEFGs known as team-adversary games. In the CFR-MIX algorithm, we first propose a new strategy representation that represents a joint action strategy using the individual strategies of all agents, maintaining cooperation between agents through a consistency relationship. To compute the equilibrium with the new strategy representation under the CFR framework, we transform the strategy consistency relationship into a consistency relationship between cumulative regret values. We then introduce a novel decomposition method for cumulative regret values to ensure this consistency. CFR-MIX algorithm employs a mixing layer to implement the decomposition method by estimating cumulative regret values of joint actions as a non-linear combination of the cumulative regret values of individual actions. Experimental results show that CFR-MIX significantly outperforms

existing algorithms. PSRO with pre-trained strategies method is designed to solve the long-term decision-making issue in large-scale pursuit-evasion games (PEGs), the special type of two-player zero-sum IIEFGs. This algorithm integrates the pre-training and fine-tuning paradigm into the PSRO framework. Specifically, we first pre-train the pursuer's policy base model against many different strategies of the evader. Then we proceed with the PSRO loop and fine-tune the pre-trained policy to attain the pursuer's best responses. Empirical evaluation shows that our approach significantly outperforms baselines in terms of speed and scalability.

The next focus is on enhancing the generalizability of existing algorithms for solving PEGs. We propose a generalizable framework, Grasper, designed to generate pursuer policies tailored to specific PEGs. Grasper features a novel architecture with two components: a graph neural network (GNN) to encode PEGs into hidden vectors and a hypernetwork to generate pursuer policies based on these hidden vectors. We develop a three-stage training pipeline involving a pre-pretraining stage for training the GNN through the GraphMAE algorithm, a pre-training stage for training the hypernetwork by utilizing heuristic-guided multi-task learning, and a fine-tuning stage for fine-tuning the pursuer base model generated by the hypernetwork to compute the best response strategy. Experimental results demonstrate that Grasper outperforms baselines in both solution quality and generalizability.

The final emphasis is on applying offline learning to game solving. We introduce the offline equilibrium finding (Offline EF) paradigm, which computes equilibrium strategies from offline datasets. Then we construct diverse datasets encompassing a range of games using several methods. These datasets serve as a basis for evaluating the performance of offline algorithms. Next, we design a novel framework, BOMB, which integrates behavior cloning and model-based methods along with a novel parameter estimation method. The model-based method adapts any online EF algorithm to the offline setting. Then, our theoretical analysis provides performance guarantees of BOMB across various datasets. Experiments demonstrate BOMB's superior efficiency over offline RL methods in computing equilibrium strategies.

To conclude, this doctoral thesis addresses key challenges in solving imperfect-information extensive-form games, focusing on scalability, generalizability, and offline learning. Through innovative algorithms like CFR-MIX, PSRO with pre-trained strategies, Grasper, and BOMB, it advances the state of the art, offering efficient and practical solutions for both theoretical and real-world applications.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

The extensive-form games are a fundamental model in game theory, providing a robust framework for modeling sequential decision-making processes in environments with multiple interacting agents [1]. These games are characterized by their ability to represent complex scenarios involving imperfect information, where players may have incomplete knowledge about the game state or the actions taken by other players. The canonical solution concept for extensive-form games is Nash Equilibrium (NE) [2], where no player can increase his utility by unilaterally deviating.

The application of extensive-form games spans a wide range of fields. In economics, extensive-form games are used to model market competition, auctions, and bargaining scenarios [3–5]. In computer science, extensive-form games are crucial in the development of algorithms for artificial intelligence, particularly in strategic games and automated negotiation systems [6–8]. Extensive research has been conducted on solving imperfect-information extensive-form games, with several notable methods emerging. These include computing NEs using linear programming [9], double oracle algorithms [10, 11], counterfactual regret minimization (CFR) [12], and policy space response oracles (PSRO) [13]. These scalable algorithms have achieved significant accomplishments in practical applications. For instance, double oracle algorithms have been effectively applied to real-world attacker-defender scenarios [10, 11]. The heads-up limit hold'em poker, a classical imperfect-information extensive-form game, was essentially solved using CFR algorithm [14]. Building

on this success, substantial progress has been made in heads-up no-limit hold'em poker with the development of advanced CFR-based systems such as Libratus [15] and DeepStack [8]. These achievements highlight the effectiveness of scalable algorithms in addressing complex, imperfect-information games, paving the way for future advancements in both theoretical and practical aspects of game theory.

Specifically, imperfect-information extensive-form games can also be used to model security-related scenarios [16, 17]. As is well-known, ensuring the safety and security of the population is a critical task for governmental law enforcement agencies, which involves preventing, deterring, and detecting crime. Many game-theoretic models have been developed to safeguard not only the general public but also critical infrastructure using only limited resources [18]. Among these security challenges, one stands out due to its unflattering statistic: police pursuits probably "injure or kill more innocent bystanders than any other kind of force" [19]. It is thus imperative to devise methods for efficiently coordinating multiple police units to capture a fleeing criminal swiftly, minimizing casualties and property damage. This scenario is commonly referred to as the *pursuit-evasion game*, a special type of two-player zero-sum imperfect-information extensive-form game. However, solving such imperfect-information extensive-form games presents significant computational challenges, particularly in large-scale scenarios with numerous possible actions and outcomes. Developing efficient algorithms for computing equilibria in these games is a critical area of research.

Given the wide applicability of imperfect-information extensive-form games and the complexity of solving them, extensive-form games continue to be a vital area of research in game theory, with ongoing advancements contributing to both theoretical insights and practical applications. In this thesis, we focus on providing efficient algorithms for solving imperfect-information extensive-form games, including pursuit-evasion games, a special case of extensive-form games.

1.2 Motivation

Due to the long history of imperfect-information extensive-form games, numerous algorithms have been developed to solve them. These algorithms range from traditional programming-based methods to advanced deep learning-based techniques.

Contemporary state-of-the-art algorithms for solving imperfect-information extensive-form games can be classified into two groups: no-regret methods derived from counterfactual regret minimization (CFR) [12] and incremental strategy-space generation methods of the policy space response oracle framework (PSRO) [13]. Although these algorithms have demonstrated success, they still face significant challenges in scalability and generalizability.

First, for large-scale imperfect-information extensive-form games, especially games with numerous possible actions, CFR and its sampling-based variants [20] will quickly run out of memory since they use a tabular form to record the regret value and average strategy. For algorithms using deep neural networks, e.g., Deep CFR [21] and Double Neural CFR [22], it is still ineffective to train the strategy network over the large action space. PSRO-based algorithms similarly struggle to solve large-scale imperfect-information extensive-form games effectively due to the high computational cost of repeated best response computations, which are extremely time-consuming in large-scale games. To mitigate these issues, we propose two algorithms, based on the CFR and PSRO frameworks, to improve their scalability.

Even with scalable solutions, many existing equilibrium-finding algorithms lack generalizability. They are often tailored to specific games. Even minor changes in the game settings require recomputation of the equilibrium strategy from scratch, which is both computationally expensive and time-consuming. This limitation highlights the need for generalizable approaches capable of adapting to different games with varying initial conditions without complete recomputation. To do this, we propose a novel algorithm that introduces a hypernetwork to generate base policies for different games, thereby avoiding the need for complete recomputation.

While our efforts to improve scalability and generalizability address key challenges in solving imperfect-information extensive-form games, a critical limitation remains: the reliance on continuous interaction with the game environment or an accurate simulator. This dependency hinders real-world applications where online interactions are impractical or prohibitively expensive. For example, CFR-based algorithms necessitate traversing the game tree to compute regret values, and PSRO and its variants demand simulations within the game environment to compute the best response oracle and estimate the entries in the meta-game. We refer to this paradigm of computing equilibrium strategies as "*online equilibrium finding*". However, in many real-world applications, such as predicting financial

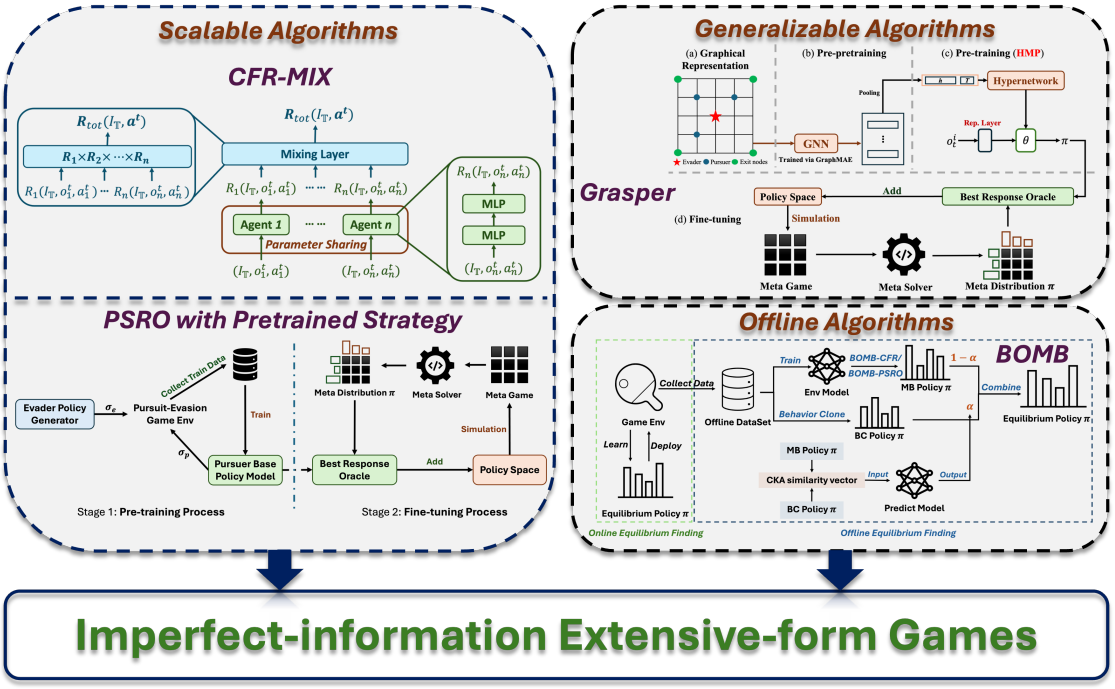


FIGURE 1.1: The main works of this thesis.

market volatility [23], network intrusion detection [24], and automated negotiations [25], the immediate interaction with the environment may be impractical or excessively costly. To address this issue, we propose the novel *offline equilibrium finding* (Offline EF) paradigm, which aims to compute the equilibrium strategies of imperfect-information extensive-form games using offline datasets.

1.3 Contributions

In this thesis, we focus on improving algorithms for solving imperfect-information extensive-form games in terms of scalability and generalizability. Figure 1.1 shows the main works of this thesis. There are three parts: scalable, generalizable, and offline algorithms. In conjunction with this figure, we summarized the key contributions of this thesis as follows.

i) We propose a new framework of CFR, ***CFR-MIX***, which aims to improve scalability by addressing the *exponential combinatorial action space problems* in a special type of imperfect-information extensive-form games known as team-adversary games. To do this, we first propose a novel strategy representation that represents the team's joint action strategy using the individual strategy of each agent inspired

by the idea of centralized training with decentralized execution [26]. Based on this strategy representation, we define a consistency relationship between these two strategy representations to maintain the cooperation among agents in the team, which guarantees that the equilibrium with the new strategy representation is a special team-maxmin equilibrium. To compute Nash equilibrium based on this individual strategy representation under the CFR framework, we transform the consistency relationship between strategies to the consistency relationship between the cumulative regret values of joint actions and the cumulative regret values of individual actions. Furthermore, a novel decomposition method over cumulative regret values is proposed to guarantee the consistency relationship between the cumulative regret values. To implement the decomposition method empirically, CFR-MIX employs a mixing layer that estimates cumulative regret values of joint actions as a non-linear combination of cumulative regret values of individual actions. To further improve the performance, the parameter sharing technique is applied among all agents in the team, which also reduces the network parameters dramatically. Finally, experimental results show that CFR-MIX outperforms state-of-the-art algorithms significantly on games in different domains.

ii) For the special type of two-player zero-sum imperfect-information extensive-form game, pursuit-evasion game, we propose a two-stage method for solving large-scale pursuit-evasion games by integrating the pre-training and fine-tuning paradigm into the PSRO framework. Specifically, we first modify the graph embedding algorithm LINE to learn a better game state representation that retains both the structure information of the graph and the node information related to the exit nodes. Then we pre-train a pursuer's policy base model against vastly different strategies of the evader with multi-task reinforcement learning. After pre-training, we formulate a novel best-response oracle that fine-tunes the pre-trained pursuer's policy within the PSRO framework. Finally, we perform extensive empirical analysis in large-scale pursuit-evasion games with both synthetic and real-world graphs, showing that ***PSRO with Pretrained Strategy*** algorithm outperforms other baselines in both solution quality and scalability. To the best of our knowledge, we are the first to introduce the pre-training and fine-tuning paradigm into an equilibrium-finding algorithm without using human behavioral data.

iii) To improve the generalizability, we propose ***Grasper***, which is the generalizable framework capable of efficiently providing highly qualified solutions for different

pursuit-evasion games with different initial conditions. To efficiently train the networks within the Grasper algorithm, we propose a three-stage training method: a pre-pretraining stage to train the GNN through GraphMAE [27], a pre-training stage to train the hypernetwork through heuristic-guided multi-task pre-training (HMP), and a fine-tuning stage to obtain the pursuer's best response policy at each PSRO iteration. Finally, we perform extensive experiments, and these results demonstrate the superiority of Grasper over different baselines.

iv) We propose a novel *offline equilibrium finding* (Offline EF) paradigm, which computes the equilibrium strategies only using offline datasets. We first create a collection of diverse offline datasets, including random datasets, expert datasets, learning datasets, and hybrid datasets in different imperfect-information extensive-form games. Then we propose the ***BOMB*** framework, which integrates behavior cloning and model-based methods along with a novel parameter estimation method and the model-based method can incorporate any online equilibrium finding algorithm into the offline context. We also provide a comprehensive theoretical analysis of the BOMB framework, offering performance guarantees under different datasets. Finally, we demonstrate the effectiveness of the BOMB framework in computing equilibrium strategies through extensive experiments on various offline datasets.

1.4 Thesis Organization

The rest of this thesis is organized as follows. Chapter 2 introduces the definition of extensive-form games and discusses two typical classes of algorithms, counterfactual regret minimization (CFR) and policy space response oracle (PSRO), for solving imperfect-information extensive-form games. Chapter 3 presents a novel CFR-based framework to solve the combinatorial action space problem for improved scalability. Chapter 4 focuses on solving a special type of imperfect-information extensive-form games – pursuit-evasion games – by introducing the pre-training and fine-tuning paradigm into the PSRO framework. In Chapter 5, we propose an algorithm for training a generalizable pursuer to solve various pursuit-evasion games. Chapter 6 introduces the offline equilibrium finding setting for imperfect-information extensive-form games and proposes an empirical algorithm for this offline setting. Finally, Chapter 7 concludes the thesis and discusses future works.

Chapter 2

Background

We begin by defining imperfect-information extensive-form games. Following this, we present a literature review on equilibrium finding algorithms, with a particular focus on two widely-used algorithms for solving imperfect-information extensive-form games: the Counterfactual Regret Minimization (CFR) algorithm and the Policy Space Response Oracle (PSRO) algorithm.

2.1 Extensive-form Games

The extensive-form game model is a widely used and versatile framework for modeling problems that involve sequential decision-making by multiple agents [1]. It allows for the representation of the dynamic and interactive nature of such problems. The game is typically represented by a rooted directed tree, where nodes represent decision states, edges denote actions taken by the players, and terminal nodes indicate the endgame utility values assigned to the players. The structure of the tree captures the flow of the game from the initial state through various decision points to the outcomes. Extensive-form games can be classified into *perfect-information extensive-form games* and *imperfect-information extensive-form games* based on the information available to players at the time of their decisions.

In perfect-information extensive-form games, each player has complete knowledge of the entire game state and the sequence of prior actions, ensuring that all players are fully aware of the current situation and the history of moves. Chess and Go are

classical examples of perfect-information games, where all information is visible to all players at all times. Conversely, imperfect-information extensive-form games involve scenarios where players have incomplete or uncertain information about the game state and the history of actions taken by other players. This lack of information requires players to make decisions based on beliefs or estimates about the unknown aspects of the game. A poker game is a quintessential example of an imperfect-information extensive-form game, as players cannot see each other's hands and must infer the hidden information based on observed actions and betting patterns. Next, we will introduce these two types of games in detail, providing a comprehensive understanding of their structures, characteristics, and strategic implications. In both types of games, our discussion will be limited to finite games, specifically those games represented as finite trees.

2.1.1 Perfect-information Extensive-form Games

Informally, a perfect-information extensive-form game, often referred to simply as a perfect-information game, is a tree in the graph-theoretical sense. In this tree, each node corresponds to a decision point for one of the players, each edge represents a possible action, and the leaf nodes represent the outcomes, with each player having a utility function over these outcomes. Following the definition in [9], we can define the perfect-information extensive-form games as follows.

Definition 2.1. A *perfect-information extensive-form game* can be represented by a tuple $\mathcal{G} = (N, H, Z, A, \chi, P, u)$, where

- $N = \{1, ..., n\}$ is a set of n players.
- Since we can unambiguously identify a node within the game tree with its history, that is, the sequence of actions leading from the root node to it, H represents a set of histories, i.e., all possible action sequences.
- Z is a set of terminal histories and it is also a subset of H, i.e., $Z \subseteq H$.
- A is a set of available actions for every non-terminal history.
- $\chi : H \backslash Z \to 2^A$ is the action function, which assigns a set of possible actions to each non-terminal history. For convenience, we use $A(h) = \{a : (h, a) \in H\}$ to refer to the set of available actions for the non-terminal history $h \in H \setminus Z$.

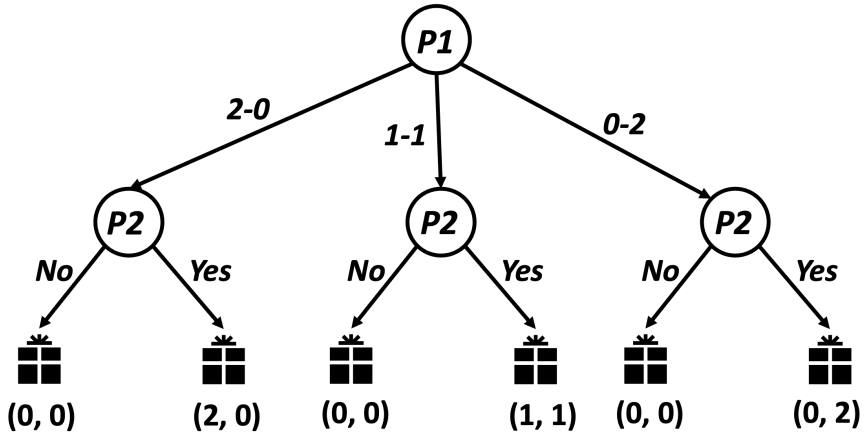


FIGURE 2.1: The Sharing game.

- $P : H \setminus Z \to N$ is a player function, which assigns to each non-terminal history a player. More specifically, $P(h)$ is the player who takes an action at the history $h \in N \setminus Z$, i.e., $P(h) \mapsto N \cup \{c\}$, where c is the "chance player", which represents stochastic events outside of the players' control.
- $u = (u_1, u_2, ..., u_n)$, where $u_i : Z \to \mathbb{R}$ is a real-valued utility function for player i on the terminal histories Z.

A simple example of a perfect-information extensive-form game is the *Sharing game.* Figure 2.1 illustrates the game tree of the sharing game. There are two players (*P1* and *P2*) who must decide how to share two indivisible and identical items. There are three possible split plans to split these two items: *P1* keeps both items (**2-0**), *P2* keeps both items (**0-2**) or they each keep one item (**1-1**). First, the first player (*P1*) proposes a split plan from these three split plans. Then the second player (*P2*) decides whether to accept or reject the proposed split plan. If *P2* accepts the proposal, the items are distributed according to the suggested split. Otherwise, neither player receives any items. Notably, when *P2* makes a decision, *P2* is fully aware of the split plan chosen by *P1*. This characteristic makes it a typical perfect-information extensive-form game.

2.1.2 Imperfect-information Extensive-form Games

In perfect-information extensive-form games, players are allowed to choose their actions at every decision node, implying that they have complete knowledge of the

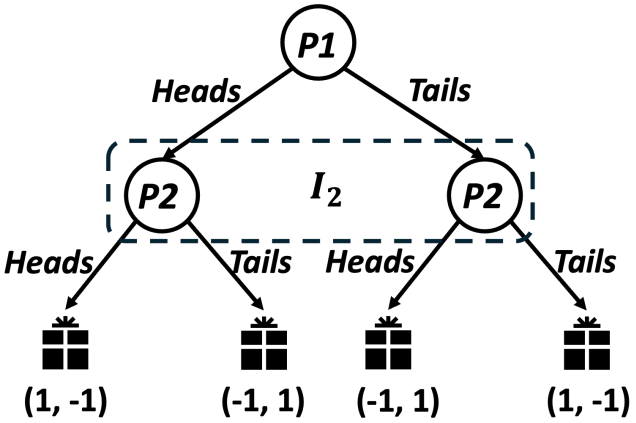


FIGURE 2.2: The Matching pennies game.

current game state and the history of moves that led to it. Thus, in such games, nodes are equated with the sequences of actions (histories) leading to them. However, this assumption of complete knowledge and perfect information may not always be realistic in many practical scenarios. For example, we may need to model agents who must act with partial or no knowledge of the actions taken by others, or even agents with limited memory of their past actions. The imperfect-information extensive-form game model provides a representation of these more general scenarios. This model accounts for situations where players make decisions without full knowledge of all prior actions, enabling the study of more complex and realistic strategic interactions. Following the definition in [9] and the definition of perfect-information extensive-form game, we provide a formal definition of imperfect-information extensive-form games as follows.

Definition 2.2. An *imperfect-information extensive-form game* can be represented by a tuple $\mathcal{G} = (N, H, Z, A, \chi, P, u, \mathcal{I})$, where

- $\mathcal{G} = (N, H, Z, A, \chi, P, u)$ is a perfect-information extensive-form game; and
- $\mathcal{I} = (\mathcal{I}_1, \mathcal{I}_2, ..., \mathcal{I}_n)$, where **information partition** $\mathcal{I}_i = (I_1, ..., I_k)$ forms a partition over histories h where player $i \in N$ takes action (i.e., $\{h \in H \setminus Z : P(h) = i\}$) with the property that $A(h) = A(h')$ whenever h and h' are in the same **information set** $I_i \in \mathcal{I}_i$.

For convenience, we can use $A(I_i)$ to represent the set of actions $A(h)$ and $P(I_i)$ to represent the player $P(h)$ for any history $h \in I_i$. This notation allows us to concisely refer to the actions available and the player making the decision at any

given information set. An example of an imperfect-information extensive-form game is the *Matching Pennies Game* in sequence form. Figure 2.2 illustrates the game tree. There are two players (*P1* and *P2*), each with a penny. First, the first player (*P1*) secretly turns their penny to either heads or tails. Subsequently, the second player (*P2*) makes their decision to turn their penny to heads or tails without knowing the outcome of *P1*'s choice. Note that there is one information set for *P2*, which includes two nodes in the game tree, i.e., two histories that *P2* cannot distinguish them. The game concludes based on the alignment of the pennies. If the pennies match (both heads or both tails), *P1* wins and keeps both pennies. Conversely, if the pennies do not match (one head and one tail), *P2* wins and keeps both pennies. This scenario exemplifies an imperfect-information extensive-form game, as *P2* must decide without knowledge of *P1*'s prior action.

It is important to note that the perfect-information extensive-form games can be considered a special case of the imperfect-information extensive-form games, in which every information set of each information partition is a singleton. In this thesis, we mainly focus on solving imperfect-information extensive-from games.

2.1.3 Nash Equilibrium

Usually, when one says "solving a game", it refers to computing the equilibrium strategy of the game. Specifically, *Nash Equilibrium* (NE) [2] is a common solution concept for imperfect-information extensive-form games due to its robustness. Under the NE strategy, no player has any incentive to unilaterally change their strategy, because any unilateral change in strategy will not lead to a better payoff for them. Next, we will introduce the strategy and the definition of NE strategy.

For an imperfect-information extensive-form game $\mathcal{G}$, the *pure strategy* of the player i is a function that assigns an action in $A(I_i)$ to each information set $I_i \in \mathcal{I}_i$. The *mixed strategy* of the player i is a function that assigns a probability distribution over $A(I_i)$ for every information set $I_i \in \mathcal{I}_i$. We use σ_i to represent the strategy of player i and Σ_i to represent the set of player i's strategies, i.e., $\sigma_i \in \Sigma_i$. The strategy profile σ is represented by a tuple of one strategy for each player, i.e., $\sigma = (\sigma_1, ..., \sigma_n)$ and σ_{-i} refers to all the strategies in σ except σ_i. Given a strategy profile, σ, and assuming players act according to σ, we can compute the reaching probability for every history h, which we denote as $\pi^\sigma(h)$. For example, if $h =$

$(a_1, ..., a_n)$ includes all players actions, where a_i is the action taken by player i, then $\pi^\sigma(h) = \prod_{i=1}^{n} \sigma_i(a_i)$. Based on this reaching probability definition, we can define the expected utility of player i given a strategy profile σ as $u_i(\sigma) = \sum_{h \in Z} \pi^\sigma(h) u_i(h)$.

Given a strategy profile σ, we can compute the *best response strategy* for player i against σ_{-i}, defined as $b(\sigma_{-i}) = \arg\max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma_{-i})$. Under the NE strategy, each player plays their best response strategy. Based on the definition of best response strategy, we can formally define an NE strategy.

Definition 2.3. (*Nash Equilibrium*) Given an imperfect-information extensive-form game $\mathcal{G}$, a strategy profile $\sigma^* = (\sigma_i^*, \ldots, \sigma_n^*)$ is said to be a *Nash equilibrium* if and only if each player's strategy σ_i^* is the best response to the strategies of the other players, i.e., it satisfies

$$u_i(\sigma^*) = \max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma^*_{-i}) = u_i(b(\sigma^*_{-i}), \sigma^*_{-i}) \qquad \forall i \in N.$$

It means that no player can improve their payoff by unilaterally deviating from their equilibrium strategy. Given a positive number $\epsilon > 0$, a strategy profile σ is said to be an ϵ-Nash equilibrium if no player can gain more than ϵ in expected payoff by unilaterally deviating from their strategy, i.e., $\forall i \in N, u_i(\sigma) \geq \max_{\sigma'_i \in \Sigma_i} u(\sigma'_i, \sigma_{-i}) - \epsilon$. This inequality can also be expressed as $u_i(\sigma) \geq u_i(b(\sigma_{-i}), \sigma_{-i}) - \epsilon$, where $b(\sigma_{-i})$ denotes the best response to σ_{-i}. Similarly, the measure of player i's incentive to deviate from a strategy profile σ can be defined as $\delta_i(\sigma) = u_i(b(\sigma_{-i}), \sigma_{-i}) - u_i(\sigma)$. Based on these definitions, one metric to measure the gap between the strategy profile σ and the NE strategy is $\textsc{NashConv}(\sigma) = \sum_{i \in N} \delta_i(\sigma)$, and $\textsc{Exploitability}(\sigma) = \textsc{NashConv}(\sigma)/n$. These two metrics measure how close a strategy profile is to an NE – the closer they are to zero, the closer the strategy profile is to the NE strategy. When they are equal to zero, the strategy σ is an actual NE strategy. In particular, for zero-sum games, $\textsc{NashConv}$ simplifies to $\sum_{i \in N} u_i(b(\sigma_{-i}), \sigma_{-i})$, since the second term sums to zero when summing over all players.

Apart from the Nash equilibrium concept, there are also other equilibrium concepts. Particularly, for n-player general-sum games, (Coarse) Correlated Equilibrium ((C)CE) is also a common solution concept. Similar to the NE, a CE is a joint mixed strategy in which no player has the incentive to deviate [28]. Formally, let S be the joint strategy space and S_i be the joint strategy space of player i. The

strategy profile σ^* forms a CCE if it satisfies for $\forall i \in N, s_i \in S_i, u_i(\sigma^*) \geq u_i(s_i, \sigma^*_{-i})$ where σ^*_{-i} is the marginal distribution of σ^* on strategy space S_{-i}. Analogous to NE, the (C)CE Gap Sum is adopted to measure the distance from the (C)CE [29]. In this thesis, we focus on computing the NE strategy for two-player zero-sum imperfect-information extensive-form games and the CCE strategy for multi-player imperfect-information extensive-form games.

2.2 Equilibrium Finding Algorithms

The contemporary state-of-the-art algorithms for solving imperfect-information extensive-form games can be roughly divided into two groups: no-regret methods derived from Counterfactual Regret Minimization, and incremental strategy-space generation methods of the Policy Space Response Oracle framework. Both groups have their advantages and disadvantages. In the following subsections, we will introduce these two groups of algorithms.

2.2.1 Counterfactual Regret Minimization

Counterfactual regret minimization (CFR) is a family of iterative algorithms that are the most popular and fastest approaches to approximately solving large-scale imperfect-information extensive-form games [12]. First, we will introduce the vanilla CFR algorithm. Then we will discuss the various CFR variants.

To facilitate the description of the CFR algorithm, we first introduce some notations. Let $u_i(\sigma, h)$ be the expected utility of player i given that the history h is reached and all players play using strategy σ from that point on. Similar to the definition of the expected utility of a player given a strategy profile, $u_i(\sigma, h)$ can be defined as

$$u_i(\sigma, h) = \sum_{z \in Z} \pi^\sigma(h, z) u_i(z),$$

where $\pi^\sigma(h, z)$ refers to the probability of going from history h to terminal history z following strategy profile σ. Let $u_i(\sigma, h \cdot a)$ be the expected utility of player i given that the history h is reached and all players play using strategy σ except that

player i selects action a in the history h. Formally,

$$u_i(\sigma, h \cdot a) = \sum_{z \in Z} \pi^\sigma(h \cdot a, z) u_i(z).$$

Similarly, we use $u_i(\sigma, I)$ to represent the expected utility of player i given that the information set I is reached and all players play using strategy σ from that point on. This value is the weighted average of the value of each history in the information set. The weight is proportional to player i's belief that they are in that specific history, given that they know they are in the information set I. Formally, it can be defined as

$$u_i(\sigma, I) = \sum_{h \in I} \pi^\sigma(h|I) u_i(\sigma, h), \qquad \text{where} \quad \pi^\sigma(h|I) = \frac{\pi^\sigma(h)}{\pi^\sigma(I)}.$$

Then we define the *counterfactual utility* $v_i(\sigma, I)$, which is the expected value of information set I given that player i tries to reach it. Given that player i tries to reach the information set, the contribution of player i to the reaching probability of the information set can be viewed as 1, i.e., $\pi_i(I) = 1$ and $\pi_i(h) = 1$. Formally,

$$v_i(\sigma, I) = \sum_{h \in I} \pi^\sigma_{-i}(h|I) u_i(\sigma, h) \qquad \text{where} \quad \pi^\sigma_{-i}(h|I) = \frac{\pi^\sigma_{-i}(h)}{\pi^\sigma_{-i}(I)}.$$

For any available action for information set I, $a \in A(I)$, we define $\sigma|_{I \to a}$ to be the strategy profile identical to σ except that player i selects the action a when in the information set I. The counterfactual utility $v_i(\sigma|_{I \to a}, I)$ can be defined as follows,

$$v_i(\sigma|_{I \to a}, I) = \sum_{h \in I} \pi^\sigma_{-i}(h|I) u_i(\sigma, h \cdot a).$$

Based on the definition of counterfactual values, we can define the *instantaneous counterfactual regret* for action a in the information set I to be the player i's regret in its decisions at the information set I in terms of counterfactual utility with an additional weighting term for the counterfactual probability that I would be

reached if the player i had tried to do so, i.e., $\pi^{\sigma}_{-i}(I)$. Formally,

$$\begin{aligned} r_i(I,a) &= \pi^{\sigma}_{-i}(I)(v_i(\sigma|_{I\to a}, I) - v_i(\sigma, I)) \\ &= \sum_{h\in I} \pi^{\sigma}_{-i}(I)\pi^{\sigma}_{-i}(h|I)(u_i(\sigma, h\cdot a) - u_i(\sigma, h)) \\ &= \sum_{h\in I} \pi^{\sigma}_{-i}(h)(u_i(\sigma, h\cdot a) - u_i(\sigma, h)) \end{aligned}$$

Next, we consider repeatedly playing the game since the CFR algorithm is an iterative algorithm. Let σ_i^t be the strategy used by player i in round t. The cumulative counterfactual regret on iteration T is defined as follows, $R_i^T(I,a) = \sum_{t=1}^{T} r_i^t(I,a)$. Let $R_i^{T,+}(I,a) = \max(R_i^T(I,a), 0)$ be the positive portion of regret because we often are most concerned about regret when it is positive.

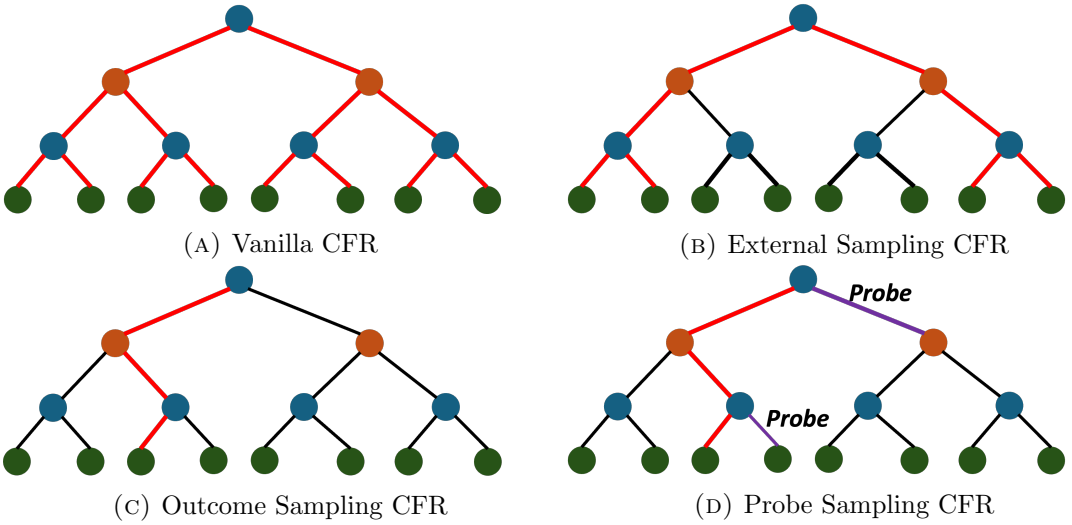
The vanilla CFR algorithm determines the strategy for each iteration by applying any of several *regret minimization* algorithms to each information set. Typically, *Regret Matching (RM)* is used as the regret minimization algorithm within the vanilla CFR algorithm due to the simplicity and lack of parameters of RM [30]. Specifically, the RM algorithm picks a distribution over actions in an information set in proportion to the positive regret of those actions. Formally, on iteration $T+1$, player i selects actions $a \in A(I)$ according to probabilities

$$\sigma^{T+1}(I,a) = \begin{cases} \frac{R_i^{T,+}(I,a)}{\sum_{b\in A(I)} R_i^{T,+}(I,b)} & \text{if } \sum_{b\in A(I)} R_i^{T,+}(I,b) > 0 \\ \frac{1}{|A(I)|} & \text{otherwise} \end{cases}$$

Before deep insight into the convergence of the CFR algorithm, we first provide a famous theorem that connects regret and the Nash equilibrium solution concept. Consider repeatedly playing an extensive-form game. Then the *average overall regret* of player i at time T is

$$R_i^T = \frac{1}{T} \max_{\sigma_i'\in\Sigma_i} \sum_{i=1}^{T} (u_i(\sigma_i', \sigma_{-i}^t) - u_i(\sigma^t))$$

Define $\overline{\sigma}_i^T$ to be the *average strategy* for player i from iteration 1 to T. The weighting item for the strategy of each iteration corresponds to the reaching probability for the information set using the strategy. In particular, for each information set



(A) Vanilla CFR (B) External Sampling CFR

(C) Outcome Sampling CFR (D) Probe Sampling CFR

FIGURE 2.3: The comparison between vanilla CFR and different sampling CFR.

$I \in \mathcal{I}_i$, for each action $a \in A(I)$, define $\overline{\sigma}_i^T = \sum_{t=1}^{T} \pi_i^{\sigma^t}(I)\sigma^t(I,a) / \sum_{t=1}^{T} \pi_i^{\sigma^t}(I)$. Then we have the following theorem about the convergence of average strategy in two-player zero-sum games.

Theorem 2.1. *In a two-player zero-sum game, at iteration T, if both player's average overall regret is less than ϵ, then average strategy $\overline{\sigma}^T$ is a 2ϵ equilibrium.*

Based on this theorem, we can analyze the convergence property of the CFR algorithm. The average cumulative counterfactual regret of the information set I in iteration T is $R_i^T(I) = \frac{1}{T} \max_{a \in A(I)} R_i^{T,+}(I,a)$. Then we have the following theorem.

Theorem 2.2 (Theorem 4[12])**.** *If player i selects actions according to regret matching algorithm, then $R_i^T(I) \leq \Delta_i \sqrt{|A_i|}/\sqrt{T}$ and consequently the average overall regret of player i, $R_i^T \leq \sum_{I \in \mathcal{I}_i} R^T(I) \leq \Delta_i |\mathcal{I}_i| \sqrt{|A_i|}/\sqrt{T}$ where $\Delta_i = \max_{z \in Z} u_i(z) - \max_{z \in Z} u_i(z)$ and $|A_i| = \max_{h:P(h)=i} |A(h)|$.*

This result establishes that the strategy in the regret matching algorithm can be used in self-play to compute an NE strategy. In addition, we can find that the bound on the average overall regret is *linear* in the number of information sets. Finally, as $T \to \infty$, $R_i^T \to 0$. Then, according to Theorem 2.1, in two-player zero-sum games, if both players' average regret $R_i^T \leq \epsilon$, then their average strategies $< \overline{\sigma}_1^T, \overline{\sigma}_2^T >$ form a 2ϵ-equilibrium.

CFR variants. The CFR algorithm has successfully been applied to solving imperfect-information extensive-form games [14] and also has a strong convergence

guarantee. However, the CFR algorithm must traverse the full game tree at every iteration, which is very time-consuming when the game tree is large. Therefore, some sampling-based CFR algorithms are proposed to effectively solve large-scale extensive-form games by avoiding traversing the full game tree, such as external sampling [20], outcome sampling [20], probe sampling [31] and other reduce variance sampling algorithms [32, 33]. Figure 2.3 shows the comparison of vanilla CFR and different sampling CFR algorithms in traversing the game tree.

The external sampling CFR algorithm has the lowest variance. However, it runs slowly because it traverses all actions of the traverser in every iteration, as shown in Figure 2.3. The outcome sampling CFR algorithm runs faster than the external sampling CFR algorithm since it only traverses one path from the root to the leaf node in every iteration. However, the outcome sampling CFR algorithm has a large variance, which influences the convergence rate. Compared to the outcome sampling CFR algorithm, the probe sampling CFR algorithm attempts to reduce variance by replacing "zeroed-out" counterfactual values of non-sampled actions with closer estimates of the true counterfactual values obtained from probing.

However, vanilla CFR and its sampling-based variants all use the tabular form to record the counterfactual regret values, which impedes their application in large-scale imperfect-information games due to limited memory. Nowadays, with the development of deep learning techniques, the neural network function approximation technique is applied to CFR to solve larger extensive-form games. For example, Deep CFR [21] is proposed to record the regret value and average strategy by training deep neural networks. Single Deep CFR [34] and Double Neural CFR [22] are also algorithms aiming to approximate CFR using deep neural networks.

2.2.2 Policy Space Response Oracle

Policy Space Response Oracle (PSRO) algorithm [13] is a generalization of the double oracle method [10], but instead of actions, PSRO uses policies as choices in the meta-game. PSRO has demonstrated strong performance in solving large-scale two-player zero-sum imperfect-information extensive-form games [35]. Here, we first introduce the PSRO algorithm and then discuss some PSRO variants.

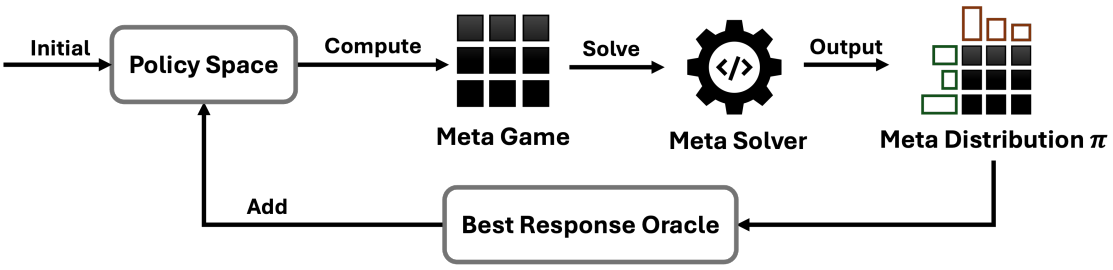


FIGURE 2.4: The overall process of policy space response oracle algorithm.

Figure 2.4 and Algorithm 1 outline the process of PSRO. At first, every player is initialized with a set of randomly generated strategies, i.e., S_i for each player i. Then for each strategy profile, $(\sigma_1, \ldots, \sigma_n) \in S_1 \times \ldots \times S_n$, we compute its expected utility to form a meta-game. Based on the meta-game, a meta-strategy or meta-distribution is initialized using different methods, such as taking the uniform strategy as the meta-strategy. Then, at each iteration of the PSRO algorithm, for every player, the best response oracle computes at least one strategy against other players' previous strategies over their meta-strategies. After computing the best response strategy (σ_i^*), the strategy is added to the policy space of the player, i.e., $S_i = S_i \cup \sigma_i^*$. Based on the new policy space, we can compute the missing entries in the meta-game. Then a meta-solver is used to compute the meta-strategy, i.e., a distribution over policies of each player (e.g., Nash equilibrium [2], projected replicator dynamics (PRD) [36], or uniform distributions). Finally, we repeat the above processes until convergence and the policy space and meta distribution are taken as output. Note that when computing the best response oracle, all other

Algorithm 1: Policy Space Response Oracle Framework [13]

Initialize policy space for each player $S_1, S_2, \ldots, S_n$;
Compute utilities for each strategy profile $(\sigma_1, \ldots, \sigma_n) \in S_1 \times \ldots \times S_n$;
Initialize meta distribution ρ;
while *iteration* t *in* $\{1, 2, \ldots, T\}$ **do**
    for *player* $i \in N$ **do**
        for *many episodes* **do**
            Sample $\sigma_{-i} \sim S_{-i}$ based on meta distribution ρ_{-i};
            Train best response oracle σ_i^* against σ_{-i};
        $S_i = S_i \bigcup \{\sigma_i^*\}$;
    Compute missing entries in mete-game based on $S_1 \times \ldots \times S_n$;
    Compute meta distribution ρ based on the meta-game using meta solver;
return *Policy space* $(S_1, \ldots, S_n)$ *and meta distribution* ρ

players' policies and the meta-strategy are fixed, which corresponds to a single-player optimization problem and can be considered as a reinforcement learning (RL) problem. Therefore, it can be solved by any RL algorithm, such as Deep Q-Network (DQN) [37] or policy gradient reinforcement learning algorithms [38].

PSRO is a general algorithm that could cover several types of algorithms. For example, iterated best response is an instance of PSRO with meta-distribution $\rho_{-i} = (0, 0, \ldots, 1, 0)$ and independent RL is also an instance of PSRO with meta-distribution $\rho_{-i} = (0, 0, \ldots, 0, 1)$. Specifically, fictitious play-based algorithms [39, 40], such as Neural Fictitious Self-Play (NFSP) [41], can also be seen as a special case of PSRO with uniform distributions as meta-distributions. Double Oracle is an instance of PSRO with player number $N = 2$ and meta-distribution set to a Nash equilibrium strategy profile of the meta-game.

PSRO variants. Although PSRO has demonstrated strong performance, its scalability remains an obstacle, impeding its application in solving large-scale extensive-form games. To make PSRO more effective in solving large-scale games, Pipeline PSRO (P2SRO) [42] is proposed by parallelizing PSRO with convergence guarantees. Extensive-Form Double Oracle (XDO) [43] is a version of PSRO where the restricted game allows mixing population strategies not only at the root of the game but every information set. It can guarantee to converge to an approximate NE in a number of iterations that are linear in the number of information sets, while PSRO may require a number of iterations exponential in the number of information sets. Neural XDO (NXDO) as a neural version of XDO learns approximate best response strategies through any deep reinforcement learning algorithm. Recently, Anytime Double Oracle (ADO) [44], a tabular double oracle algorithm for 2-player zero-sum games is proposed to converge to a Nash equilibrium while decreasing exploitability from one iteration to the next. Anytime PSRO (APSRO) [44] as a version of ADO calculates best responses via reinforcement learning algorithms. Except for NEs, other equilibrium solution concepts, for example, (Coarse) Correlated equilibrium ((C)CE) can also be computed under the PSRO framework. Joint Policy Space Response Oracles (JPSRO) [29] is proposed for training agents in n-player, general-sum extensive-form games, which provably converges to (C)CEs.

2.3 Chapter Summary

In this chapter, we introduced the definition of imperfect-information extensive-form games and discussed two typical classes of algorithms for solving these games. Although these algorithms have achieved significant accomplishments, they still suffer from scalability and generalizability issues. To address these challenges, the following chapters propose two scalable algorithms based on the CFR and PSRO frameworks, respectively, and one generalizable algorithm to solve large-scale pursuit-evasion games, a special type of imperfect-information extensive-form game. Finally, we provide an offline algorithm for solving imperfect-information extensive-form games using offline datasets.

Part I

Scalable Algorithms for IIEFGs

Chapter 3

Solving Combinatorial Action Space Problem via CFR-MIX

3.1 Introduction

In this Chapter[1], we focus on a special type of two-player imperfect-information extensive-form zero-sum games, known as *Team-Adversary Games*. In these games, a team of cooperative agents plays against an adversary, with all agents in the team sharing a single utility function. This game model captures many real-world scenarios, such as many policemen coordinating to catch an attacker [46]. The primary challenge in solving large-scale team-adversary games is the *exponential combinatorial action space problem*: the team's joint action is the combination of each agent's action, which scales exponentially with the number of agents. Figure 3.1 illustrates this problem. Suppose there are eight agents in the team and each agent has four actions (up, down, left, and right), then the size of the team's joint action space would be 4^8, which is exponential with the number of agents.

Counterfactual regret minimization (CFR) [12] is one of the most popular algorithms to solve imperfect-information extensive-form games. It is an iterative algorithm that approximates a Nash equilibrium through repeated self-play between two regret-minimizing algorithms. To improve scalability, several sampling-based CFR variants [20, 31] have been proposed to solve large-scale imperfect-information

[1]This Chapter has been published in [45].

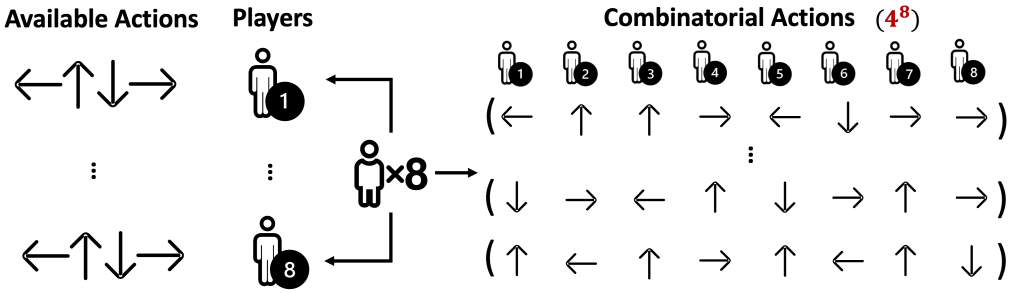


FIGURE 3.1: The combinatorial action space problem

extensive-form games effectively. Nowadays, neural network function approximation is applied to CFR to solve larger games. Deep CFR [21], Single Deep CFR [34], and Double Neural CFR [22] are algorithms that aim to approximate CFR using deep neural networks. However, these algorithms struggle to solve large-scale team-adversary games due to the following reasons.

First, vanilla CFR and its sampling-based variants use tabular forms to record the regret values and average strategies of joint actions. When applied to team-adversary games, these methods quickly become impractical due to limited memory capacity. Second, these deep learning-based CFR variants also cannot solve team-adversary games effectively since training the strategy network over such a large action space is inefficient. Lastly, one way to avoid computing the joint action strategy is to let each agent compute its policy independently. However, this approach fails to promote the cooperative interaction between agents in the team and has no theoretical guarantee of convergence to equilibrium in multi-player games [47]. Therefore, it is impractical to solve this type of imperfect-information extensive-form games with existing algorithms.

To solve the *exponential combinatorial action space problems* in team-adversary games, we propose a new CFR-based framework: ***CFR-MIX***. Our contributions are summarized as follows: i) We propose a novel strategy representation that represents the team's joint action strategy using the individual strategy of each agent. ii) We define a consistency relationship between the novel strategy representation and the traditional strategy representation to ensure that the equilibrium with the novel strategy representation remains unchanged. iii) To compute NE with the novel strategy representation under the CFR framework, we transform the consistency relationship between strategy representations to the consistency relationship between the cumulative regret values of joint actions and the cumulative regret

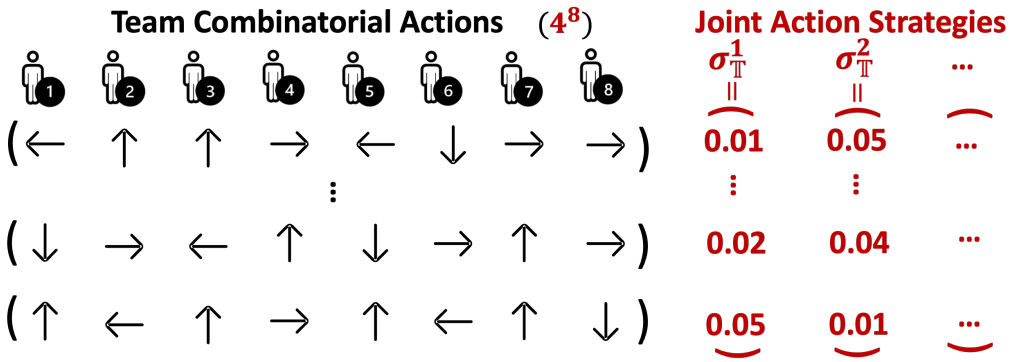


FIGURE 3.2: The strategy representation over joint action space.

values of individual actions. iv) We propose a novel decomposition method over cumulative regret values to guarantee the consistency relationship between cumulative regret values. v) We employ a mixing layer that estimates the cumulative regret values of joint actions as a nonlinear combination of cumulative regret values of individual actions to implement the decomposition method. vi) The parameter sharing technique is applied among all agents in the team, which improves performance and significantly reduces the network parameters. Finally, experimental results show that CFR-MIX outperforms state-of-the-art CFR-based algorithms.

3.2 Team-Adversary Game

In this section, we introduce the definition of the *Team-Adversary Game*, a special type of imperfect-information extensive-form game. Since we have already provided the formal definition of imperfect-information extensive-form games in Chapter 2, here, we will focus on the unique features of the team-adversary game. In this type of game, there are two players: an adversary $\mathbb{V}$ and a team $\mathbb{T}$. The team consists of multiple agents, $\mathbb{T} = \{1, \ldots, n\}$. Every agent in team $i \in \mathbb{T}$ has its own set of available actions, denoted as $A_{\mathbb{T}_i}$. Therefore, the team's available action set comprises all possible combinations of the individual agents' actions, i.e., $A_{\mathbb{T}} = \times_{i \in R} A_{\mathbb{T}_i}$. Here, we focus on zero-sum games, which means that the utility functions of the team and the adversary are perfectly opposing. Formally, this is expressed as $u_{\mathbb{T}}(z) = -u_{\mathbb{V}}(z)$ for all terminal histories $z \in Z$.

As we introduced in Chapter 2, a player's behavior strategy σ_i is a function that maps every information set of player i to a probability distribution over $A(I_i)$.

In particular, the strategy of the team is defined over the joint action space, as illustrated in Figure 3.2. In this chapter, we adopt Nash equilibrium (NE) [2] as our solution concept for team-adversary games. An NE is a strategy profile where no player can gain more reward by unilaterally switching to a different strategy. Formally, in team-adversary games, an NE is a strategy profile σ^* such that

$$u_i(\sigma^*) = \max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma^*_{-i}) \quad \forall i \in \{\mathbb{V}, \mathbb{T}\}$$

An approximation of an NE, or ϵ-NE is a strategy profile σ^* where

$$u_i(\sigma^*) + \epsilon \geq \max_{\sigma'_i \in \Sigma_i} u_i(\sigma'_i, \sigma^*_{-i}) \quad \forall i \in \{\mathbb{V}, \mathbb{T}\}$$

3.3 Novel Strategy Representation

In this Chapter, we focus on solving team-adversary games where a team of agents plays against an adversary, with all agents sharing a single utility function. It is important to note that the team's joint action space grows exponentially with the number of team players. This makes it impractical to solve this type of game using existing approaches due to the large joint action space of the team. To mitigate the exponentially combinatorial action space problem, we propose a novel strategy representation for the team, which could significantly reduce the strategy space over the joint action space.

3.3.1 Individual Strategy Representation

Here, we introduce a novel strategy representation to denote the team's joint action strategy using the individual strategies of all the team players over their own action spaces. Unlike the traditional joint action strategy representation, the size of the individual strategy spaces of all the team agents over their own action spaces is linear with the number of agents. It is worth noting that all agents share the same history of the team $\mathbb{T}$, i.e., the player function can be extended as $P(h) = \mathbb{T} = \{1, \ldots, n\}$. In other words, all the team agents share the same information sets of the team $\mathbb{T}$. Therefore, the novel strategy representation of the team is defined as $f_{\mathbb{T}} = (\sigma_1, \sigma_2, \ldots, \sigma_n)$, where σ_i is agent i's strategy which maps each information

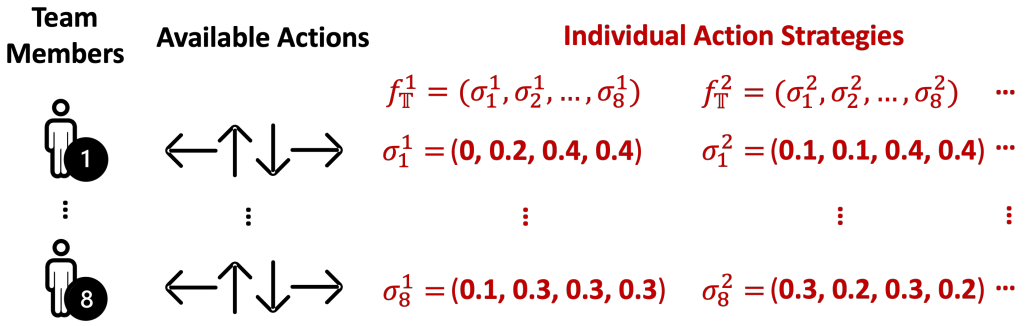


FIGURE 3.3: The strategy representation over individual action space.

set of player $\mathbb{T}$ to a probability distribution over agent i's action space $A_{\mathbb{T}_i}$, as shown in Figure 3.3. For clarity, we refer to this proposed novel representation as the *individual strategy representation* and denote the traditional joint action strategy representation as the *joint strategy representation*. Given a strategy profile $\sigma = (\sigma_{\mathbb{V}}, f_{\mathbb{T}}) = (\sigma_{\mathbb{V}}, \sigma_1, \ldots, \sigma_n)$, the expected payoff of each agent is the same, $u_i(\sigma) = \sum_{z \in Z} u_{\mathbb{T}}(z)\pi^{\sigma}(z)$, where $\pi^{\sigma}(z)$ is the reaching probability of terminal history z if all players choose actions according to strategy profile σ. This means that each agent i chooses its action according to its strategy σ_i independently, and the adversary makes its decision following its strategy $\sigma_{\mathbb{V}}$.

3.3.2 Strategy Consistency Relationship

The individual strategy representation can significantly reduce the search space since it grows linear with the number of agents. However, it is important to note that it is not capable of fully representing the team's entire joint action strategy space. To clarify the relationship between the individual strategy representation and traditional joint strategy representation, we propose a *consistency relationship* which can be defined as follows,

$$\boldsymbol{\sigma}_{\mathbb{T}}(I, \boldsymbol{a}) = \sigma_1(I, a_1) \times \sigma_2(I, a_2) \times \ldots \times \sigma_n(I, a_n) = \prod_{i=1}^{n} \sigma_i(I, a_i), \tag{3.1}$$

where $\boldsymbol{a} = (a_1, \ldots, a_n)$, $\boldsymbol{\sigma}_{\mathbb{T}}(\cdot)$ denotes the team's strategy over joint action space and $\sigma_i(\cdot)$ denotes the agent i's strategy over its own action space. In short, the joint strategy representation can be decomposed into the individual strategy representation following the consistency relationship equation. When given a team's strategy $\boldsymbol{\sigma}_{\mathbb{T}}(I, \cdot)$ over the joint action space for an information set I, we can obtain

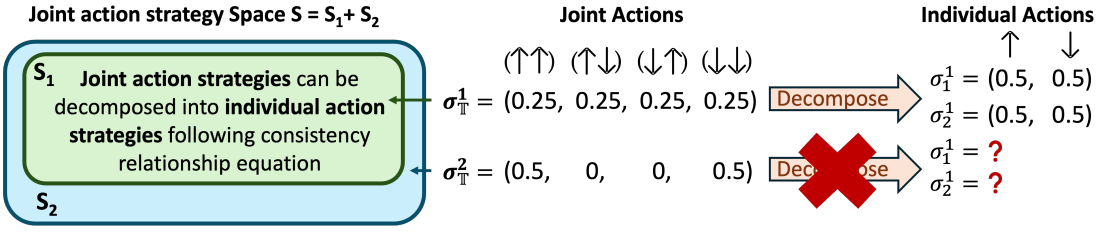


FIGURE 3.4: The relationship between novel and joint strategy representations.

the individual strategy for each agent $\sigma_i(I, \cdot)$ by solving the following nonlinear equation set,

$$\begin{cases} \boldsymbol{\sigma}_{\mathbb{T}}(I, \boldsymbol{a}_1) = \sigma_1(I, a_{11}) \times \sigma_2(I, a_{21}) \times \ldots \times \sigma_n(I, a_{n1}) = \prod_{i=1}^{n} \sigma_i(I, a_{i1}) \\ \qquad\qquad \ldots \quad \ldots \\ \boldsymbol{\sigma}_{\mathbb{T}}(I, \boldsymbol{a}_L) = \sigma_1(I, a_{1L}) \times \sigma_2(I, a_{2L}) \times \ldots \times \sigma_n(I, a_{nL}) = \prod_{i=1}^{n} \sigma_i(I, a_{iL}) \\ \forall \boldsymbol{a}_j \in A_{\mathbb{T}}(I) \quad \boldsymbol{a}_j = (a_{1j}, ..., a_{nj}) \end{cases}$$

where L is the number of the team's joint actions, i.e., $L = |A_{\mathbb{T}}(I)|$. Clearly, this equation set does not always have a unique solution, which means that not every team's strategy over joint action space can be decomposed into individual strategies of all agents following the strategy consistency.

According to the consistency relationship, the entire joint action space S of the team can be divided into two parts: S_1, where strategies satisfy the consistency relationship, and S_2, where strategies do not satisfy the consistency relationship, as shown in Figure 3.4. To better understand this division, we provide an example of a strategy in S_2. Consider a team with two agents, where the action set for each agent is $\{a_1, a_2\}$ and $\{b_1, b_2\}$, respectively. Given a joint action strategy $\boldsymbol{\sigma} = (0.5, 0, 0, 0.5)$ over the joint action set $\{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2)\}$. Obviously, for this joint action strategy, we cannot obtain the corresponding individual strategy representation satisfying the consistency relationship (Equation (3.1)).

Based on Figure 3.4, the strategy space for a two-player team-adversary game can be represented by the tuple $(S, S_{\mathbb{V}})$, where $S_{\mathbb{V}}$ denotes the strategy space of the adversary. When using the individual strategy representation, the team's strategy space is S_1. In this case, the tuple $(S_1, S_{\mathbb{V}})$ also forms the strategy space for a two-player team-adversary game with the consistency relationship satisfied. Depending

TABLE 3.1: Utility Matrix

Action	(a_1, b_1)	(a_1, b_2)	(a_2, b_1)	(a_2, b_2)
c_1	1, -1	1, -1	0, 0	0, 0
c_2	-1, 1	-1, 1	-2, 2	-2, 2

on the location of the team's NE strategy in a two-player team-adversary game with the strategy space $(S, S_{\mathbb{V}})$, there can be two different outcomes.

1. If the team's NE strategy for the game with the strategy space (S, S_V) is located in the strategy space S_1, then the team's NE strategy of the game with the strategy space $(S_1, S_{\mathbb{V}})$ equals the team's NE strategy of the game with the strategy space $(S, S_{\mathbb{V}})$. In this chapter, we focus on the type of games where the team's strategy can be decomposed as defined by Equation 3.1. Theorem 3.1 formally describes this outcome. To provide a clearer understanding, we present a toy example. Consider a team with two agents, where the action set for each agent is $\{a_1, a_2\}$ and $\{b_1, b_2\}$, respectively, and the action set for the adversary is $\{c_1, c_2\}$. Here, we consider a one-step extensive-form game whose utility can be represented by a matrix, shown in Table 3.1. We can find the pure NE strategy of the game to be $((a_2, b_1), c_1)$ or $((a_2, b_2), c_1)$. The pure joint action strategy can be easily decomposed into individual strategies for each agent. Accordingly, the mixed NE strategy for the team would be a joint action strategy $\sigma = (0, 0, p, 1-p)$ over the joint action set $\{(a_1, b_1), (a_1, b_2), (a_2, b_1), (a_2, b_2)\}$, where $p \in [0, 1]$. According to the strategy consistency, we can decompose this mixed strategy of the team into individual strategies for each agent, i.e., $\sigma_1 = (0, 1)$ over the action set $\{a_1, a_2\}$ and $\sigma_2 = (p, 1-p)$ over the action set $\{b_1, b_2\}$.

Theorem 3.1. *Under the individual strategy representation, the Nash Equilibrium strategy profile between the adversary and the team remains unchanged if Equation (3.1) holds for the team's Nash equilibrium strategy.*

Proof. Let $\sigma = (\sigma_{\mathbb{V}}, \boldsymbol{\sigma}_{\mathbb{T}})$ be the Nash equilibrium strategy profile and $\sigma_i(I, a_i)$ be the strategy of agent i. We can know that

$$u_{\mathbb{V}}(\sigma_{\mathbb{V}}, \boldsymbol{\sigma}_{\mathbb{T}}) = \max_{\sigma'_{\mathbb{V}} \in \sum_{\mathbb{V}}} u_{\mathbb{V}}(\sigma'_{\mathbb{V}}, \boldsymbol{\sigma}_{\mathbb{T}}), \qquad u_{\mathbb{T}}(\sigma_{\mathbb{V}}, \boldsymbol{\sigma}_{\mathbb{T}}) = \max_{\boldsymbol{\sigma}'_{\mathbb{T}} \in \sum_{\mathbb{T}}} u_{\mathbb{V}}(\sigma_{\mathbb{V}}, \boldsymbol{\sigma}'_{\mathbb{T}}).$$

From the consistency relationship between the individual action strategy and the joint action strategy, we can know that the team's equilibrium strategy satisfies the consistency relationship. Therefore, $\boldsymbol{\sigma}_{\mathbb{T}} = \sigma_1 \times \ldots \times \sigma_n$. Then, we have

$$u_{\mathbb{V}}(\sigma_{\mathbb{V}}, \boldsymbol{\sigma}_{\mathbb{T}}) = \max_{\sigma'_{\mathbb{V}} \in \sum_{\mathbb{V}}} u_{\mathbb{V}}(\sigma'_{\mathbb{V}}, \sigma_1 \times \ldots \times \sigma_n) = \max_{\sigma'_{\mathbb{V}} \in \sum_{\mathbb{V}}} u_{\mathbb{V}}(\sigma'_{\mathbb{V}}, \sigma_1, \ldots, \sigma_n),$$

From above, we know that $\sigma_{\mathbb{V}}$ is still the best response strategy against individual strategies and $\boldsymbol{\sigma}_{\mathbb{T}} = \sigma_1 \times \ldots \times \sigma_n$ is also the best response strategy against $\sigma_{\mathbb{V}}$. Therefore, $\sigma = (\sigma_{\mathbb{V}}, \sigma_1, \ldots, \sigma_n)$ is also an NE strategy profile and $\sigma = (\sigma_{\mathbb{V}}, \boldsymbol{\sigma}_{\mathbb{T}})$ is the same as $\sigma = (\sigma_{\mathbb{V}}, \sigma_1, \ldots, \sigma_n)$. In summary, if the team's NE strategy satisfies the consistency relationship, the NE strategy remains unchanged under the individual strategy representation. □

2. If the team's NE strategy for the game with strategy space $(S, S_{\mathbb{V}})$ is located within the S_2 space, then the team's NE strategy for the game with strategy space $(S_1, S_{\mathbb{V}})$ will differ from that in the game with strategy space $(S, S_{\mathbb{V}})$. However, it will be a special type of Team-Maxmin Equilibrium (TME) [48] strategy that adheres to the consistency relationship defined for the TME strategy. In this scenario, the team may incur a utility loss compared to their NE reward in the game with the $(S, S_{\mathbb{V}})$ strategy space. This is because, in the game with strategy space $(S, S_{\mathbb{V}})$, agents can be considered to be able to always communicate and coordinate their actions, whereas, in the game with strategy space $(S_1, S_{\mathbb{V}})$, such communication and joint action are restricted due to strategy consistency requirements. Essentially, the utility of the team's joint strategy is not always lower than the utility of the team's strategy which can be decomposed. Although we do not provide a solid bound on the potential utility lost here, experimental results show that solving the game under the assumption that all joint strategies can be decomposed yields better strategies more quickly than solving the large game with the joint strategy space $(S, S_{\mathbb{V}})$. This could be because the exponential action space significantly influences the computation process of the joint action strategy. In the following, we do not focus on this case. Instead, we concentrate on the case where the team's strategy can be decomposed as described by Equation 3.1.

3.4 CFR-MIX

Now we move to introduce a novel algorithm, CFR-MIX, to compute an NE for the team-adversary game with strategy space $(S_1, S_{\mathbb{V}})$. Firstly, we establish the consistency relationship between cumulative regret values, which is derived from Equation (3.1). Secondly, we propose a novel decomposition method for cumulative regret values based on this consistency relationship. Then, we present our algorithm, CFR-MIX, where the decomposition method is implemented through a mixing layer. Finally, we provide the theoretical bound for our algorithm.

3.4.1 Regret Consistency Relationship

For imperfect-information extensive-form games, one of the most popular algorithms for computing NEs is the CFR-based algorithm, which leverages regret-matching to compute strategies based on cumulative regret values. In other words, the probability of each action is related to the cumulative regret value of that action. Therefore, to compute the individual strategy for each agent, we need to know the cumulative regret values of its actions explicitly. However, since all the agents share the same team utility function, we can only obtain the regret values of joint actions. To address this issue, we propose a decomposition method for cumulative regret values. Before describing the proposed decomposition method, we first propose a consistency relationship between accumulated regret values of the team's joint actions and those of the team member's actions.

This relationship is derived from the consistency relationship between strategy representations. Here, we present the derivation process. When $\sum_{\forall \boldsymbol{a}'} R_{tot}(I, \boldsymbol{a}')^+ > 0$ and $\sum_{b \in A_i(I)} R_i(I, b)^+ > 0$, the consistency relationship between regret values would be derived as follows,

$$\forall \boldsymbol{a} = (a_1, \ldots, a_n), \boldsymbol{\sigma}_{\mathbb{T}}(I, \boldsymbol{a}) = \sigma_1(I, a_1) \times \sigma_2(I, a_2) \times \ldots \times \sigma_n(I, a_n) = \prod_{i=1}^{n} \sigma_i(I, a_i) \tag{3.2}$$

$$\forall \boldsymbol{a} = (a_1, \ldots, a_n), \frac{R_{tot}(I, \boldsymbol{a})^+}{\sum_{\forall \boldsymbol{a}'} R_{tot}(I, \boldsymbol{a}')^+} = \prod_{i=1}^{n} \left(\frac{R_i(I, a_i)^+}{\sum_{b \in A_i(I)} R_i(I, b)^+} \right) \tag{3.3}$$

where $R_{tot}(\cdot)$ and $R_i(\cdot)$ represent the cumulative regret values of a team's joint action and the individual action of team member i, respectively. Note that Equation (3.2) is the strategy consistency relationship. In CFR-based algorithms, the strategies are computed using regret-matching based on cumulative regret values, which means that $\sigma(I,a) = \frac{R^+(I,a)}{\sum_{b\in A(I)} R^+(I,b)}$ when $\sum_{b\in A(I)} R^+(I,b) > 0$. Based on this, we can easily derive Equation (3.3) from Equation (3.2). If $\sum_{\forall \boldsymbol{a}'} R_{tot}(I,\boldsymbol{a}')^+ \leq 0$, then according to regret-matching, the strategy would be a uniform strategy. To ensure the strategy consistency relationship is maintained, the cumulative regret values of individual actions must satisfy $\sum_{b\in A_i(I)} R_i(I,b)^+ \leq 0$, $\forall i \in \{1,..n\}$. Thus, when the consistency relationship between cumulative regret values is upheld, the consistency relationship between strategy representations is also preserved.

3.4.2 Product-Form Decomposition

To guarantee these consistency relationships, we propose a product-form decomposition method for cumulative regret values, which can be defined as follows:

$$\forall \boldsymbol{a} = (a_1, \ldots, a_n), \quad R_{tot}(I,\boldsymbol{a}) = \prod_{i=1}^{n} R_i(I,a_i). \tag{3.4}$$

Here, we set $R_{tot} = 0$ if $R_{tot} \leq 0$, following the setting of regret-matching+ [49], which is a regret-minimizing algorithm that operates similarly to regret-matching. Unlike regret matching, which ignores actions with cumulative negative regret value, regret-matching+ actively resets any cumulative negative regret value back to zero. For the remainder of this section, we assume that all cumulative regret values are non-negative.

Theorem 3.2. *If product-form decomposition (Equation (3.4)) holds, the consistency relationship between the joint strategy representation and the individual strategy representation (Equation (3.1)) can be guaranteed.*

Proof. When $\sum_{\forall \boldsymbol{a}'} R_{tot}(I,\boldsymbol{a}') > 0$, we can get

$$\boldsymbol{\sigma}(I,\boldsymbol{a}) = \frac{R_{tot}(I,\boldsymbol{a})}{\sum_{\forall \boldsymbol{a}'} R_{tot}(I,\boldsymbol{a}')} = \frac{\prod_{i=1}^{n} R_i(I,a_i)}{\sum_{\forall \boldsymbol{a}'} R_{tot}(I,\boldsymbol{a}')}, \tag{3.5}$$

$$\prod_{i=1}^{n}\sigma(I,a_i)=\prod_{i=1}^{n}\frac{R_i(I,a_i)}{\sum\limits_{a_i^j\in A_i(I)}R_i(I,a_i^j)}$$
$$=\frac{\prod\limits_{i=1}^{n}R_i(I,a_i)}{\prod\limits_{i=1}^{n}\sum\limits_{a_i^j\in A_i(I)}R_i(I,a_i^j)}. \tag{3.6}$$

These above equations are derived by the regret matching+ and product-form decomposition equation.

$$\begin{aligned}
&\prod_{i=1}^{n}\sum_{a_i^j\in A_i(I)}R_i(I,a_i^j)\\
=&\prod_{i=1}^{n}[R_i(I,a_i^1)+R_i(I,a_i^2)+...+R_i(I,a_i^{|A_i(I)|})]\\
=&\prod_{i=1}^{n}R_i(I,a_i^1)+\prod_{i=1}^{n-1}R_i(I,a_i^1)R_n(I,a_n^2)+...+\prod_{i=1}^{n}R_i(I,a_i^{|A_i(I)|})\\
=&R_{tot}(I,\boldsymbol{a}_1)+...+R_{tot}(I,\boldsymbol{a}_{|A_1(I)|\times...\times|A_n(I)|})=\sum_{\forall\boldsymbol{a}'}R_{tot}(I,\boldsymbol{a}').
\end{aligned}$$

This derivation follows the rules of polynomial multiplication and product-form decomposition equation. From this equation and the above two equations (Equations (3.5-3.6)), we can find that $\boldsymbol{\sigma}(I,\boldsymbol{a})=\prod_{i=1}^{n}\sigma(I,a_i)$. Thus, Equation (3.1) holds. □

3.4.3 Mixing Layer

Here, we move to introduce our algorithm, CFR-MIX, which employs a mixing layer to implement the proposed product-form decomposition method under the deep learning-based CFR framework. Next, we will explain why we opted against implementing the product-form decomposition method within a tabular-based CFR framework, using a simple example to illustrate our reason. Assume that in a team-adversary game, there are two agents in the team, i.e., $\mathbb{T}=\{1,2\}$. At an information set $I_{\mathbb{T}}$, we suppose that agent 1 has two available actions denoted by $\{a_{11},a_{12}\}$ and agent 2 has two available actions denoted by $\{a_{21},a_{22}\}$. Under the CFR framework, we can only obtain the cumulative regret values of all joint actions $R_{tot}(I,\boldsymbol{a})$, where $\boldsymbol{a}\in\{(a_{11},a_{21}),(a_{11},a_{22}),(a_{12},a_{21}),(a_{12},a_{22})\}$. Then we compute

the accumulative regret value of each agent's actions by solving a nonlinear equation set which can be defined as follows,

$$\begin{cases} R_{tot}(I_{\mathbb{T}}, (a_{11}, a_{21})) = & R_1(I_{\mathbb{T}}, a_{11}) \times R_2(I_{\mathbb{T}}, a_{21}) \\ R_{tot}(I_{\mathbb{T}}, (a_{11}, a_{22})) = & R_1(I_{\mathbb{T}}, a_{11}) \times R_2(I_{\mathbb{T}}, a_{22}) \\ R_{tot}(I_{\mathbb{T}}, (a_{12}, a_{21})) = & R_1(I_{\mathbb{T}}, a_{12}) \times R_2(I_{\mathbb{T}}, a_{21}) \\ R_{tot}(I_{\mathbb{T}}, (a_{12}, a_{22})) = & R_1(I_{\mathbb{T}}, a_{12}) \times R_2(I_{\mathbb{T}}, a_{22}) \end{cases}$$

Solving this equation set becomes challenging when the game has a large number of team members, as it requires significant computational resources. Consequently, we opt to implement the product-form decomposition method within the deep learning-based CFR framework, leveraging the impressive function approximation properties of neural networks. In another scenario, if there is no solution for this nonlinear equation set, it implies that the regret value cannot be decomposed, and neither can the strategy. Although there is no theoretical guarantee in such a case, our neural network framework is still capable of providing approximate results, whereas these tabular-based methods cannot solve this case. To this end, we implement the product-form decomposition method via a mixing layer in a deep learning-based CFR framework. According to the product-form decomposition method defined in Equation (3.4), we design the mixing layer using the simple product operation, as shown in Figure 3.5.

Next, we use the Double Neural CFR framework as the base CFR framework to describe our CFR-MIX algorithm. It should be noted that our mixing layer can be applied to any deep learning-based CFR framework. In our CFR-MIX algorithm, we adopt the probe sampling algorithm as the traverse algorithm to traverse the game tree and compute regret values for traversed information sets. It is important to note that we employ the individual strategy representation in our CFR-MIX algorithm. Therefore, during the traversal process, when it turns the team to act, every agent in the team plays according to its own strategy and composes their individual strategies into the joint action strategy which is used to compute regret values only. Because all agents share one team utility function, it computes regret values for all joint actions based on the joint action strategy and joint action values. Finally, CFR-MIX trains one cumulative regret value network and one average strategy network for each player.

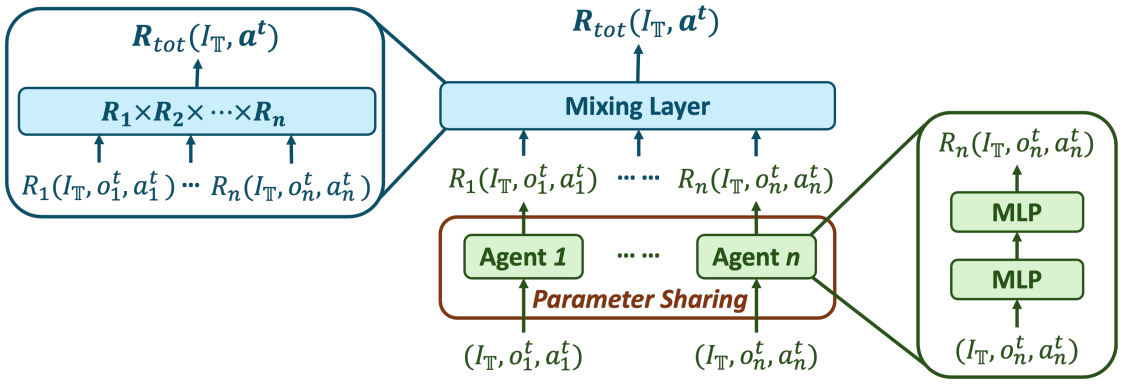


FIGURE 3.5: Architecture of cumulative regret neural network for team

The crux of our CFR-MIX algorithm lies in the cumulative regret neural network of the team, which implements the decomposition method via a mixing layer. Figure 3.5 shows the team's cumulative regret neural network architecture. In this structure, the mixing layer is engineered to perform a product operation on the accumulative regret values of all team members' actions, subsequently generating the accumulative regret values associated with the team's joint actions. The AGENT network serves as the regret network for the team members. It takes the information sets $I_{\mathbb{T}}$, agent i's observations, and agent i's available actions as inputs and outputs the cumulative regret value of agent i's actions. Then the mixing layer takes all the outputs of the AGENT network as inputs and outputs the cumulative regret value of the corresponding joint action. This constitutes the forward process.

We can obtain the accumulative regret value of the team's joint action via traversal processes. Then we adopt the Mean Squared Error (MSE) as the loss function. Finally, we can train the regret network using the Stochastic Gradient Descent (SGD) method. In this way, the mixing layer can guarantee the consistency relationship between cumulative regret values. According to Theorem 3.2, the mixing layer can also guarantee the consistency relationship between strategy representations.

To get the individual strategy of each agent, we only use the AGENT network to estimate the cumulative regret value for each agent's action and compute the agent's strategy using regret-matching+ based on the estimated cumulative regret value. Therefore, CFR-MIX only needs to traverse the action spaces of all the agents which are linear with the number of agents. On the contrary, to get the joint strategy, Double Neural CFR needs to traverse the whole joint action space which is exponential with the number of agents. Furthermore, in CFR-MIX, we adopt the

parameter sharing technique which means that all agents share the same AGENT network. Parameter sharing can reduce the parameters and improve performance significantly. For the average strategy network, we also only use one network for all agents. The training data of the strategy network are all agents' strategies. However, in the Double Neural CFR algorithm, the training data of the average strategy network are the team strategies defined over all the joint actions.

3.4.4 Convergence Analysis

Recall that we focus on a subset of joint action strategy space which can be decomposed into individual action strategies. Therefore, we compute the Nash equilibrium of the imperfect-information extensive-form game with strategy space (S_1, S_V) using the proposed CFR-MIX algorithm and provide a regret bound of our CFR-MIX algorithm under mild conditions based on Deep CFR [21].

Theorem 3.3. *Let T denote the number of CFR-MIX iterations, $|A|$ the maximum number of actions at any infoset, and K the number of traversals per iteration. Let $L^t_{\mathcal{R}}$ be the average MSE loss for $\mathcal{R}_p(I, a|\theta^t)$ on a sample in $M_{r,p}$ at iteration t, and let $L^t_{\mathcal{R}^*}$ be the minimum loss achievable for any function $\mathcal{R}$. Let $L^t_{\mathcal{R}} - L^t_{\mathcal{R}^*} \leq \epsilon_{\boldsymbol{L}}$. If the value memories are sufficiently large and **Equation (3.4) holds**, then with probability $1 - \rho$ total regret of player p at time T is bounded by*

$$R^T_p \leq (1 + \frac{\sqrt{2}}{\sqrt{\rho K}})\Delta|I_p|\sqrt{|A|}\sqrt{T} + 4T|I_p|\sqrt{|A|\Delta\epsilon_{\boldsymbol{L}}}$$

As $T \to \infty$, the average regret $\frac{R^T_p}{T}$ is bounded by $4|\mathcal{I}_p|\sqrt{|A|\Delta\epsilon_{\boldsymbol{L}}}$ with high probability.

Proof. We know that the establishment of Equation 3.4 guarantees the consistency relationship between strategies (Theorem 3.2). Therefore, under Theorem 3.1, we know that the NE strategy remains unchanged under the new strategy representation. Then, we can get that the regret bound for the joint strategy representation is the same as for the individual strategy representation. Another difference is that we use the probe sampling algorithm and Deep CFR uses the external sampling algorithm. Our proof follows the proof of Theorem 1 in [21] which provides the regret bound for the joint strategy representation.

Assume that an online learning scheme plays the strategy using the regret-matching method, defined as follows:

$$\sigma^t(I,a) = \begin{cases} \dfrac{y_t^+(I,a)}{\sum_a y_t^+(I,a)}, & \text{if} \quad \sum_a y_t^+(I,a) > 0 \\ \text{arbitrary value}, & \text{otherwise}, \end{cases} \tag{3.7}$$

Corollary 3.0.6 [50] provides the upper bound of the total regret by leveraging a function of the L2 distance between y_t^+ and $R^{T,+}$ on each information set. Therefore, we can get the following:

$$\begin{aligned} \max_{a\in A}(R^T(I,a))^2 &\leq |A|\Delta^2 T + 4\Delta|A| \sum_{t=1}^{T}\sum_{a\in A}\sqrt{(R_+^t(I,a) - y_+^t(I,a))^2} \\ &\leq |A|\Delta^2 T + 4\Delta|A| \sum_{t=1}^{T}\sum_{a\in A}\sqrt{(R^t(I,a) - y^t(I,a))^2} \end{aligned}$$

As shown in Equation (3.7), $\sigma^t(I,a)$ is invariant to rescaling across all actions at an information set, it's also the case that for any $C(I) > 0$,

$$\max_{a\in A}(R^T(I,a))^2 \leq |A|\Delta^2 T + 4\Delta|A| \sum_{t=1}^{T}\sum_{a\in A}\sqrt{(R^t(I,a) - C(I)y^t(I,a))^2}. \tag{3.8}$$

Let $x^t(I)$ be an indicator variable which is 1 if the information set I was traversed on iteration t. If the information set I was traversed then regret value $\tilde{r}^t(I)$ was stored in regret memory $M_{V,p}$, otherwise regret value $\tilde{r}^t(I) = 0$. Assume for now that regret memory $M_{V,p}$ is not full, so all sampled regrets are stored in the memory.

Let Π^t be the fraction of iterations on which $x^t(I) = 1$, and let

$$\epsilon^t(I) = ||E_t[\tilde{r}^t(I)|x^t(I) = 1] - V(I,a|\theta^t)||_2 \tag{3.9}$$

Let's insert the canceling factor of $\sum_{t'=1}^{t} x^{t'}(I)$ and set $C(I) = \sum_{t'=1}^{t} x^{t'}(I)$. Then we can get:

$$\max_{a\in A}(\tilde{R}^T(I,a))^2 \leq |A|\Delta^2 T + 4\Delta|A|\sum_{t=1}^{T}(\sum_{t'=1}^{t} x^{t'}(I))\sum_{a\in A}\sqrt{(\frac{\tilde{R}^t(I,a)}{\sum_{t'=1}^{t} x^{t'}(I)} - y^t(I,a))^2}$$

$$= |A|\Delta^2 T + 4\Delta|A|\sum_{t=1}^{T}(\sum_{t'=1}^{t} x^{t'}(I))||E_t[\tilde{r}^t(I)|x^t(I)=1] - V(I,a|\theta^t)||_2$$

$$= |A|\Delta^2 T + 4\Delta|A|\sum_{t=1}^{T} t\Pi^t(I)\epsilon^t(I)$$

$$\leq |A|\Delta^2 T + 4\Delta|A|T\sum_{t=1}^{T}\Pi^t(I)\epsilon^t(I)$$

The first term of this expression is the same as the regret bound of the tabular-form CFR algorithm, while the second term accounts for the approximation error. In Deep CFR [21], Theorem 3 shows the regret bound for K-external sampling, for the case of K-probe sampling, we can get the same results. Thus, we can get

$$\max_{a\in A}(\tilde{R}^T(I,a))^2 \leq |A|\Delta^2 TK^2 + 4\Delta|A|TK^2\sum_{t=1}^{T}\Pi^t(I)\epsilon^t(I)$$

for K-probe sampling case. Following the same derivation as Theorem 3 in [51], the above regret bound would lead to the bound of average regret as follows,

$$\overline{R}_p^T \leq \sum_{I\in\mathcal{I}_p}((1+\frac{\sqrt{2}}{\sqrt{\rho K}})\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + \frac{4}{\sqrt{T}}\sqrt{|A|\Delta\sum_{t=1}^{T}\Pi^t(I)\epsilon^t(I)})$$

Simplifying the first term and rearranging,

$$\overline{R}_p^T \leq (1+\frac{\sqrt{2}}{\sqrt{\rho K}})|\mathcal{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + \frac{4\sqrt{|A|\Delta}}{\sqrt{T}}\sum_{I\in\mathcal{I}_p}\sqrt{\sum_{t=1}^{T}\Pi^t(I)\epsilon^t(I)})$$

$$= (1+\frac{\sqrt{2}}{\sqrt{\rho K}})|\mathcal{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + \frac{4\sqrt{|A|\Delta}}{\sqrt{T}}|\mathcal{I}_p|\frac{\sum_{I\in\mathcal{I}_p}\sqrt{\sum_{t=1}^{T}\Pi^t(I)\epsilon^t(I)}}{|\mathcal{I}_p|}$$

$$\leq (1+\frac{\sqrt{2}}{\sqrt{\rho K}})|\mathcal{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + \frac{4\sqrt{|A|\Delta|\mathcal{I}_p|}}{\sqrt{T}}\sqrt{\sum_{t=1}^{T}\sum_{I\in\mathcal{I}_p}\Pi^t(I)\epsilon^t(I)}$$

Now, consider the average MSE loss $\boldsymbol{L}_{\mathcal{R}}^{T}(\boldsymbol{M}_{r}^{T})$ at iteration T over the samples in strategy memory $\boldsymbol{M}_{r}^{T}$. We start by stating two well-known lemmas:

1. The MSE can be decomposed into bias and variance components

$$E_x[(x-\theta)^2] = (\theta - E[x])^2 + Var(\theta)$$

2. The mean of a random variable minimizes the MSE loss $\arg\min_\theta E_x[(x-\theta)^2] = E[x]$ and the value of the loss when $\theta = E[x]$ is $Var(x)$.

$$\begin{aligned}
\boldsymbol{L}_{\mathcal{R}}^{T} &= \frac{1}{\sum_{I\in\mathcal{I}_p}\sum_{t=1}^{T} x^t(I)} \sum_{I\in\mathcal{I}_p}\sum_{t=1}^{T} x^t(I)||\tilde{r}^t(I) - \mathcal{R}(I|\theta^T)||_2^2 \\
&\geq \frac{1}{|\mathcal{I}_p|T} \sum_{I\in\mathcal{I}_p}\sum_{t=1}^{T} x^t(I)||\tilde{r}^t(I) - \mathcal{R}(I|\theta^T)||_2^2 \\
&= \frac{1}{|\mathcal{I}_p|T} \sum_{I\in\mathcal{I}_p} \Pi^T(I) E_t[||\tilde{r}^t(I) - \mathcal{R}(I|\theta^T)||_2^2 | x^t(I) = 1]
\end{aligned}$$

Let $\mathcal{R}^*$ be the model that minimizes $\boldsymbol{L}^T$ on $\boldsymbol{M}_r^T$. Using the above two lemmas,

$$\boldsymbol{L}_{\mathcal{R}}^{T} \geq \frac{1}{|\mathcal{I}_p|T} \sum_{I\in\mathcal{I}_p} \Pi^T(I)(||\mathcal{R}(I|\theta^T) - E_t[\tilde{r}^t(I)|x^t(I)=1]||_2^2 + \boldsymbol{L}_{\mathcal{R}^*}^{T}). \tag{3.10}$$

Thus, we can get the following inequations:

$$\boldsymbol{L}_{\mathcal{R}}^{T} - \boldsymbol{L}_{\mathcal{R}^*}^{T} \geq \frac{1}{|\mathcal{I}_p|} \sum_{I\in\mathcal{I}_p} \Pi^T(I)\epsilon^T(I) \qquad \sum_{I\in\mathcal{I}_p} \Pi^T(I)\epsilon^T(I) \leq \mathcal{I}_p|(\boldsymbol{L}_{\mathcal{R}}^{T} - \boldsymbol{L}_{\mathcal{R}^*}^{T})$$

Furthermore, we can get the inequation for the regret:

$$\begin{aligned}
\overline{R}_p^T &\leq (1 + \frac{\sqrt{2}}{\sqrt{\rho K}})|\mathcal{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + \frac{4\sqrt{|A|\Delta|\mathcal{I}_p|}}{\sqrt{T}} \sqrt{\sum_{t=1}^{T}\sum_{I\in\mathcal{I}_p} \Pi^t(I)\epsilon^t(I)} \\
&\leq (1 + \frac{\sqrt{2}}{\sqrt{\rho K}})|\mathcal{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + \frac{4\sqrt{|A|\Delta|\mathcal{I}_p|}}{\sqrt{T}} \sqrt{\sum_{t=1}^{T} |\mathcal{I}_p|(\boldsymbol{L}_{\mathcal{R}}^{T} - \boldsymbol{L}_{\mathcal{R}^*}^{T})} \\
&\leq (1 + \frac{\sqrt{2}}{\sqrt{\rho K}})|\mathcal{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + \frac{4\sqrt{|A|\Delta|\mathcal{I}_p|}}{\sqrt{T}} \sqrt{T|\mathcal{I}_p|\epsilon_{\boldsymbol{L}}} \\
&= (1 + \frac{\sqrt{2}}{\sqrt{\rho K}})|\mathcal{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + 4|\mathcal{I}_p|\sqrt{|A|\Delta\epsilon_{\boldsymbol{L}}}
\end{aligned}$$

So far we have assumed that $\boldsymbol{M}_r$ contains all sampled regrets. The number of samples in the memory at iteration t is bounded by $K \cdot |\mathcal{I}_p| \cdot t$. If $K \cdot |\mathcal{I}_p| \cdot T < |\boldsymbol{M}_r|$ then the memory will never be full, and we can make this assumption. Let $\rho = T^{-\frac{1}{4}}$.

$$P(\overline{R}_p^T > (1 + \frac{\sqrt{2}}{\sqrt{T^{-\frac{1}{4}}K}})|\mathcal{I}_p|\Delta\frac{\sqrt{|A|}}{\sqrt{T}} + 4|\mathcal{I}_p|\sqrt{|A|\Delta\epsilon_{\boldsymbol{L}}}) < T^{-\frac{1}{4}}$$

Therefore, for any $\epsilon > 0$,

$$\lim_{T\to\infty} P(\overline{R}_p^T - 4|\mathcal{I}_p|\sqrt{|A|\Delta\epsilon_{\boldsymbol{L}}} > \epsilon) = 0$$

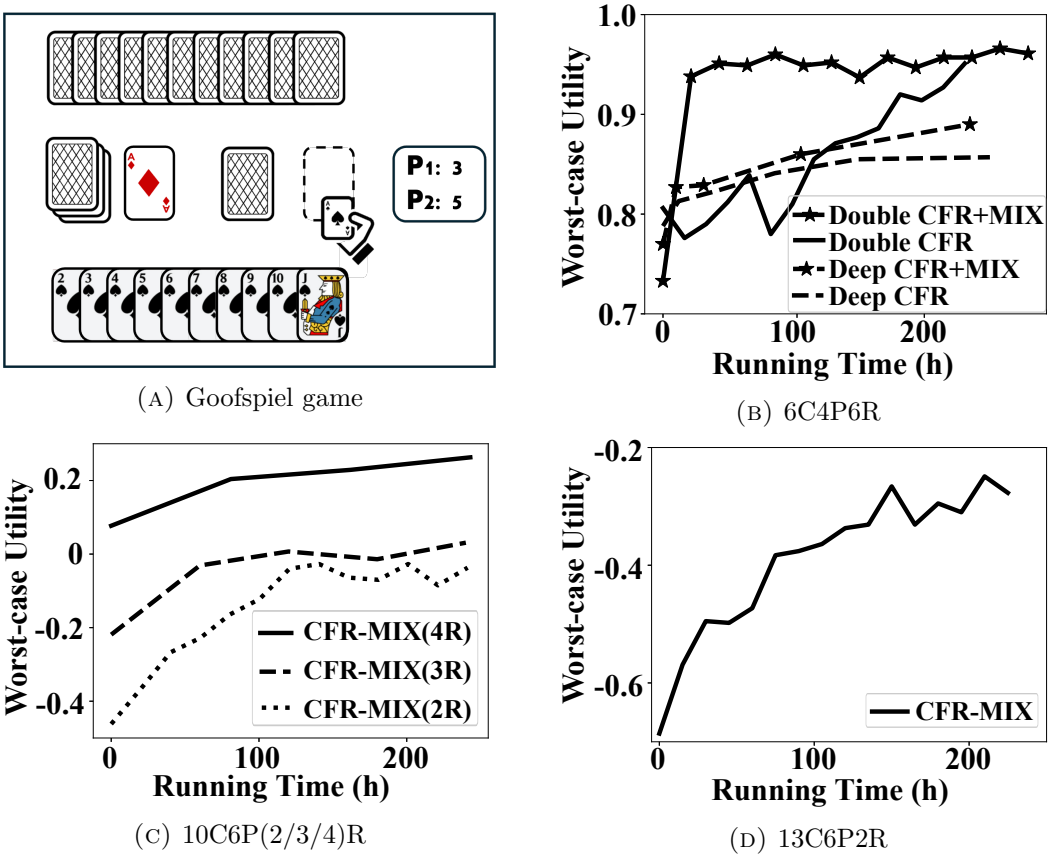
□

3.5 Empirical Evaluation

To evaluate the effectiveness of our CFR-MIX algorithm in solving team-adversary games, we deploy two deep learning-based algorithms, Double Neural CFR [22]and Deep CFR [21], as two base frameworks and integrate the mixing layer into each framework, respectively. For clarity, we use CFR-MIX or Double CFR+MIX to denote the algorithm that incorporates a mixing layer into Double Neural CFR, while Deep CFR+MIX refers to the algorithm that includes a mixing layer in Deep CFR. Without loss of generality, we implement these algorithms on team-adversary games from two distinct domains to assess the performance of our proposed mixing layer. Additionally, we conduct a comparison experiment between CFR-MIX and CFR-MIX without the parameter sharing technique to examine the performance of the parameter sharing technique. All experiments are conducted on a server with a 10-core 3.3GHz Intel i9-9820X CPU and an NVIDIA RTX 2080 Ti GPU.

3.5.1 Goofspiel Game

First, we choose a poker game called Goofspiel, as referenced in [52]. In the Goofspiel game, shown in Figure 3.6a, at the beginning of the game, each player receives a hand of cards numbered A to K. In each round, players secretly bid on the top point-valued card from a point card stack using the cards in their hand, taking turns. The player who bids the highest card value wins the point value of the top



(A) Goofspiel game

(B) 6C4P6R

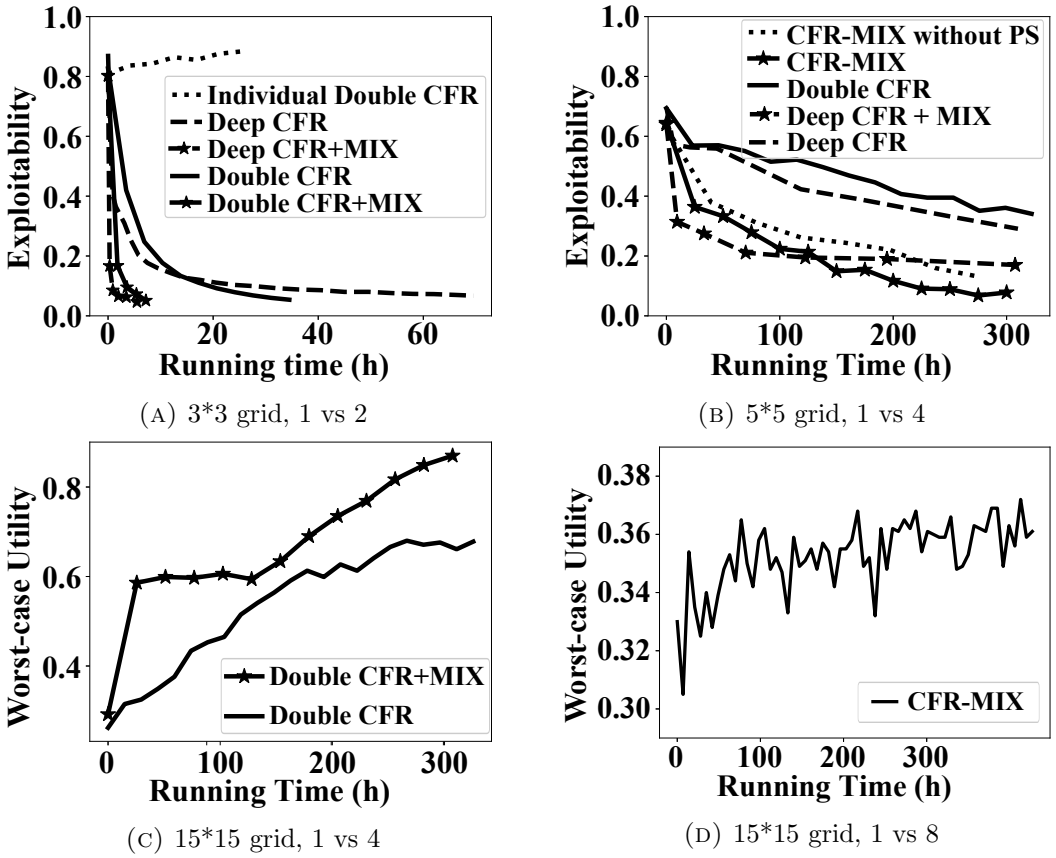
(C) 10C6P(2/3/4)R

(D) 13C6P2R

FIGURE 3.6: Goofspiel games (C: card, P: team player, R: round)

point-valued card. Finally, the player with the highest total points wins the game. In our experiments, we consider a team-adversary variant of the Goofspiel game, in which there are two players and a team player includes several agents. The team is deemed victorious as long as any of its team members win. Additionally, we consider an imperfect-information version of the game [53, 54], where players only know the outcome of each round rather than the cards used by other players for bidding. For the sake of simplicity, the number preceding the letter C refers to the number of hand cards, the number preceding the letter P denotes the number of team members, and the number preceding the letter R indicates the number of rounds played in the game. Figure 3.6 shows the experimental results.

For the Goofspiel game with the 6C4P6R setting, algorithms incorporating a mixing layer outperform their base algorithm counterparts. Both Double CFR and Double CFR-MIX converge to comparable results, while Deep CFR converges to a lower result than Deep CFR-MIX. This could be attributed to the approximation error stemming from the neural network, as Deep CFR requires training a neural network to generate a strategy over a vast action space. In the Goofspiel game



(A) 3*3 grid, 1 vs 2

(B) 5*5 grid, 1 vs 4

(C) 15*15 grid, 1 vs 4

(D) 15*15 grid, 1 vs 8

FIGURE 3.7: Pursuit-evasion games

with the 6C4P6R setting, we exclusively implement our CFR-MIX algorithm, as the joint action space in these games is exceedingly large (approximately 10^6 and 13^6), rendering the baselines impractical. Fortunately, in these games, our CFR-MIX algorithm can obtain satisfactory strategies within a limited timeframe.

3.5.2 Pursuit-Evasion Game

The second game we selected is the NEtwork purSuiT game (NEST) [16, 17], a more realistic version of the pursuit-evasion game. In this game, a team of pursuers aims to capture the evader, while the evader strives to avoid capture [55]. In the NEST game, tracking devices are taken into account, enabling the pursuer team to obtain real-time location information about the evader. Additionally, the game requires setting a predetermined number of steps. If the evader fails to escape within the specified number of steps or gets captured by the pursuer, the game ends, and the pursuer receives a one-unit reward. Conversely, if the evader manages to escape within the specified number of steps, the pursuer earns zero payoffs. For

our testbed, we select a grid map and designate certain nodes as exit nodes. The initial locations for both players are chosen randomly. In this context, we consider varying grid map sizes and differing numbers of team members in the pursuer team.

Figure 3.7 shows the results of pursuit-evasion games with different sizes. From Figures 3.7a-3.7c, we observe that these algorithms with the mixing layer outperform their original algorithms, consistent with the results in the Goofspiel game. To circumvent the large joint action space, we also execute an algorithm in which every agent in the team independently employs Double Neural CFR (Individual Double CFR). Figure 3.7a presents the results, indicating that individual Double CFR fails to achieve low exploitability in a limited time. This finding supports the assertion that independent learning algorithms are inadequate for solving team-adversary games. We also carry out an ablation study to validate the performance of the parameter-sharing technique. The results, as depicted in Figure 3.7b, show that the CFR-MIX algorithm performs better than that without the parameter-sharing technique. In the pursuit-evasion game featuring a 15x15 grid map and four agents in the pursuer team, we did not implement Deep CFR-related algorithms, as Deep CFR is time-consuming and requires large memory for solving such a large game. Despite these limitations, our Double CFR+MIX still significantly outperforms the Double Neural CFR algorithm. Lastly, in the game with a 15x15 grid map and eight agents in the pursuer team, only our CFR-MIX can be executed, as the joint action space for the team is too large for the baseline algorithms. Although our CFR-MIX does not converge within the limited time, it exhibits a tendency to grow in worst-case utility. It suggests that our CFR-MIX can obtain a better strategy compared to the initial random strategy.

Extensive experimental results across games from two distinct domains demonstrate that our CFR-MIX significantly outperforms other CFR-based algorithms. In large-scale team-adversary games with many team members, existing algorithms struggle to solve these games due to the exponential growth of the action space for the team. In contrast, our CFR-MIX is capable of obtaining satisfactory results within a limited time, even in these challenging scenarios.

3.6 Chapter Summary

In this Chapter, we propose a novel CFR-based framework, CFR-MIX, to solve team-adversary games with the combinatorial action space problem to improve scalability. Instead of using the traditional strategy representation over joint-action space, we represent the team's strategy with a novel strategy representation over individual action space to reduce the strategy space. To maintain cooperation among agents in the team, we define a consistency relationship between these two strategy representations. Moreover, to compute the equilibrium under the new strategy representation, we transform the consistency relationship between strategy representations into a consistency relationship between cumulative regret values. Then, we propose a novel product-decomposition method over cumulative regret values to guarantee the consistency relationship between these values. Under the guarantee of these consistency relationships, we propose the CFR-MIX algorithm, which employs a mixing layer to implement the novel decomposition method. Finally, experimental results in games of different domains show that our algorithm significantly outperforms other state-of-the-art CFR-based algorithms.

Chapter 4

Solving Pursuit-Evasion Games via Pretrained Strategy

4.1 Introduction

In this Chapter[1], we focus on solving a specific game, a *Pursuit-Evasion Game*. As we all know, preventing, deterring, and detecting crime to ensure the safety and security of the population is an immensely serious task entrusted to various governmental law enforcement forces. Due to its importance, many game-theoretic models have been developed to safeguard not just the general public, but also critical infrastructure, using limited resources [18]. Among the problems studied, one stands out due to its troubling statistic: police pursuits probably injure or kill more innocent bystanders than any other kind of force [19]. Therefore, it is extremely important to devise scalable methods to efficiently coordinate multiple police units to capture a fleeing criminal as soon as possible, minimizing casualties and property damages. This scenario is commonly referred to as a pursuit-evasion game. It is a special type of two-player zero-sum imperfect-information extensive-form game and also a special team-adversary game played on a graph.

As we introduced in Chapter 2, the contemporary state-of-the-art algorithms for solving imperfect-information extensive-form games can be roughly divided into two groups: no-regret methods derived from CFR [12], and incremental strategy-space generation methods of the PSRO framework [13]. There are indeed CFR

[1]This chapter has been published in [56].

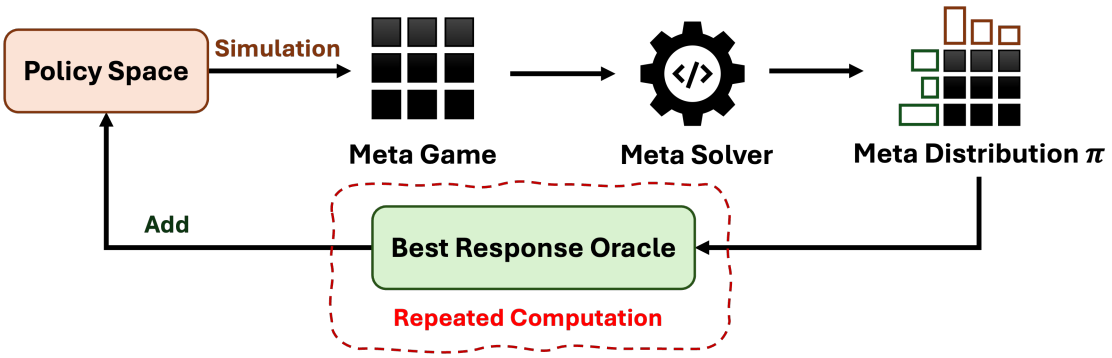


FIGURE 4.1: The structure of police space response oracles.

or PSRO variants capable of solving shallow pursuit-evasion games with many pursuers, such as CFR-MIX [45] introduced in Chapter 3, or deep pursuit-evasion games with a smaller space of legal actions [57, 58]. However, these algorithms cannot easily be applied to solve large-scale pursuit-evasion games due to the long-term decision-making process and extensive action space. For example, although CFR-MIX can solve the large-scale combinatorial action space problem, if the pursuit-evasion game involves a long-term decision-making process, the game tree would be very large, making it inefficient to traverse.

To address the long-term decision-making process in large-scale pursuit-evasion games, we intend to use the PSRO algorithm [13], as it incorporates reinforcement learning (RL) to effectively manage long-term decision-making. PSRO is a generalization of the double-oracle algorithm. Therefore, when the learning is sufficiently precise, PSRO inherits the guarantees of convergence to Nash Equilibrium (NE) from the double-oracle algorithm. Figure 4.1 illustrates the workflow of the PSRO algorithm. PSRO works in iterations, computing the best responses to equilibria of meta-games gradually increasing in size. Finding the best responses constitutes the PSRO's key bottleneck, and several recent adaptations attempted to mitigate it through parallelization [42, 59]. However, these PSRO variants still need to compute the best responses from scratch, restarting the computation in each iteration. The repeated best response computation is the main obstacle to scaling the algorithms to real-world problems. To mitigate this issue, we propose that by pre-training the best responses and later fine-tuning them, we can solve large-scale pursuit-evasion games that are both deep and have a large branching factor.

Pre-training and fine-tuning is a celebrated approach from the area of natural language processing (NLP). For example, BERT [60] is a pre-training method that

can learn a general-purpose language-understanding model on a large text corpus and then use this model for downstream NLP tasks through fine-tuning it. Pre-training is also a key component of the GPT model [61] and was used in AlphaGo [62] and AlphaStar [63] to imitate the behaviors in high-quality human behavioral data. Despite the huge impact of the pre-training and fine-tuning approach on NLP and partially also on computer vision [64, 65], it is still rarely used outside these areas [66]. Especially in the context of general game theory when human behavioral data is not available, the question of whether this approach can be used to enhance the efficiency of equilibrium-finding methods – in terms of optimizing data usage, reducing computational costs, and improving scalability – remains unanswered.

To fill this gap, we propose a two-stage method for solving large-scale pursuit-evasion games by integrating the pre-training and fine-tuning paradigm into the PSRO framework. Our contributions are threefold: i) We modify the graph embedding algorithm `LINE` to learn a better game state representation that can retain both the structure information of the graph and the node information related to the exit nodes. ii) We show how to pre-train a pursuer's policy base model against different strategies of the evader with multi-task reinforcement learning. iii) We formulate a novel best-response oracle in PSRO that fine-tunes the pre-trained pursuer's policy. Finally, we perform extensive empirical analysis in large-scale pursuit-evasion games with both synthetic and real-world graphs to show that our algorithm outperforms other baselines in both solution quality and scalability, which demonstrates the effectiveness of pre-training and fine-tuning. To the best of our knowledge, we are the first to introduce the pre-training and fine-tuning paradigm into an equilibrium-finding algorithm without human behavioral data.

4.2 Pursuit-Evasion Game

In this section, we move to introduce the class of the game we are interested in, the *Pursuit-Evasion Game*. A pursuit-evasion game is a special type of imperfect-information extensive-form game with two players – the pursuer and the evader, hence $N = \{\mathbf{p}, e\}$. We assume the pursuer has control over n law enforcement units, i.e., $\mathbf{p} = (p_1, p_2, ..., p_n)$, equipped with tracking devices capable of monitoring

the real-time location of the evader[2]. In reality, both players act simultaneously, making a pursuit-evasion game a simultaneous-move extensive-form game where the evader moves first, followed by the pursuer who makes decisions without being aware of the current action of the evader.

The pursuit-evasion game is played on an urban road map, which can be represented as a graph $G = (V, E)$, where V is a vertex set, and E is an edge set. A subset $\overline{V} \subset V$ denotes the set of exit nodes from which the evader can escape. For a vertex $v \in V$, let $\mathcal{N}(v)$ define the set of neighboring vertices of vertex v. Let $l_0^e, l_0^{\mathbf{p}}$ be the initial locations of evader and pursuer. Note that $l_0^{\mathbf{p}} = (l_0^{p_1}, l_0^{p_2}, ..., l_0^{p_n})$. By T we denote the number of steps in which the game terminates. At time step $t < T$, the history of the game is a sequence of past locations of both players, $h = (l_0^e, l_0^{\mathbf{p}}, ..., l_{t-1}^e, l_{t-1}^{\mathbf{p}})$. Because the evader cannot know the location of the pursuer, the evader's information set can be defined as $I_e = \{h | h = (l_0^e, l_0^{\mathbf{p}}, l_1^e, *, ..., l_{t-1}^e, *)\}$, where $*$ is any possible location of the pursuer. The actions available to the evader are the neighboring vertices of the evader's current location, i.e., $A_e(h) = \mathcal{N}(l_{t-1}^e)$. The evader makes their decision $l_t^e \in A_e(h)$ based on the information available in set I_e. When the evader moves from l_{t-1}^e to l_t^e, the history becomes $h = (l_0^e, l_0^{\mathbf{p}}, ..., l_{t-1}^e, l_{t-1}^{\mathbf{p}}, l_t^e)$. Then it is up to the pursuer to act in the information set defined as $I_{\mathbf{p}} = \{h | h = (l_0^e, l_0^{\mathbf{p}}, ..., l_{t-1}^e, l_{t-1}^{\mathbf{p}}, *)\}$. The action set of the pursuer at the current history h is a Cartesian product of the sets of neighboring vertices of each unit, i.e., $A_{\mathbf{p}}(h) = \{(l^{p_1}, l^{p_2}, ..., l^{p_n}) | l^i \in \mathcal{N}(l_{t-1}^i), \forall i \in \{p_1, p_2, ..., p_n\}\}$. When the pursuer moves from $l_{t-1}^{\mathbf{p}}$ to $l_t^{\mathbf{p}}$ after choosing an action from $A_{\mathbf{p}}(h)$, it is up to the evader to act again. This process repeats until a termination condition is met. The game ends when (i) the pursuer catches the evader, i.e., $l_t^e \in l_t^{\mathbf{p}}$; (ii) the evader escapes; or (iii) the game reaches the time step T. In cases (i) and (iii), the pursuer obtains a single reward (1), and the evader gets a single penalty (-1). Otherwise, if the evader successfully escapes, the pursuer obtains a single penalty (-1), and the evader gets a single reward (1).

Since the pursuit-evasion game is an imperfect-information extensive-form game, we may leverage the standard methods for finding its Nash Equilibrium (NE) [2]. However, applying these methods directly is not feasible due to the large combinatorial action space of pursuit-evasion games. Some adaptations hence consider various (and often domain-specific) incremental generation strategies [17, 67, 68],

[2]This version of pursuit-evasion games is also called the NEtwork purSuiT games (NEST) [17] or the Network Security Games (NSGs) [57, 58]

or introduce new strategy representations in an attempt to avoid the curse of dimensionality associated with the game's strategy space [45]. However, neither of these approaches scales to deep games with a high number of time steps. To solve such games, we develop a PSRO-based algorithm embedded with pre-training and fine-tuning of pursuer's strategies to accommodate both the high branching factor and the long interactions between the players.

4.3 Related Works

In this section, we introduce the PSRO algorithm and multi-task reinforcement learning algorithms, which will be used in our algorithm.

4.3.1 Policy-Space Response Oracles

Double oracle is an incremental strategy-space generating algorithm that is guaranteed to converge to Nash equilibrium in two-player zero-sum games [11]. The idea of the algorithm is to gradually expand a small meta-game, consisting of only a subset of all possible actions of the players, by including their best responses (from the entire action space) against the equilibrium of the meta-game. Once all the best responses are already present in the meta-game, meaning the current solution cannot be improved, the algorithm terminates. PSRO is a generalization of the double oracle, but instead of actions, PSRO uses policies as choices in the meta-game. Due to its favorable scalability, PSRO achieved state-of-the-art performance in large-scale two-player zero-sum games [13]. In PSRO, every player starts with a random policy, which constitutes the initial meta-game. Then, as in double-oracle, it computes the best response policies of all players and adds them to the meta-game. Any reinforcement learning (RL) algorithm can be employed for computing the best responses. The meta-game in PSRO is represented as an empirical game matrix, generated by having each policy of the first player interact with each policy of the second player and recording the average utilities as entries. The solution (i.e., the meta-strategy) is computed from the empirical game matrix using any meta-solver. Once PSRO converges, it outputs the final distribution over population policies, which approximates an NE of the original game.

A notable disadvantage of PSRO is the fact that training the best response policy is computationally demanding, making it too slow for large games. Several recent works have focused on addressing this issue. For example, McAleer et al. introduced the Pipeline PSRO (P2SRO), which parallelizes the computation while maintaining convergence guarantees through a hierarchical pipeline of reinforcement learning (RL) workers [42]. Through various speedup techniques, contemporary PSRO-based methods have mastered some large games like Stratego (in the case of P2SRO) or Starcraft [63]. Yet, even these breakthroughs still require computing the best response from scratch in each iteration.

4.3.2 Multi-Task Reinforcement Learning

Multi-task learning is an area of machine learning widely applied to problems in natural language processing (NLP) or computer vision (CV) due to its ability to significantly improve learning efficiency and prediction accuracy through joint learning [69]. Its subfield applied to RL tasks, has recently drawn the attention of the RL community as well, motivated by the idea that similar decision-making policies may be applied to different tasks in similar environments [70]. Each RL task is modeled as a Markov decision process (MDP), and the goal of RL is to choose actions that maximize cumulative rewards. Wilson et al. suggest that to model a distribution over MDPs, we may employ a hierarchical Bayesian infinite mixture model [71]. In this approach, previously learned distributions are used as an informative prior for model-based Bayesian RL when solving a new MDP. Multiple tasks may be trained simultaneously using a framework of shared experiences that uses task-specific rewards to identify similar parts, defined as shared regions, which can guide the experience-sharing of task policies [72]. Recent work has shown that multi-task pre-training with fine-tuning on new tasks performs equally or better than meta-learning pre-training with meta-adaptation in RL tasks [73]. This motivates the adoption of multi-task learning as our pre-training algorithm.

4.4 Pre-Training and Fine-Tuning in PSRO

In this section, we describe our algorithm for solving large pursuit-evasion games using pre-trained strategies. The algorithm can be divided into two stages: the

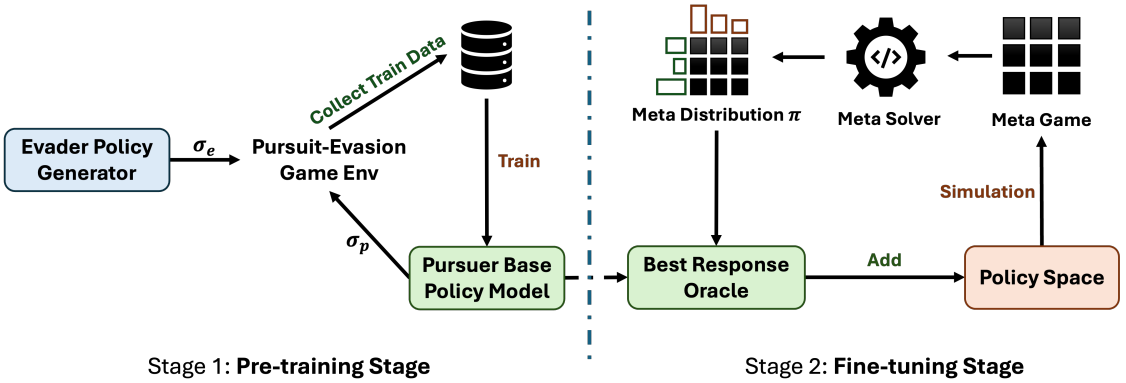


FIGURE 4.2: Structure of our PSRO with pre-trained strategies algorithm.

pre-training stage and the fine-tuning stage, as shown in Figure 4.2. In the pre-training stage, we first learn a good representation of the state and then train the policy base model of the pursuer against different evader strategies using multi-task learning. In the fine-tuning stage, we use the pre-trained base model in the pursuer's best response oracle in PSRO to arrive at the equilibrium of the game.

4.4.1 Game State Representation

First, we consider the computation of the best response of the pursuer during the pre-training stage. As described in Section 4.2, the information set of the pursuer is determined by the past locations of both players, i.e., $(l_0^e, l_0^{\mathbf{P}}, ..., l_{t-1}^e, l_{t-1}^{\mathbf{P}})$. This history can be considered as the state in the best response computation task. If we directly use the vertex index in the graph to represent the state, this representation would not retain the graph's structural information since the vertex index is essentially meaningless. To obtain a better state representation to facilitate learning, we adopt a graph embedding algorithm to learn the representation of the graph vertex and replace the vertex index with the vertex's learned representation. These embeddings need to preserve not only the structure information of the graph but also the node information proximity. We modify an existing graph embedding algorithm to achieve this.

Large-scale information network embedding (`LINE`) is a graph embedding algorithm that aims to embed large information networks into low-dimensional vector spaces [74]. This method can preserve both the first-order and second-order proximities. First-order proximity refers to the local pairwise proximity between the

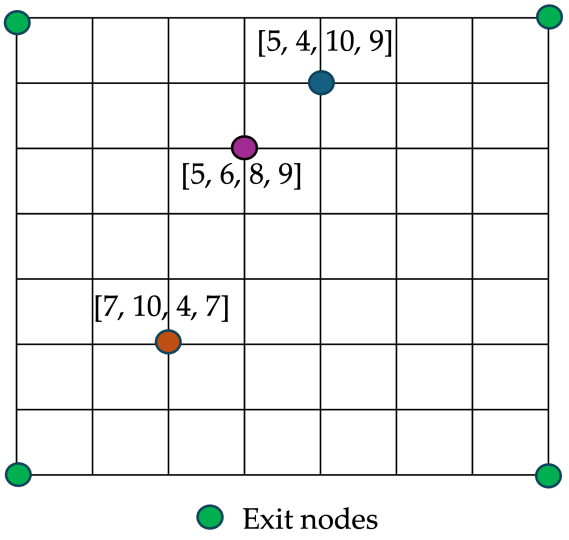


FIGURE 4.3: Example of node information vector.

vertices, while second-order proximity assumes that vertices sharing many connections to other vertices are similar. Note that in pursuit-evasion games, these exit nodes are important for both players. We hence use the information related to these exit nodes as node information and require nodes with similar node information to have similar embeddings, whether they are connected or not. Therefore, in addition to the first-order and second-order proximities, we also aim to preserve this so-called node information proximity.

Node Information. We define the node information of a vertex as the shortest distance between the vertex and these exit nodes. Figure 4.3 shows some examples of node information vectors. Formally, we use D^i_{NI} to denote the node information vector for vertex $v_i \in V$.

Node Information Proximity. For any pair of vertices (v_i, v_j), we use the cosine similarity as the similarity metric of node information, defined as follows:

$$f_{NI}(D^i_{NI}, D^j_{NI}) = \frac{D^i_{NI} \cdot D^j_{NI}}{||D^i_{NI}|| \times ||D^j_{NI}||}.$$

Objective Function. For any pair of vertices (v_i, v_j), we define the node similarity between v_i and v_j as a joint probability between the two vertices:

$$p_{NI}(v_i, v_j) = \frac{1}{1 + \exp(-u_i^T \cdot u_j)},$$

where $u_i \in R^d$ is the low-dimensional vector representation of vertex v_i. Moreover, we denote the empirical probability of the above distribution $p_{NI}(\cdot,\cdot)$ over the space $V \times V$ as $\hat{p}_{NI}(i,j) = f_{NI}(D^i_{NI}, D^j_{NI})$, where f_{NI} is the cosine similarity function. Therefore, to preserve the node information proximity, we need to minimize the following objective function:

$$O_d = d(\hat{p}_{NI}(\cdot,\cdot), p_{NI}(\cdot,\cdot)),$$

where $d(\cdot,\cdot)$ is the distance metric between these two distributions. In case of the KL-divergence, the function would be reduced to:

$$\begin{aligned} O_{KL} &= d(\hat{p}_{NI}(\cdot,\cdot), p_{NI}(\cdot,\cdot)) \\ &= \sum_{\forall(v_i,v_j)} f_{NI}(D^i_{NI}, D^j_{NI}) \log \frac{f_{NI}(D^i_{NI}, D^j_{NI})}{p_{NI}(v_i, v_j)}. \end{aligned}$$

After eliminating the constants, we finally arrive at:

$$O_{KL} = -\sum_{\forall(v_i,v_j)} f_{NI}(D^i_{NI}, D^j_{NI}) \log p_{NI}(v_i, v_j).$$

To preserve the structure information of the graph and node information at the same time, we add O_{KL} to the objective function of the `LINE` algorithm.

State Representation. After obtaining the embedding for all vertices in the graph, we can replace the vertex index with its corresponding embedding in the information set representation $(l^e_0, l^{\mathbf{P}}_0, ..., l^e_{t-1}, l^{\mathbf{P}}_{t-1})$. However, the size of the state representation quickly increases with the increase of time step t. For this reason, we propose to represent the information set using only the locations of both players and the number of remaining time steps. Note that such a simplification still carries all the necessary information, as the locations of both players in the next time step depend only on their current positions and their actions. In the experimental section, we also show that this new state representation performs equally well or better than the traditional state representation using historical locations.

4.4.2 Policy Base Model for Pursuer

The computation of the pursuer's best response strategy against a fixed evader's strategy can be regarded as a reinforcement learning (RL) task. The environment of this task depends mainly on the rules of the game and the fixed evader's policy. Specifically, the task changes according to the strategy of the evader. Under the PSRO framework, the pursuer's best response strategy is computed in each iteration, with each iteration characterizing a specific RL task identified by a different strategy of the evader. We can assume that these RL tasks may be seen as similar, following the same distribution, and are determined by the evader's strategy.

We use this similarity in practice to speed up the computation of the pursuer's best response under the PSRO framework. To this end, we pre-train the policy base model of the pursuer and quickly fine-tune it to adapt to a new, similar RL task. To achieve this, we need to solve two problems: how to generate the tasks, and how to learn the policy base model.

Task Generation. Since these RL tasks under the PSRO framework are determined by the evader's strategy, we need to generate as many different evader strategies as possible to obtain enough tasks for training the policy base model of the pursuer. Recall that the goal of the evader is to escape through exit nodes in the graph. Therefore, we focus on the so-called High-Level Actions of the evader, which proved useful in earlier work [57, 58]. Under this action representation, the evader chooses an exit node to escape from and then samples a path from the initial location to this exit node, rather than deciding where to go in the next step as in regular strategies. In the PSRO algorithm, when computing the best response strategy for the pursuer, the fixed evader strategy is a mixed strategy produced by the meta distribution and previous best response strategies. Without loss of generality, we generate the mixed strategy for the evader randomly. In practice, this means that if there are n exit nodes, we randomly generate a distribution σ_e over n exit nodes. In this way, we can generate enough RL tasks for pre-training across a wide range of meaningful evader strategies.

Policy Learning. After obtaining enough RL tasks, we move to the pre-training phase of the pursuer's policy base model. Our priority is that the model generalizes well, meaning it needs to be able to fine-tune quickly to deliver acceptable performance when faced with a new task determined by a previously unseen evader's

Algorithm 2: Pre-train Pursuer's Policy Base Model

```
Initialize a pursuer's policy network θ and buffer B;
for train epoch in {1, 2, ..., n} do
    for number of tasks sampled do
        Generate an evader's strategy σ_e randomly;
        Sample training data using evader's strategy σ_e and pursuer's policy θ;
        Add sampled training data into buffer B;
    Train pursuer's policy θ using training data in buffer B ;
    Clear buffer B;
return policy network θ.
```

strategy. For this purpose, we adopt a multi-task reinforcement learning approach to guide the pre-training process. By doing so, we complete all the training tasks simultaneously with a single RL policy network. When adapting to a new task, we fine-tune the policy network using the same algorithm as in the pre-training stage. The framework of multi-task reinforcement learning is agnostic to the underlying base RL algorithm. We opt for a popular on-policy method called proximal policy optimization (PPO) [75], which is known to perform extremely well across a wide range of tasks. The diagram and the entire procedure of pre-training are depicted in Figure 4.2 and Algorithm 2.

First, we initialize a policy network θ and a buffer B. In the pre-training process, we use the experience-sharing approach to conduct multi-task learning. We begin the training by obtaining some tasks from several generated evader strategies. Then we use the current policy θ to sample the training data from these tasks and add the data into the buffer B. We train the policy θ based on the data in the buffer using the base RL algorithm (e.g., PPO). We finish the epoch by emptying the buffer. This whole process is repeated for a given number of training epochs.

Since the pursuer controls several law enforcement units, their joint action space grows exponentially in the number of units. Consequently, training the joint action strategy over the large joint action space using PPO would be highly ineffective. To mitigate this issue, we replace the vanilla PPO with the Multi-Agent PPO (MAPPO) [76], a variant of PPO with centralized value function inputs. The motivation for choosing a multi-agent RL algorithm is that all units need to cooperate to capture the escaping evader. Computing the best response of the pursuer with many units may hence be regarded as a multi-agent cooperative RL problem. To use MAPPO as the base RL algorithm for multi-task reinforcement learning,

we only need to change the training pattern in Algorithm 2. We note that any multi-agent cooperative RL algorithm may be used in the pre-training stage.

4.4.3 PSRO with Pre-trained Strategy

Algorithm 3: PSRO with Pre-trained Pursuer's Policy Model

Input: Pre-trained policy θ_p of the pursuer

Initialize policy θ_e^0 of the evader randomly, $S_e = \{\theta_e^0\}$;
Initialize pursuer's policy using pre-trained policy θ_p, i.e., $\theta_p^0 = \theta_p$, $S_p = \{\theta_p^0\}$;
Initialize meta distribution ρ;
while *epoch i in* $\{1, 2, \ldots, n\}$ **do**
    Compute the evader's best response θ_e^i against pursuer's strategy ρ_p;
    $\theta_p^i = \theta_p$;
    for *few episodes* **do**
        Sample $\theta_e \sim \rho_e$;
        Train oracle θ_p^i over sampling data $\phi \sim (\theta_e, \theta_p^i)$;
    $S_e = S_e \cup \{\theta_e^i\}$, $S_p = S_p \cup \{\theta_p^i\}$;
    Compute the missing entries in the meta-game based on $S_e \times S_p$;
    Compute a meta-strategy ρ based on mete-game using meta solver;
return *Current solution strategy* ρ *and* (S_e, S_p).

Here, we describe how to integrate the pre-trained model of the pursuer's policy into the PSRO framework. The diagram of the integration and its method are shown in Figure 4.2 and Algorithm 3. Initially, we randomly initialize the evader's strategy and initialize the pursuer's strategy using the pre-trained policy base model. Then, we follow the standard PSRO loop: in each iteration, we compute the best responses of both players using their appropriate oracles, update the meta-game matrix, and compute the meta-strategy. Without a doubt, the key component of PSRO is the computation of best responses, which we derive in the next two paragraphs.

Best Response Oracle for Evader. Since the goal of the evader is to escape through these exit nodes in the graph and they cannot obtain additional information about the pursuer's location during the game, its best strategy should be to follow a path from its initial location to an exit node. However, evaluating every path to select the best one is inefficient because the number of paths grows exponentially with the number of time steps. Therefore, we employ the High-Level Actions [57, 58] described in the Task Generation part. The evader needs to select which exit node to escape from and sample one path from the initial location to

that exit node. We then estimate the value of each exit node through simulations. The best response strategy of the evader is selected according to these values.

Best Response Oracle for Pursuer. The primary difference between vanilla PSRO and our PSRO with integrated pre-training and fine-tuning paradigm lies in the best response computation for the pursuer. Instead of training the pursuer's best response strategy from scratch, we only need to fine-tune the pre-trained pursuer's policy base model to arrive at the best response strategy. As shown in Algorithm 3, we start with the pre-trained policy model (Line 6) and sample the training data against the evader's mixed strategy. Then we fine-tune the pre-trained model on the sampled data using the same RL algorithm as used in the pre-training stage but with a smaller learning rate (Lines 7-9).

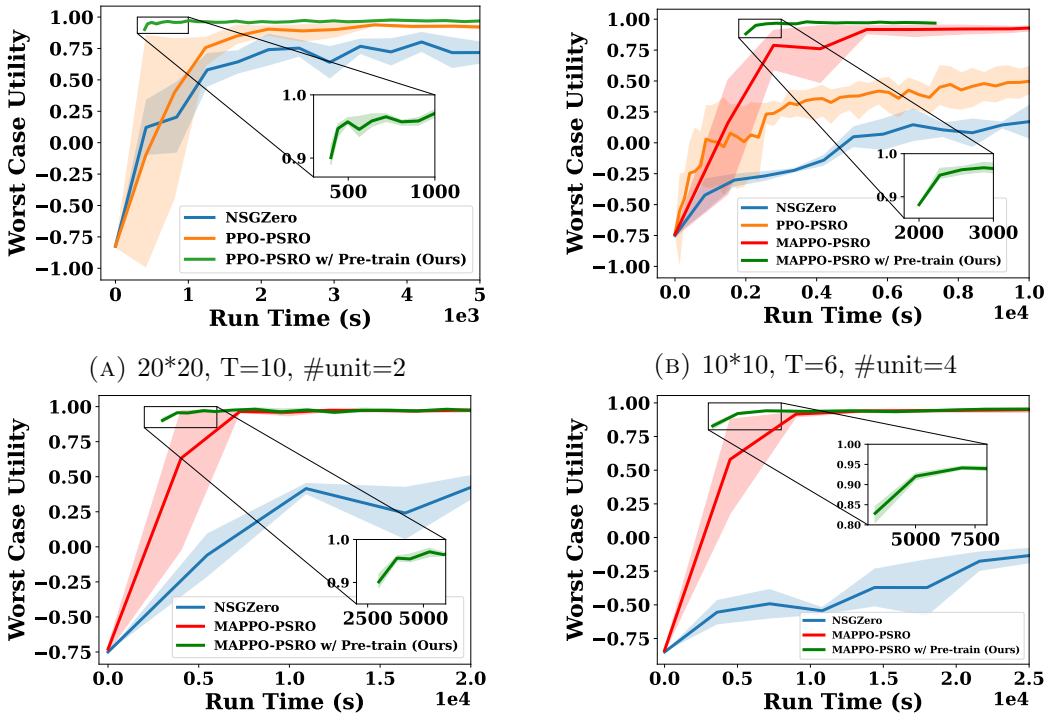
4.5 Empirical Evaluation

To evaluate the effectiveness of our algorithm, we compare it against vanilla PSRO [13] and NSGZero [58], which is the state-of-the-art algorithm for solving pursuit-evasion games, in games with different sizes of graphs[3]. Furthermore, we perform several ablation studies to identify the effects of each component of our algorithm.

4.5.1 Experimental Setting

Pursuit-evasion games are played on graphs, and the size of these graphs partly determines the sizes of the games. In the experiments, we use different sizes of synthetic graphs and a real-world map. For the synthetic graphs, we generate different sizes of grid worlds, and for the real-world graph, we extract the graph from the Singapore map via OSMnx [77]. Since we focus on the quality of the pursuer's strategy, the measurement we used is the worst-case utility for the pursuer, which is the utility of the pursuer against the evader's best response strategy. To increase the robustness, we run each experiment with three seeds and plot the average values and variances in every figure. Experiments are performed on a workstation with a ten-core 3.3GHz Intel i9-9820X CPU and NVIDIA RTX 2080 Ti GPU.

[3] As mentioned in Section 4.1, CFR variants cannot effectively solve games with such long time horizons, so we do not include them in the experimental comparison.



(A) 20*20, T=10, #unit=2

(B) 10*10, T=6, #unit=4

(C) 20*20, T=11, #unit=4

(D) 50*50, T=26, #unit=4

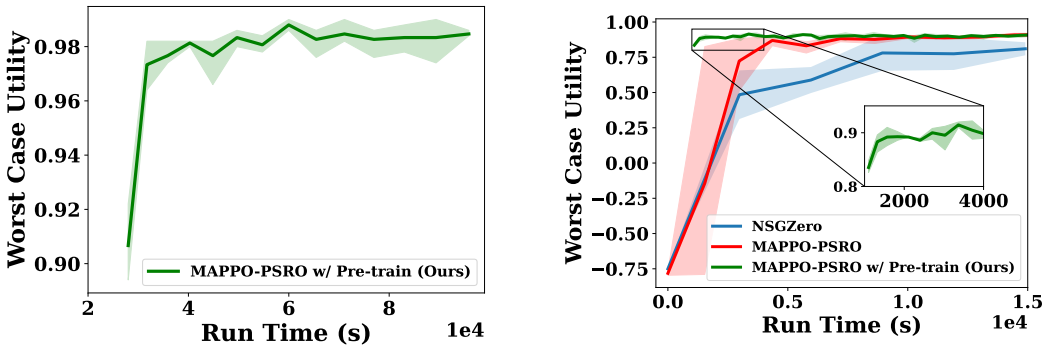
FIGURE 4.4: Results on games with different graphs.

4.5.2 Experimental Results

To evaluate the performance of our algorithm, we conduct experiments on pursuit-evasion games with different sizes of synthetic graphs and real-world graphs.

Synthetic Graphs. First, we adopt pursuit-evasion games that play on synthetic graphs as testbeds. Figure 4.4a shows the results of the game on a 20*20 grid graph. Since the number of units is only two in this game, we use PPO as the base RL algorithm of the multi-task reinforcement learning to pre-train the pursuer's policy base model. From the figure, we can see that PPO-PSRO and NSGZero show similar performance, while both perform worse than our algorithm (PPO-PSRO with Pre-train). When performing PSRO to get the NE strategy, although it takes some time for pre-training, our algorithm starts with a high worst-case utility and improves through fine-tuning. This demonstrates the effectiveness of pre-training and the necessity of fine-tuning. Next, we consider pursuit-evasion games with more units controlled by the pursuer.

Figure 4.4b shows the results in the game with four units played on a 10*10 grid graph. In this game, we use the multi-agent RL algorithm (i.e., MAPPO) as the



(A) 100*100, T=51, #unit=8

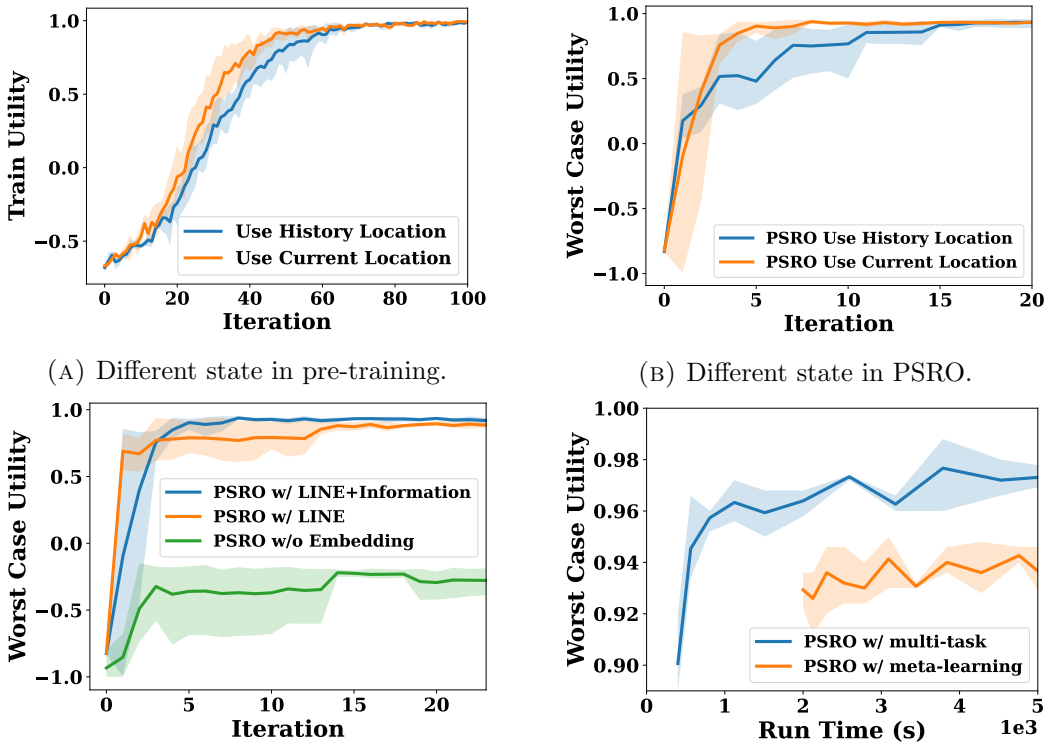
(B) Singapore Map, T=15, #unit=6

FIGURE 4.5: Results on games with large-scale and real-world graphs.

best response computation strategy to mitigate the exponentially growing action space problem. The results show that MAPPO-PSRO and MAPPO-PSRO with Pre-train perform better than PPO-PSRO and NSGZero, indicating that the large joint action space hampers the performance of PPO. Although NSGZero tries to mitigate the large action space issue by only considering legal actions and taking legal action as the input, it still suffers from the issue of combinatorial action space when the graph is compact. Figures 4.4b-4.4d show the results in games with four units played on different sizes of graphs. They demonstrate that our algorithm performs best in all games, even though it requires more time for pre-training.

To further evaluate the scalability of our algorithm, we test it on a game played on a 100*100 grid graph, which involves 10000 nodes and 39600 edges. The number of units is set to eight, and the time horizon is 51. Figure 4.5a shows only the results of our two-stage algorithm (PSRO with Pre-train) since PSRO and NSGZero cannot converge to a satisfactory result in the limited time. It indicates that only our algorithm can solve such large games within an acceptable timeframe.

Real-world Graphs. Finally, we evaluate our algorithm in a pursuit-evasion game with a real-world graph extracted from the Singapore map, which involves 372 nodes and 1098 edges, with a maximum degree of 13. Figure 4.5b shows the results, indicating that our two-stage algorithm performs better and faster than vanilla PSRO. This demonstrates that our algorithm can effectively handle large-scale pursuit-evasion games regardless of the graph structure.



(A) Different state in pre-training.

(B) Different state in PSRO.

(C) Different representation method.

(D) Different pre-training algorithm.

FIGURE 4.6: Results of the ablation studies.

4.5.3 Ablation Study

To evaluate the effectiveness of each component in our algorithm – especially whether the state representation and the multi-task learning improve the performance – we ran several ablation experiments in a game played on a 20*20 grid graph. The pursuer in this game controls two units, and the time horizon is set to be 10. We employ PPO as the algorithm for computing the best response strategy.

Different State Representation. As described in Section 4.4.1, instead of using the historical locations of both players, we only use the current positions of both players and the remaining time as the state representation. To compare these two state representations, we perform the pre-training process and PSRO using both state representations, with each using the same node embeddings. Figures 4.6a-4.6b show the results. We can find that the performances of these two state representations are similar, and the new state representation performs even better. This indicates that the new state representation can perform equally well or even better than the state representation using historical locations.

To evaluate the effectiveness of the graph embedding algorithm, we use the state representation without graph embedding (only using the vertex index) as a baseline, and the `LINE` algorithm is also taken as another baseline to test the effect of node information. Figure 4.6c shows the results. It indicates that PSRO with `LINE`+Information converges to the highest worst-case utility, while PSRO without embedding cannot converge to a high utility. PSRO with `LINE` and PSRO with `LINE`+Information perform much better than PSRO without embedding, demonstrating that node embeddings can improve performance. Although PSRO with `LINE` performs better than PSRO with `LINE`+Information at the beginning, PSRO with `LINE`+Information eventually converges to a higher worst-case utility and runs more stable than PSRO with `LINE`. It suggests that the node information can aid in solving pursuit-evasion games and make the learning phase more stable.

Different Pre-training Algorithms. Meta-learning aims to bootstrap from a set of tasks to learn faster on a new task. Therefore, a meta-learning algorithm can also be used to replace the multi-task reinforcement learning algorithm in our framework to pre-train the policy base model for the pursuer. Here, we adopt Model-Agnostic Meta-Learning (MAML) [78] as the representative of meta-learning, as it is one of the popular meta-learning algorithms and compatible with any model trained with gradient descent and applicable to various learning problems, including RL tasks.

To test the performance of our algorithm using different pre-training algorithms, we pre-train the policy base models using multi-task learning and meta-learning algorithms, respectively. Then, we fine-tune these two pre-trained models under the PSRO framework. Figure 4.6d shows the results. It demonstrates that both methods can start with a high worst-case utility, indicating that both pre-trained policy models are well-trained. From the figure, we can also see that the meta-learning algorithm requires more time to pre-train the policy base model compared to multi-task learning. Furthermore, the pre-trained model using multi-task learning seems to fine-tune better than the pre-trained model using meta-learning. This indicates that while both algorithms can produce well-trained models, multi-task learning tends to be computationally cheaper than meta-learning.

4.6 Chapter Summary

In this Chapter, we propose a two-stage method for solving large-scale pursuit-evasion games by introducing the pre-training and fine-tuning paradigm under the PSRO framework. Instead of training the pursuer's best response strategy from scratch, we fine-tune a pre-trained policy model to speed up the computation of the best response in the PSRO algorithm. To obtain a pre-trained policy model, we use a graph embedding algorithm to achieve a good state representation to facilitate the pre-training and adopt a multi-task reinforcement learning algorithm to pre-train the pursuer's policy base model. Finally, extensive experimental results in pursuit-evasion games on both synthetic and real-world graphs show that our two-stage algorithm outperforms other baselines in terms of speed and scalability.

Part II

Generalizable Algorithm for IIEFGs

Chapter 5

Solving a Set of Pursuit-Evasion Games via GRASPER

5.1 Introduction

In this Chapter[1], we focus on proposing a generalizable algorithm for solving a set of IIEFGs, *Pursuit-Evasion Games*. As we introduced in Chapter 4, the pursuit-evasion game is a special type of two-player zero-sum imperfect-information extensive-form game and it can be used to model many real-world urban security scenarios. In particular, the pursuit-evasion game (PEG) has been extensively employed to model the interactions between a team of pursuers (e.g., police forces) and an evader (e.g., a criminal) on graphs (e.g., urban street networks) [16, 17, 57, 80]. Several methods have been proposed to effectively solve PEGs, such as counterfactual regret minimization (CFR)-based algorithms [80] and policy-space response oracles (PSRO)-based algorithms [57, 58]. Particularly, PSRO with pre-trained strategies [56] introduced in Chapter 4 is custom designed for solving PEGs.

Although many existing works have achieved significant success, they only focus on solving specific PEGs with predetermined initial conditions, e.g., the initial locations of all players and exits, and the time horizon of the game. Unfortunately, these conditions may vary substantially in real-world scenarios, where crimes can occur at any location in a city and at any time. When the initial conditions change,

[1]This chapter has been published in [79].

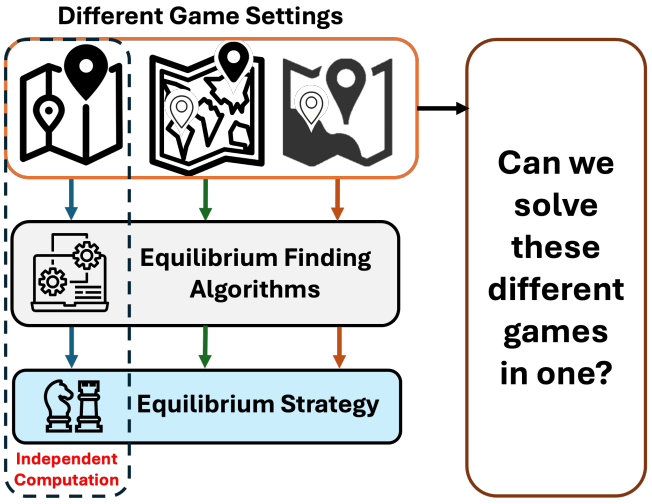


FIGURE 5.1: The re-computation problem.

existing algorithms must solve the new PEG from scratch, which is computationally demanding and time-consuming [56], restricting the real-world deployment of current algorithms. Figure 5.1 illustrates this problem. Therefore, this brings us to the core research question: *Can we solve* ***a set of different pursuit-evasion games*** *in one?* Put simply, can we develop a new approach capable of solving different PEGs with varying initial conditions effectively?

To this end, we introduce GRASPER: a GeneRAlist purSuer for Pursuit-Evasion pRoblems, which can effectively solve different PEGs by generating the pursuer's policies conditional on the initial conditions of the PEGs. Specifically, GRASPER consists of two critical components. First, as the PEG is played on a graph, it is natural to use a *Graph Neural Network (GNN)* to encode the PEG with the given initial conditions into a hidden vector. Second, inspired by recent work on generalization over games with different population sizes [81], we introduce a *hypernetwork* to generate the base policy for the pursuer conditional on the hidden vector obtained by the GNN. This generated base policy then serves as a starting point for the pursuer's best response oracle at each PSRO iteration.

Provided the architecture of GRASPER, the next challenge is how to efficiently train these proposed neural networks. Multi-task learning is one of the common methods to endow networks with generalizable ability [69]. However, directly applying the multi-task learning method to train these neural networks within the architecture of GRASPER is inefficient since jointly training the GNN and hypernetwork using

multi-task learning could be time-consuming. To achieve effective training, we propose an efficient three-stage training method. First, since the GNN is only used to encode the initial conditions that are fixed during the game playing, we introduce a pre-pretraining stage to train the GNN by using self-supervised graph learning methods such as Graph Masked Auto-Encoder (GraphMAE) [27]. Second, we fix the GNN and pre-train the hypernetwork by using a multi-task training procedure where the training data is sampled from different PEGs with different initial conditions. In this stage, to overcome the low exploration efficiency due to the pursuers' random exploration and the evader's rationality, we propose a Heuristic-guided Multi-task Pre-training (HMP) where a reference policy derived by heuristic methods such as Dijkstra is used to regularize the pursuer policy. Finally, we follow the PSRO framework and obtain the pursuer's best response policy at each iteration by fine-tuning the base policy generated by the hypernetwork.

Our contributions are threefold: i) We propose GRASPER which is the first generalizable framework capable of efficiently providing highly qualified solutions for different PEGs with different initial conditions. ii) We propose a three-stage training method to efficiently train the networks of GRASPER, which includes a pre-pretraining stage to train the GNN through GraphMAE, a pre-training stage to train the hypernetwork through heuristic-guided multi-task pre-training (HMP), and a fine-tuning stage to obtain the pursuer's best response policy at each PSRO iteration. iii) We perform extensive experiments, and the results demonstrate the superiority of GRASPER over different baselines in terms of solution quality and generalizability. Finally, we demonstrate that GRASPER provides a versatile approach for solving pursuit-evasion problems across a broad range of scenarios, enabling practical deployment in real-world situations.

5.2 Problem Statement

In this section, we first briefly introduce the definition of pursuit-evasion games. Then, we discuss the research problem we aim to solve in this Chapter.

5.2.1 Pursuit-Evasion Game

As introduced in Chapter 4, the pursuit-evasion game (PEG) is a special type of imperfect-information extensive-form game with two players – a pursuer and an evader, i.e., $N = \{\mathbf{p}, e\}$. The pursuer is assumed to have control over n law enforcement units, denoted as $\mathbf{p} = \{p_1, p_2, ..., p_n\}$, and the pursuer can obtain the real-time location information of the evader with the help of tracking devices.

Here, we follow the definition of pursuit-evasion game in Chapter 4. The pursuit-evasion game is played on urban road maps, represented by a graph $G = (V, E)$, where V is the set of vertices and E is the set of edges. Let $\overline{V} \subset V$ denote the set of exit nodes from which the evader can escape and T the predetermined steps in which the game terminates. At time step $t \leq T$, the locations of the evader and pursuer are denoted by l_t^e and $l_t^{\mathbf{p}} = (l_t^{p_1}, l_t^{p_2}, \ldots, l_t^{p_n})$, respectively. Then, the history of the game at time step t is a sequence of past locations of both players, i.e., $h = (l_0^e, l_0^{\mathbf{p}}, ..., l_{t-1}^e, l_{t-1}^{\mathbf{p}})$. The available action set for both players is the neighboring vertices of the player's current location, i.e., $A_e(h) = \mathcal{N}(l_{t-1}^e)$ and $A_{\mathbf{p}}(h) = \{(l^{p_1}, l^{p_2}, ..., l^{p_n}) | l^i \in \mathcal{N}(l_{t-1}^i), \forall p_i \in \mathbf{p}\}$ where $\mathcal{N}(v)$ denotes the set of neighboring vertices of vertex v. According to the definition of history, we define the information set for each player as the set consisting of indistinguishable histories. As the evader cannot get the pursuer's real-time location information, the information set of the evader is defined as $I_e = \{h | h = (l_0^e, l_0^{\mathbf{p}}, l_1^e, *, ..., l_{t-1}^e, *)\}$, where $*$ represents any possible location of the pursuer. Although the pursuit-evasion game is a simultaneous-move game, we can model it as a turn-based extensive-form game by assuming that the evader acts first and then the pursuer commits without any information about the evader's current action. Then the pursuer's information set can be defined as $I_{\mathbf{p}} = \{h | h = (l_0^e, l_0^{\mathbf{p}}, ..., l_{t-1}^e, l_{t-1}^{\mathbf{p}}, *)\}$.

Note that at each time step t, each agent within the pursuer team gets an observation $o_t^{p_i} = (l_t^{\mathbf{p}}, l_t^e, p_i, t) \in \mathcal{O}^i$, which includes all players' current locations, the ID number of the agent within the team, and the current time step number. Each agent within the pursuer team p_i constructs a policy[2] $\sigma^p : \mathcal{O}^i \to \Delta(A_{p_i})$, which assigns a probability distribution over the action set $A_{p_i}(o_t^{p_i}) = \mathcal{N}(l_t^{p_i}), \forall p_i \in \mathbf{p}$.

[2]All agents with the pursuer team share one policy. As the observations include ID numbers, agents with different ID numbers would have different behavior [82]. Δ denotes the simplex.

As for the evader's policy, we follow previous works [57, 58] that employ High-Level Actions for the evader, as used in Chapter 4. That is, the evader only chooses the exit node to escape from and then samples one path from the initial location to the chosen exit node, instead of deciding actions step by step. Specifically, at time step $t = 0$, the evader first determines an exit node $v \in \overline{V}$ using the policy σ^e, which is a probability distribution over exit nodes, i.e., $\Delta(\overline{V})$. Then the evader samples one path from the initial location l_0^e to the chosen exit node v. Finally, the evader takes actions based on the path when playing the game with the pursuer.

Here, we give some remarks on the adaptation of the High-Level Actions of the evader. (i) In our game setting, the evader lacks real-time access to the pursuers' locations, requiring them to act without any information about their whereabouts. Therefore, sampling one path for the evader would not lose much information compared with the case where the evader acts step by step. (ii) Training the pursuer against an evader who chooses the shortest path, a worst-case scenario for the pursuer, enhances the robustness of the pursuer's policy. (iii) Though it is a simplification, the problem setting remains highly complex due to the multiple exits and diverse players' initial conditions, enlarging the task space for the pursuer.

In summary, given a graph G with the specific set of exit nodes $\overline{V}$, the initial locations of the pursuer and evader $(l_0^{\mathbf{P}}, l_0^e)$, and the predetermined number of game steps T, we can define a specific PEG as $\mathcal{G} = (G, \overline{V}, l_0^p, l_0^e, T)$. The game ends when the termination conditions are met. The termination conditions include three cases: (i) the pursuer catches the evader within T, i.e., $l_t^e \in l_t^p, t \leq T$; (ii) the evader escapes from an exit node within T, i.e., $l_t^e \in \overline{V}, t \leq T$; (iii) the game reaches the predetermined time step T. Let $t' \leq T$ be the time step that the game is terminated. Then, for all $t < t'$, $r_t^p = r_t^e = 0$. For $t = t'$, in cases (i) and (iii), the pursuer receives a reward $r_t^p = 1$ while the evader incurs a penalty $r_t^e = -1$. In case (ii), the evader gains a reward $r_t^e = 1$, and the pursuer suffers a loss $r_t^p = -1$.

5.2.2 Generalizability Issue

Although PSRO has been successfully applied to solve pursuit-evasion games, such as introduced in Chapter 4, unfortunately, previous works typically focus on solving a specific pursuit-evasion game with predetermined initial conditions which are not always fixed in real-world scenarios. As we introduced before, a pursuit-evasion

game can be represented by $\mathcal{G} = (G, \overline{V}, l_0^p, l_0^e, T)$. The initial conditions of the game may change in real-world scenarios, as described in the following.

(i) The initial locations of the pursuers and the evader (l_0^p, l_0^e) are not always fixed since attacks (thieves, crimes, terrorists) can occur at any time and location in a city; (ii) The locations of the exit nodes V' may change due to temporary closures and openings; (iii) The time horizon T might vary, as the time required to pursue the evader is not always the same. When any of these initial conditions change, the pursuit-evasion game adapts accordingly. As a consequence, existing algorithms can only solve the modified pursuit-evasion game from scratch, leading to significant time consumption and inefficiency. Even the algorithm presented in the previous Chapter – PSRO with pre-trained pursuer strategies – still suffers from such an issue since the base policy is pre-trained under the premise that the initial conditions of the pursuit-evasion game are fixed. In other words, a new base policy must be pre-trained from scratch for the modified pursuit-evasion game since the original base policy may not be a good starting point for the pursuer's BR policy in the modified game (even worse than a randomly initialized BR policy).

In this section, we aim to address this issue by developing a generalist pursuer capable of learning and adapting to different pursuit-evasion games with varying initial conditions without the need to restart the training process from the beginning.

5.3 Related Works

Pursuit-evasion games (PEGs) have been extensively applied to model various real-world problems such as security and robotics [55, 83–88]. To efficiently solve PEGs and different variants, many algorithms such as value iteration [55] and incremental strategy generation [16, 17] have been introduced. Nonetheless, these methods encounter scalability issues as they typically rely on linear programming. On the other hand, PEG can be viewed as a particular type of two-player imperfect-information extensive-form game (IIEFG). Thus, the algorithms used for solving large-scale IIEFGs, such as counterfactual regret minimization (CFR) [12] and Policy-Space Response Oracles (PSRO) [13], have been applied to tackle large-scale PEGs [57, 80]. However, when solving large-scale PEGs using PSRO, there exist significant computational challenges as it involves computing the best response

strategy multiple times. To mitigate this issue, we have proposed to integrate the pre-training and fine-tuning paradigm into the PSRO framework to improve its scalability, introduced in Chapter 4. Despite the success, all these algorithms are tailored to solve a specific PEG with predetermined initial conditions. When these conditions change, existing algorithms must recompute the NE strategy from scratch (about one to two hours for a PEG on a 10×10 grid map [56]), which hinders their real-world applicability[3]. To address this limitation, we propose GRASPER, which uses the PSRO algorithm to compute the NE strategy and can generate different pursuers' base strategies for different pursuit-evasion games based on their initial conditions without recomputing the NE strategy from scratch.

The generalizability of algorithms and models over different games has also gained increasing attention and remarkable progress has been achieved in recent research. Neural equilibrium approximators that directly predict the equilibrium strategy from game payoffs in normal-form games (NFGs) have been theoretically proven PAC learnable [90, 91] and can generalize to different games with desirable solution quality [90–93]. However, it remains under-explored when going beyond NFGs. To this end, we make the first attempt to consider the generalization problem in the domain of PEGs, a special type of imperfect-information extensive-form game that has a wide range of real-world applications [55, 88] and is far more complicated than NFGs. Specifically, we propose a novel algorithmic framework that can efficiently solve different pursuit-evasion games with varying initial conditions and demonstrate strong generalization ability through extensive experiments.

Our work is also related to self-supervised graph learning since the pursuit-evasion game is played on a graph. Recent works have shown that generative self-supervised learning [94] can be applied to graph learning and outperform contrastive methods which require complex training strategies [95, 96], high-quality data augmentation [97], and negative samples that are often challenging to construct from graphs [98]. Therefor, we employ the state-of-the-art, GraphMAE [27], to learn a good representation of a pursuit-evasion game with the given initial conditions.

Multi-task learning [69, 70] has been applied to various domains including natural language processing (NLP) [99], computer vision (CV) [100], and reinforcement learning (multi-task RL) [71–73, 101]. Due to its strong generalizability, we employ

[3]Note that classical heuristic algorithms such as Dijkstra are also less applicable owing to this reason. Moreover, it is less meaningful to assume that there is at least one pursuer at each exit as the pursuer's resources are typically limited [89].

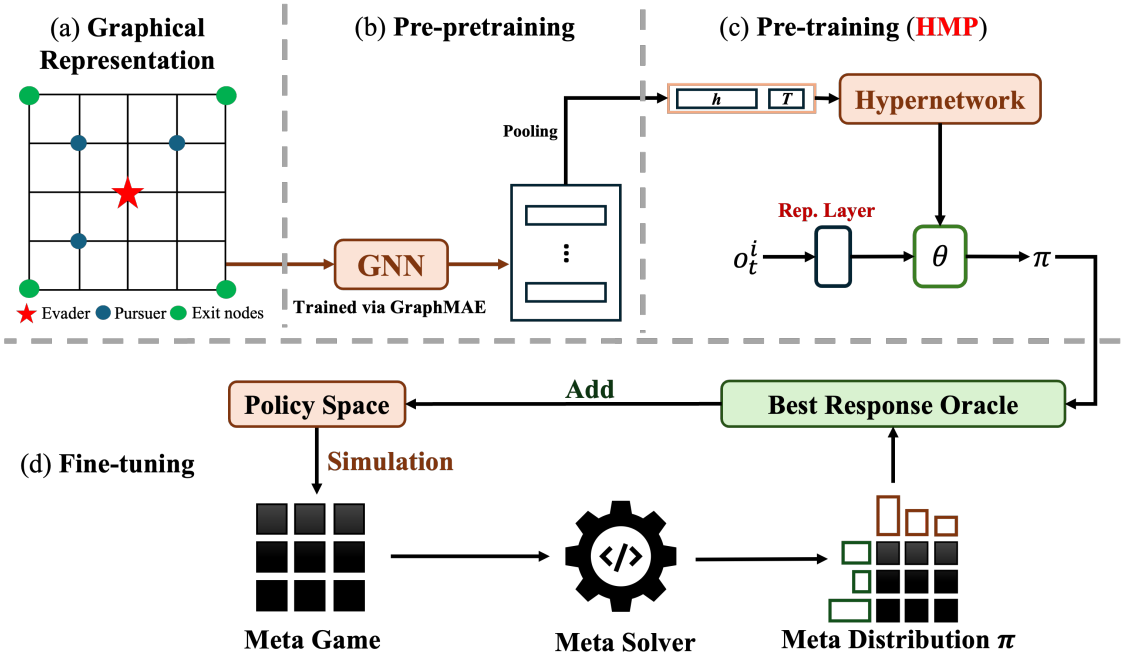


FIGURE 5.2: Architecture and training pipeline of GRASPER.

multi-task RL during our pre-training process, enabling the pre-trained policy to be quickly fine-tuned for efficient policy development in new tasks.

5.4 GRASPER

In this section, we introduce our proposed method, GRASPER, illustrated in Figure 5.2. We first present the architecture of GRASPER including several innovative components, and then we introduce the training pipeline which consists of three stages to efficiently train the neural networks within GRASPER framework.

5.4.1 Architecture

5.4.1.1 Graphical Representations of PEGs

To generate the pursuer's strategy for a specific pursuit-evasion game, we propose to take the initial conditions of the specific pursuit-evasion game as an input of a neural network. To this end, we encode the initial conditions of a pursuit-evasion game except for T into a graph (Figure 5.2(a)). The predefined number of game

steps T can be directly fed into the neural network since it is just a number. Specifically, given a PEG $\mathcal{G} = (G, \overline{V}, l_0^p, l_0^e, T)$, these initial conditions $\overline{V}, l_0^p$ and l_0^e can be encoded into the graph G by associating each node within the graph with a vector consisting of the following parts: (i) a binary bit $\{0, 1\}$ where 1 indicates that the node is an exit, (ii) a binary bit $\{0, 1\}$ where 1 signifies that the evader's initial location is this node, (iii) the number of pursuers on this node $\{0, \ldots, n\}$ (the total number of pursuers across all nodes equals n), and (iv) additional information regarding the topology of the graph, such as the degree of the node. It provides a universal representation of any PEG with any initial condition.

5.4.1.2 Game-conditional Base Policies Generation

After representing a pursuit-evasion game as a graph, it is natural to leverage a Graph Neural Network (GNN) to encode the pursuit-evasion game with the given initial conditions into a hidden vector. As shown in Figure 5.2(b), we first feed the graphical representation of the initial conditions into the GNN and get the representations of all the nodes of the graph. Then, we use a pooling operation to integrate all the node representations into a hidden vector which will be concatenated with the time horizon T to get the final representation of the PEG. Next, to generate a base policy conditional on the PEG, we introduce a hypernetwork [102], a neural network that takes the final representation of the PEG as input and outputs the parameters (weights and biases) of the policy network (Figure 5.2(c)). Finally, the base policy network serves as a starting point for the training of the pursuer's best response policy in each iteration of the PSRO algorithm (Figure 5.2(d)).

5.4.1.3 Observation Representation Layer

As described earlier, the pursuer's policy is a mapping that associates each observation with a probability distribution over the available action set. Notably, an observation consists of the positions of both players. Representing these positions by mere index numbers of vertices in the graph does not provide much useful information for training since the index numbers are meaningless. Therefore, we intend to seek a more compact and meaningful representation of these observations. Previous works [56, 57] propose to train a node embedding model for this purpose. Unfortunately, such a model is often tailored and trained for a specific

graph, limiting its generalizability to other graphs. This lack of generalizability makes this method unsuitable for our problem. To address this issue, we adopt a representation layer to encode the pursuer's observations, which is an approach that is not limited to a specific graph.

As given in Section 5.2.1, the observation of the agent p_i within the pursuer team, $o_t^{p_i} = (l_t^{\mathbf{P}}, l_t^e, p_i, t)$ includes three parts: the players' current locations $(l_t^{\mathbf{P}}, l_t^e)$, the id number p_i, and the current time step t. Thus, the representation layer consists of three components, each of which is a "*torch.nn.Embedding*" which has been extensively used to encode an integer to a compact representation. The outputs of these three components are concatenated to obtain the representation of the pursuer's observation. This representation layer will be trained jointly with the hypernetwork during the pre-training process.

The strong generalization ability of GRASPER benefits from several designs of our architecture. First, the graphical representation offers a universal representation of any pursuit-evasion game, regardless of the graph's topology. Second, GNN can encode different pursuit-evasion games into fixed-size hidden vectors, which can be directly fed into the hypernetwork (otherwise, additional techniques are required if the sizes of the hidden vectors are varied). Finally, the hypernetwork is designed to generate a specialized policy tailored to a given pursuit-evasion game.

5.4.2 Training Pipeline

Now we introduce the training pipeline of GRASPER, which involves three stages: *pre-pretraining*, *pre-training*, and *fine-tuning*. The rationale behind the three-stage training pipeline is to progressively enhance the ability of the neural networks to generate adaptive policies for PEGs with different initial conditions. As illustrated in Figure 5.2, the pre-pretraining stage is designed to train the GNN to produce game representations using self-supervised graph learning techniques. The goal is to extract meaningful features from the graphical representation of PEGs to facilitate downstream tasks. In the pre-training stage, the hypernetwork is trained using heuristic-guided multi-task learning to generate base models tailored to specific PEGs. Finally, the fine-tuning stage refines these base policy models to compute the best response strategies efficiently within the PSRO framework, significantly improving the computational efficiency of best response strategy computations.

Before delving into the specifics, we first describe how to generate the training set. The training set $\mathcal{D}$ should consist of different pursuit-evasion games for training. To this end, we generate the training set by randomizing the initial conditions, denoted by $(G, \overline{V}, l_0^{\mathrm{p}}, l_0^e, T)$. However, this approach may yield certain games that lack meaningful training value. For example, when the evader's initial location is in such proximity to the exit nodes that the pursuer becomes incapable of capturing the evader regardless of its movements. To exclude these trivial cases, we introduce a filter condition when generating the training set – the shortest path from the evader's initial location to any exit nodes must exceed a predetermined number of game steps.

5.4.2.1 Stage I: Pre-pretraining

As the hypernetwork takes a feature vector as input, we first use a GNN to encode the graphical representation of the specific PEG into a fixed-size hidden vector. As the GNN is solely employed to derive the effective representation from the PEG's graphical interpretation, we introduce a pre-pretraining stage (Figure 5.2(b)) to pre-train the GNN before the actual pre-training stage. This approach is more efficient compared to jointly training the GNN and hypernetwork in the pre-training stage. Specifically, for each game in the training set $\mathcal{G} \in \mathcal{D}$, let $\boldsymbol{A}_{\mathcal{G}}$ and $\boldsymbol{X}_{\mathcal{G}}$ denote the adjacency matrix and feature matrix of the underlying graph, respectively. We first obtain the latent code matrix $\boldsymbol{H}_{\mathcal{G}} = f_{GNN}(\boldsymbol{X}_{\mathcal{G}}, \boldsymbol{A}_{\mathcal{G}})$ by the GNN and train the GNN via any self-supervised graph learning method (we use the recent SOTA method, GraphMAE [27]). The implementation follows GraphMAE. Here, we use the well-known Graph Convolutional Network (GCN) as the encoder. For the specific architecture of the GCN, the number of GCN layers is 2, the hidden size is 128, the output size of each node embedding is 32, the dropout rate is 0.5, and the batch size for training the GCN is 32. Then, we get the hidden vector by pooling the latent code matrix $\boldsymbol{h}_{\mathcal{G}} = pool(\boldsymbol{H}_{\mathcal{G}})$, which will be fed into the hypernetwork.

5.4.2.2 Stage II: Pre-training

Given a fixed evader's policy in a specific PEG, computing the pursuer's best response policy can be seen as an RL task. Thus, we can apply the multi-task RL

Algorithm 4: Pre-training Stage

1 Initialize GRASPER and the episode buffer $\mathcal{B} \leftarrow \emptyset$;
2 **for** *train epoch* $= 1, 2, \cdots$ **do**
3 Uniformly sample c_1 games from the training set $\mathcal{D}$;
4 **for** *each of the* c_1 *games* **do**
5 Randomly generate c_2 evader's policies ;
6 Generate pursuer's base policy $\sigma^{\mathbf{p}}_{\boldsymbol{\theta}} \leftarrow \text{GRASPER}(\mathcal{G})$;
7 **for** *each of the* c_2 *evader's policies* σ^e **do**
8 Sample data using σ^e, $\sigma^{\mathbf{p}}_{\boldsymbol{\theta}}$, and $\hat{\sigma}^{\mathbf{p}}$;
9 Add the data into the episode buffer $\mathcal{B}$;
10 Train the networks by optimizing the loss function $\mathcal{L}$;
11 Clear the episode buffer $\mathcal{B} \leftarrow \emptyset$;

algorithm to guide the pre-training process, which is shown in Algorithm 4. Different from the algorithm introduced in Chapter 4, in these RL tasks, except for the change in the evader's policy, the game's initial conditions also change. To obtain these RL tasks for pre-training, we first randomly sample c_1 games from the training set (Line 3), and then for each game, we randomly sample c_2 evader's policies (Line 5). Once the game and the evader's policy are fixed, the RL task is generated. During pre-training, for each game, we first feed the hidden vector of the game (obtained by the trained GNN) and the time horizon into the hypernetwork to generate the pursuer's base policy, and then for each evader's policy, we collect the training data using the pursuer's base policy (accompany by the representation layer) into the episode buffer. Finally, we train the hypernetwork and the representation layer jointly based on the episode buffer (Lines 6-9). Following the algorithm proposed in Chapter 4, to deal with the large action space caused by multiple pursuer agents, we employ MAPPO [76] as the underlying RL algorithm.

However, we found that simply applying the MAPPO under the multi-task learning framework can result in low efficiency due to random exploration in the environment. To clarify, consider the example illustrated in Figure 5.3, which shows the need for a more efficient pre-training method. Assume that the evader's policy is to take the shortest path to one of the exits (denoted by the red path). If the pursuer explores the environment randomly (the orange path), it will probably lose the game and then receive a negative reward. This situation can occur frequently at the beginning of the pre-training process because the pursuer's initial policy is

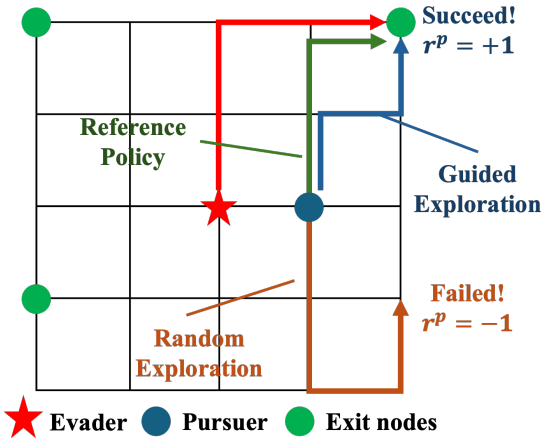


FIGURE 5.3: Illustration of heuristic-guided multi-task pre-training.

invariably random. To mitigate this exploration inefficiency[4], we propose a novel learning scheme – heuristic-guided multi-task pre-training (HMP).

Note that in the RL tasks used for pre-training, we can acquire the evader's policy, which can be used to guide the exploration of the pursuer's policy. Specifically, given the exit node chosen by the evader's policy σ^e, we first induce a reference policy $\hat{\sigma}^{\mathbf{p}}$ (represented by the green path) for the pursuer using heuristic methods such as the Dijkstra algorithm. Then, apart from the actions sampled by the generated policy $\sigma^{\mathbf{p}}_{\boldsymbol{\theta}}$ (Line 6), we also sample the reference actions using the reference policy $\hat{\sigma}^{\mathbf{p}}$ and add the data to the training buffer (Lines 8-9). Let $\mathcal{L}(\boldsymbol{\theta})$ denote the original loss function for training the actor within the MAPPO algorithm. The HMP is implemented by introducing an additional loss into the original loss function: $\mathcal{L} = \mathcal{L}(\boldsymbol{\theta}) + \alpha \mathrm{KL}(\sigma^{\mathbf{p}} \| \hat{\sigma}^{\mathbf{p}})$ where $\alpha \in [0, 1]$ controls the weight of the guidance of the reference policy and KL represents the Kullback–Leibler divergence (for the reference policy $\hat{\sigma}^{\mathbf{p}}$, the action probability distribution is obtained by setting the probability of the reference action to 1 while all others to 0).

5.4.2.3 Stage III: Fine-tuning

In this stage, we integrate the pre-trained pursuer policy into the PSRO framework, as shown in Algorithm 5. The pursuer's policy $\sigma^{\mathbf{p}}_0$ is initialized using the output

[4]Notice that many exploration methods in RL such as RND [103] typically encourage the policy to explore novel states of the environment, which are different from our design where random exploration is less favored.

Algorithm 5: Fine-tuning Stage

1 **Require**: GRASPER, pursuit-evasion game $\mathcal{G}$;
2 Initialize $S_0^{\mathbf{p}} = \{\sigma_0^{\mathbf{p}} \leftarrow \text{GRASPER}(\mathcal{G})\}$, $S_0^e = \{\sigma_0^e\}$;
3 Initialize mete game U_0, meta distribution ρ ;
4 **for** *epoch* $k = 1, 2, \cdots K$ **do**
5 Compute the evader's best response policy σ_k^e against $\rho_{k-1}^{\mathbf{p}}$;
6 Initialize the pursuer's best response policy $\sigma_k^{\mathbf{p}} \leftarrow \sigma_0^{\mathbf{p}}$;
7 Train $\sigma_k^{\mathbf{p}}$ against ρ_{k-1}^e for few episodes ;
8 Expansion: $S_k^{\mathbf{p}} = S_{k-1}^{\mathbf{p}} \cup \{\sigma_k^{\mathbf{p}}\}$, $S_k^e = S_{k-1}^e \cup \{\sigma_k^e\}$;
9 Update meta-game matrix U_k through simulation ;
10 Compute $\rho_k^{\mathbf{p}}$ and ρ_k^e using a meta-solver on U_k ;
11 **Return**: $S_K^{\mathbf{p}}$, S_K^e, $\rho_K^{\mathbf{p}}$, ρ_K^e

neural network from the pre-trained hypernetwork, which takes the graphical representation of the specific PEG as an input. Simultaneously, the evader's policy σ_0^e is randomly initialized (Line 2). Then we follow the standard PSRO framework. In each iteration k, the best response (BR) policies for both players, $\sigma_k^{\mathbf{p}}$ and σ_k^e, are computed using their respective best response oracles (Lines 5-7). These best response strategies are then added to the policy sets $S_k^{\mathbf{p}}$ and S_k^e (Line 8), and the meta-game matrix U_k is updated through simulation (Line 9). Finally, the meta distribution $(\rho_k^{\mathbf{p}}, \rho^e)$ is computed using any meta-solver (Line 10).

The best response oracles for both players are the important components of the PSRO algorithm. The evader's best response oracle used here is the same as used in the algorithm introduced in Chapter 4. The key difference between our fine-tuning stage and the standard PSRO algorithm lies in the training of the pursuer's best response policy. Specifically, given the pre-trained policy $\sigma_0^{\mathbf{p}}$ conditional to the initial conditions, we can use it as the starting point for the computation of the pursuer's BR policy (Line 6), rather than training from scratch. This allows us to simply fine-tune the pre-trained policy $\sigma_0^{\mathbf{p}}$ over a few episodes (Line 7) to quickly obtain the BR policy, significantly enhancing the learning efficiency.

5.5 Empirical Evaluation

In this section, we perform experiments to evaluate the performance of GRASPER and the effectiveness of different components[5].

[5]The code is available at https://github.com/IpadLi/Grasper.

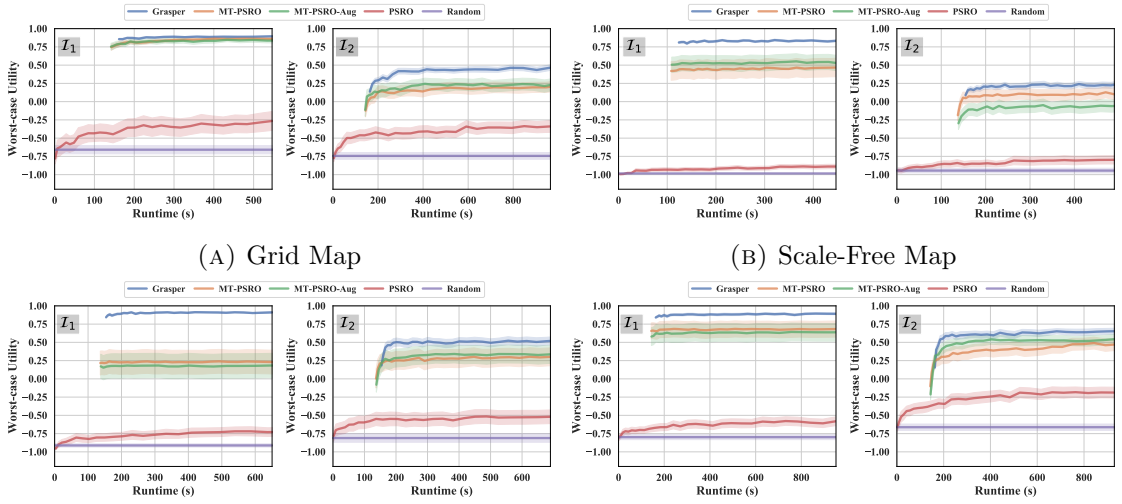
5.5.1 Experimental Setting

Hyperparameters. The number of agents within the pursuer team is $n = 5$, the number of exit nodes is 8, the number of game steps T is set to be $6 \leq T \leq 10$, and the number of pre-training episodes is 20 million (20M). For the PSRO algorithm, the number of episodes used for training the best response is 10. We conduct experiments on four maps: the grid map with size 10×10, the scale-free graph with 300 nodes, the Singapore map [57] with 372 nodes, and the Scotland-Yard map [104] with 200 nodes. To simulate the situation where a road might be temporally blocked due to congestion or traffic accidents, we set the probability of an edge between two nodes to 0.8 for the grid map, 0.9 for the Singapore map, and 1.0 (i.e., no congestion) for the other two maps. All experiments are performed on a machine with a 24-core Intel(R) Core(TM) i9-12900K and NVIDIA RTX A4000.

Worst-case Utility. Since the PEG is a zero-sum game, we use the pursuer's worst-case utility to measure the pursuer's policy, as used in previous Chapters.

Training and Test Sets. (1) We generate $|\mathcal{D}| = 1000$ games as the training set $\mathcal{D}$. During the generation process, the minimum length of the evader's shortest path is set to 6 for the grid map and 5 for other maps. (2) We create two test sets, $\mathcal{D}_1$ and $\mathcal{D}_2$, each containing 30 games. (i) $\mathcal{D}_1$ includes the games sampled from the training set $\mathcal{D}_1 \subset \mathcal{D}$ (in-distribution test). (ii) $\mathcal{D}_2$ contains the games distinct from the training set $\mathcal{D}_2 \cap \mathcal{D} = \emptyset$ (out-of-distribution test). To avoid trivial cases (the games that are either too difficult or too simple for the pursuers), we constrain the zero-shot performance of GRASPER (i.e., the worst-case utility of the generated policy without fine-tuning) within the range: $[0.8, 0.9]$ for $\mathcal{D}_1$ and $[0.1, 0.2]$ for $\mathcal{D}_2$.

Baselines. (i) Multi-task PSRO (MT-PSRO): the state-of-the-art approach adapted from the algorithm introduced in Chapter 4, which also uses the observation representation layer and HMP. (ii) MT-PSRO with augmentation (MT-PSRO-Aug): the hidden vector obtained from the pre-trained GNN and the time horizon are concatenated to the output of the observation representation layer. (iii) PSRO: the standard PSRO method. (iv) Random: the pursuer randomly selects actions.



(A) Grid Map (B) Scale-Free Map

(C) Singapore Map (D) Scotland-Yard Map

FIGURE 5.4: Experimental results (The shaded area shows the standard error).

5.5.2 Experimental Results

The experimental results are summarized in Figure 5.4. The x-axis is the running time. For the purpose of a fair comparison, apart from the running time of the fine-tuning stage (i.e., the PSRO procedure), we also include the running time used for pre-pretraining and pre-training stages (called the pre-training time for convenience). Since the games in the training set are uniformly randomly sampled during the pre-training process, we approximate the pre-training time of each game by averaging the total pre-training time over the size of the training set. Then, for each testing game, we add the pre-training time to the running time[6]. Note that the horizontal gap between 0 and the start of the line represents the pre-training time. From these results, we can draw several conclusions as follows.

(i) Given a fixed number of episodes for the fine-tuning process, GRASPER can start from and converge to a higher average worst-case utility than the baselines, although it takes a certain pre-training time, demonstrating the effectiveness of the pre-pretraining and pre-training in accelerating the PSRO procedure. Note that MT-PSRO and MT-PSRO-Aug also employ pre-pretraining or pre-training, but they perform worse than GRASPER, showcasing the superiority of GRASPER.

[6]Note that the amortized pre-training time for GRASPER, MT-PSRO, and MT-PSRO-Aug is similar as the *pre-pretraining time* is very short as shown in Table 5.2.

TABLE 5.1: Ablation studies. The results are obtained in the grid map. ✓ means the module is used.

	HMP	Observation Representation Layer	Utility
$\mathcal{D}_1$	✓	✓	$\mathbf{0.90 \pm 0.01}$
	✓		-0.54 ± 0.06
		✓	-0.05 ± 0.17
			-0.52 ± 0.08
$\mathcal{D}_2$	✓	✓	$\mathbf{0.45 \pm 0.04}$
	✓		-0.60 ± 0.06
		✓	-0.64 ± 0.11
			-0.63 ± 0.06

(ii) For a fair comparison, MT-PSRO-Aug also integrates information about the initial conditions of the pursuit-evasion games. The results clearly show the necessity of the hypernetwork in GRASPER. This can be also partly verified by comparing MT-PSRO and MT-PSRO-Aug where their performance is comparable, meaning that naively integrating the information about the initial conditions does not bring much benefit and novel designs are necessary.

(iii) An interesting result is that even on the test set $\mathcal{D}_1$ (in-distribution test), MT-PSRO and MT-PSRO-Aug, the strongest baselines, perform worse on all the other maps than on the grid map. We hypothesize the reason is that the other maps are more heterogeneous than the grid map. For example, the degree of the nodes varies from 1 to 16 in the Singapore map while it remains between 2 to 4 in the grid map. Therefore, the games generated on the Singapore map share much less similarity. In this sense, the information about the initial conditions is particularly important when solving different pursuit-evasion games.

(iv) In all cases, the performance of GRASPER is much more stable than the baselines (smaller standard error) as GRASPER can generate distinct policies for different PEGs. In contrast, other baselines either entirely ignore or naively integrate the information about the initial conditions of the PEGs, which renders them hard to generalize to different PEGs, leading to larger performance variance.

(v) The results on the test set $\mathcal{D}_2$ show that GRASPER can solve unseen games, exhibiting stronger generalization ability than all baselines.

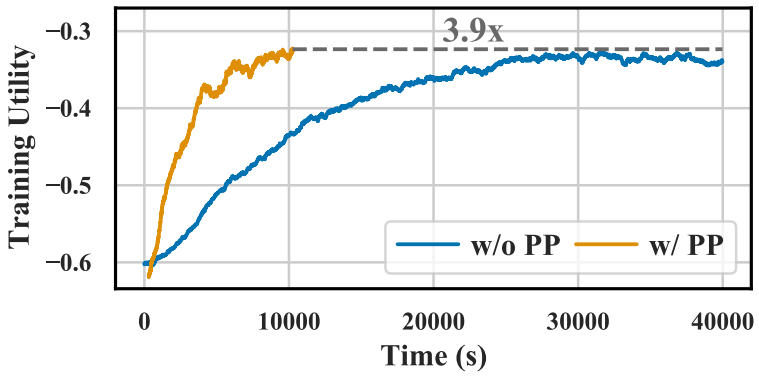


FIGURE 5.5: Pre-training curves.

5.5.3 Ablation Study

In this section, we perform ablation studies to show the effectiveness of different components, which further deepens the understanding of our proposed framework.

Effectiveness of Different Modules. First, we study the contribution of HMP and the observation representation layer to the performance of GRASPER, as shown in Table 5.1. The results show that we can get better performance (high worst-case utility and small standard error) only when combining the two components, meaning that both two components are indispensable for GRASPER.

Effectiveness of Pre-pretraining. Next, we investigate the effectiveness of the pre-pretraining stage in accelerating the whole training procedure of GRASPER. Since jointly training GNN and the other parts of GRASPER for 20M pre-training episodes requires a long running time, in this ablation study, we focus on the first 2M pre-training episodes and compare the running time of GRASPER with pre-pretraining (w/ PP) and without pre-pretraining (w/o PP). The training curves are shown in Figure 5.5, which shows that using pre-pretraining can significantly accelerate the training procedure (3.9 times faster than without using pre-pretraining).

The quantitative values of the running time of the pre-pretraining and pre-training are given in Table 5.2. As the pre-pretraining time (304.2 seconds) is much shorter than the pre-training time (9954.9 seconds), the curves of GRASPER, MT-PSRO, and MT-PSRO-Aug shown in Figure 5.4 start from a similar position in the x-axis.

Influence of Evader's Initial Location. We perform some experiments using GRASPER to provide some insights into the PEG. In Figure 5.6, we present the

TABLE 5.2: Running time (second).

	without Pre-pretraining	with Pre-pretraining
Pre-pretraining Stage	N/A	304.2
Pre-training Stage	39977.3	9954.9
Total	39977.3	10259.1 (**3.9x**)

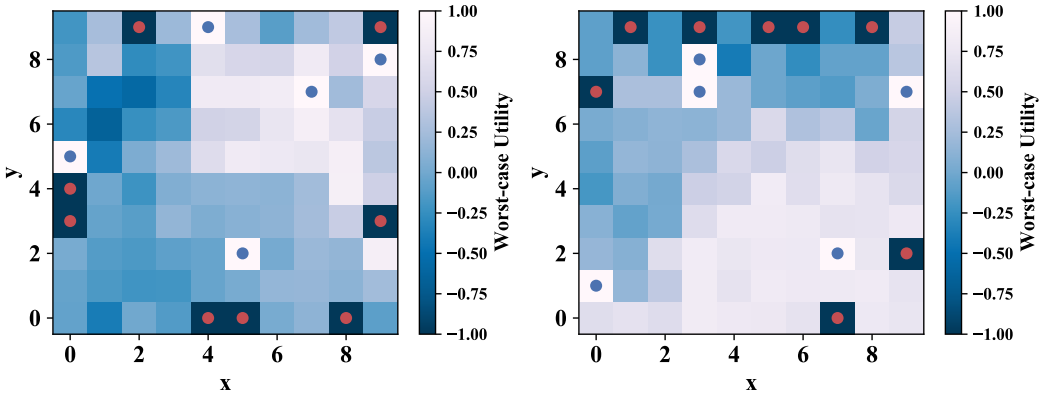


FIGURE 5.6: Zero-shot worst-case utility of the pursuer for each possible evader's initial location. Red dots are exits and blue dots are pursuers' initial locations.

pursuer's utility when the evader randomizes the initial location over the grid map. We found that in some areas the pursuers can have high utility. For example, in the top-right of the left figure, there are three pursuers and only one exit, which means it could be hard for the evader to escape. In the bottom-right of the right figure, as the pursuer's initial location is near the two exits, it could be easy for the pursuer to catch the evader, even though there is only one pursuer in this area. The results reflect the intuition that GRASPER can generate distinct policies for different games and hence, the performance is more stable.

5.6 Chapter Summary

In this Chapter, we investigate how to efficiently solve different PEGs with varying initial conditions. First, we propose a novel generalizable framework, GRASPER, which includes several critical components: (i) a GNN to encode a specific PEG into a hidden vector, (ii) a hypernetwork to generate the base policies for the pursuers conditional on the hidden vector and time horizon, (iii) an observation representation layer to encode the pursuers' observations into compact and meaningful

representations. Second, we introduce an efficient three-stage training method which includes: (i) a pre-pretraining stage that learns robust PEG representations through GraphMAE, (ii) a heuristic-guided multi-task pre-training stage that leverages a reference policy derived from heuristic methods such as Dijkstra to regularize pursuer policies, and (iii) a fine-tuning stage that utilizes PSRO to generate pursuer policies on designated PEGs. Finally, extensive experiments demonstrate the superiority of GRASPER over baselines in terms of solution quality and generalizability. To the best of our knowledge, this is the first attempt to consider the generalization problem in the domain of PEGs.

Part III

Offline Algorithm for IIEFGs

Chapter 6

Equilibrium Finding for IIEFGs via Offline Learning

6.1 Introduction

In this Chapter[1], we focus on solving imperfect-information extensive-form games under the offline setting. As we introduced in previous Chapters, imperfect-information extensive-form games (IIEFGs) provide a versatile framework for modeling the interactions between multiple players under stochastic and imperfect information settings [1]. The canonical solution concept is Nash Equilibrium (NE), where no player can increase his own utility by unilaterally deviating. There are various methods designed for solving extensive-form games, including linear programming [9], double-oracle algorithms [10], counterfactual regret minimization (CFR) [12], and policy-space response oracles (PSRO) [13]. These methods have been successfully applied to real-world large-sale EFGs, e.g., pursuit-evasion games [56, 57], poker games [7, 105, 106] and Stratego [107].

Despite the successes of existing equilibrium finding algorithms, they require continuous interaction with the game environment or an accurate simulator. For example, CFR-based algorithms necessitate traversing the game tree to compute regret values, and PSRO and its variants demand simulations within the game environment to compute the best response oracle and estimate the entries in the

[1]This Chapter is currently under review.

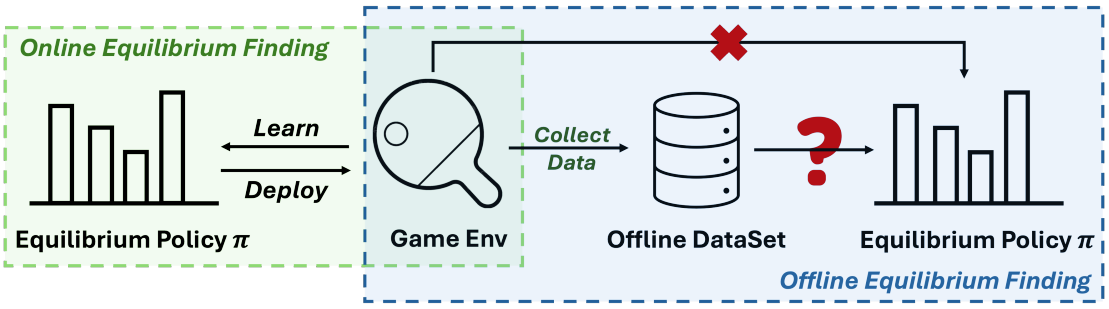


FIGURE 6.1: Comparison between online and offline equilibrium finding.

meta-game. We call this paradigm to compute NE as "*online equilibrium finding*". However, in many real-world applications, such as predicting financial market volatility [23], network intrusion detection [24], and automated negotiations [25], the immediate interaction with the environment may be impractical, or excessively costly. Figure 6.1 illustrates the comparison between online and offline equilibrium finding settings. To mitigate this issue, an accurate simulator can be built to replace the real environment. However, there are two main issues: i) the difficulties of building accurate simulators, especially for complicated EFGs such as sports games, and ii) the sample inefficiency in the complex simulator, as the equilibrium computation requires a large amount of data with high quality. Therefore, *offline learning* is a preferred option for equilibrium finding in real-world applications.

The offline learning paradigm has been successfully applied in the reinforcement learning area in addressing numerous real-world problems [108]. Offline reinforcement learning (Offline RL) approach utilizes existing datasets, eliminating the need for continuous online data collection, which may be costly and impractical. These algorithms fall into two categories: model-free and model-based. Model-free algorithms learn an optimal policy directly from the pre-collected datasets, examples include Constrained Q-learning (CQL) [109] and Best-Action Imitation Learning (BAIL) [110]. In contrast, model-based algorithms first learn a dynamic model from the offline dataset and then make planning based on the model, such as Model-based Offline Reinforcement Learning (MOReL) [111] and Model-based Offline Policy Optimization (MOPO) [112]. The success of these algorithms showcases the significant impact of the offline learning paradigm in advancing RL applications.

In recent years, several attempts have been made to formalize the offline learning paradigm in the context of games. StarCraft II Unplugged [113] provides a dataset of human game-plays in a two-player zero-sum symmetric game. Some

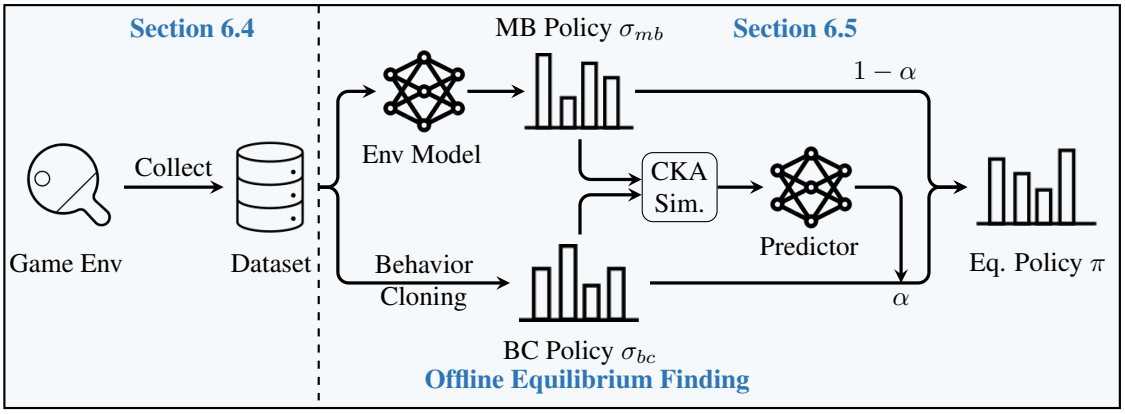


FIGURE 6.2: Our works in offline equilibrium finding setting.

previous works [114, 115] also explore the necessary properties of offline datasets for two-player zero-sum Markov games to successfully infer their NEs. However, these works mainly focus on solving Markov games, leaving a gap in the literature when it comes to solving imperfect-information extensive-form games under the offline setting. Furthermore, to our knowledge, there has been no study focusing specifically on multi-player games in an offline setting. More importantly, there is a notable absence of systematic definitions and research efforts aimed at formalizing offline learning within the context of games.

To address this gap, we propose the novel *offline equilibrium finding* (Offline EF) paradigm, in which equilibrium strategies are computed solely from offline datasets. However, there are several challenges for Offline EF. First, the absence of comprehensive benchmarking standards complicates the evaluation and comparison of offline algorithm performance. Without universally accepted benchmarks, it becomes difficult to objectively measure progress within the field. Second, accurately computing or approximating equilibrium strategies solely from offline datasets is inherently difficult. Specifically, data from just two action profiles are often insufficient for determining proximity to an equilibrium strategy, as equilibrium identification requires reference to all other potential action profiles [114]. Third, the quality and completeness of data within offline datasets can significantly impact the effectiveness of derived strategies. Offline datasets often fail to cover all possible game states, and this lack of comprehensive coverage can skew the algorithm's ability to generalize from the available data.

To address these challenges in the Offline EF paradigm, our contributions are threefold, as shown in Figure 6.2. i) We have created a collection of diverse offline

datasets, including random datasets, expert datasets, learning datasets, and hybrid datasets in different imperfect-information extensive-form games. ii) We propose the BOMB framework, which integrates a behavior cloning technique and a model-based method along with a novel parameter estimation method. The model-based method can incorporate any online EF algorithm into the offline context. iii) We provide a comprehensive theoretical analysis for our BOMB framework, offering performance guarantees under different datasets. Finally, we demonstrate the effectiveness of our BOMB framework in computing equilibrium strategies in an offline setting through extensive experiments on various offline datasets.

6.2 Problem Statement

To facilitate the widespread application of game theory, we extend the offline learning framework into imperfect-information extensive-form games and introduce the *offline equilibrium finding* paradigm, which focuses on learning equilibrium strategies solely from historical game-playing data. A formal definition of the offline equilibrium finding paradigm is provided as follows:

Definition 6.1 (Offline EF)**.** Given an offline dataset $\mathcal{D}$ of an IIEFG $\mathcal{G}$, generated by an unknown behavior strategy profile σ, the *offline equilibrium finding* paradigm is to deduce a strategy profile σ^* from the offline dataset $\mathcal{D}$ to achieving a minimal gap from the equilibrium strategy.

Building on the definition of the Offline EF paradigm, we can instantiate this paradigm by defining a metric for the gap from the equilibrium strategy, such as the NashConv for NE [2] and (C)CE Gap Sum for (C)CE [28] as introduced in Chapter 2. While Offline EF shares similarities with offline RL to some extent, it also presents distinct differences and unique challenges.

Firstly, unlike offline RL, which aims to compute an optimal strategy [108], the Offline EF paradigm seeks to achieve an equilibrium strategy. This objective necessitates an iterative process to calculate the best response strategy, introducing distinct complexities. Secondly, the Offline EF paradigm involves at least two players, making the game dynamics particularly sensitive to distribution shifts and other uncertainties, a stark contrast to offline RL. Thirdly, while in offline RL,

the data from two actions may suffice to determine which action is better, in the Offline EF paradigm, simply comparing the data of two action tuples is inadequate for identifying which tuple is closer to an equilibrium strategy, as equilibrium identification requires reference to other action tuples [114]. In the subsequent sections, we investigate Offline EF from the perspectives of datasets, theory, and methods.

6.3 Related Works

In this section, we discuss some existing algorithms related to the Offline EF paradigm, including opponent modeling, online equilibrium finding, and offline reinforcement learning. We then conclude by highlighting the limitations of these existing algorithms when applied to the Offline EF paradigm.

6.3.1 Opponent Modeling

Opponent Modeling (OM) is necessary for multi-agent settings where secondary agents with competing goals also adapt their strategies, yet it remains challenging because policies interact with each other and change [116]. One simple idea of opponent modeling is to build a model each time a new opponent or group of opponents is encountered [117]. However, it is infeasible to learn a new model every time. A better approach is to represent an opponent's policy with an embedding vector. Grover et al. [118] use a neural network as an encoder, taking the trajectory of one agent as input. Imitation learning and contrastive learning are also used to train the encoder. Then, the learned encoder can be combined with RL algorithms by feeding the generated representation into the policy and/or value network.

DRON [116] and DPIQN [119] are two algorithms based on DQN, which use a secondary network that takes observations as input and predicts opponents' actions. However, if the opponents can also learn, these methods become unstable. Therefore, it is necessary to take the learning process of opponents into account. Foerster et al. [120] propose a method named Learning with Opponent-Learning Awareness (LOLA), in which each agent shapes the anticipated learning of the other agents in the environment. Further, the opponents may still be learning continuously during execution. Al-Shedivat et al. [121] propose a method based

on a meta-policy gradient named Meta-MPG, which uses trajectories from current opponents to perform multiple meta-gradient steps and constructs a policy that favors updating the opponents. Meta-MAPG [122] extends Meta-MPG by including an additional term that accounts for the impact of the agent's current policy on the future policies of opponents, similar to LOLA. Yu et al. [123] propose model-based opponent modeling (MBOM), which employs the environment model to adapt to various opponents.

In the Offline EF paradigm, our goal is to compute the equilibrium strategy solely based on the offline dataset. Applying opponent modeling algorithms alone is not sufficient for the Offline EF paradigm since it only aims at computing the best response strategy instead of the equilibrium strategy.

6.3.2 Online Equilibrium Finding

In Chapter 2, we have introduced some algorithms in computing the equilibrium strategy of imperfect-information extensive-form games, including no-regret methods, which are derivatives of CFR [12], and incremental strategy space generation methods under the PSRO framework [13]. We call these algorithms "online equilibrium finding" (Online EF) algorithms since they all work in an online way, i.e., they need interaction with the game environment or an accurate simulator. Another special approach within the online EF method is the empirical game theoretic analysis (EGTA), an empirical methodology that bridges the gap between theoretical game analysis and practical simulations for strategic reasoning [124].

In EGTA, game models are iteratively extended through a process of generating new strategies based on learning from experience with prior strategies. The strategy exploration problem, which focuses on how to assemble an effective portfolio of policies for EGTA, is one of the most challenging issues [125]. Schvartzman and Wellman [126] deployed tabular RL as a best-response oracle in EGTA for strategy generation. They also addressed the general problem of strategy exploration in EGTA and investigated whether better options exist beyond best-responding to an equilibrium [127]. The investigation of strategy exploration advanced significantly with the PSRO framework [13], a flexible framework for iterative EGTA where, at each iteration, new strategies are generated through reinforcement learning. Note that when employing NE as the meta-strategy solver, PSRO reduces to the double

oracle (DO) algorithm [10]. In EGTA, a space of strategies is examined through simulation, which necessitates a simulator, and the policies are known in advance.

However, in the Offline EF paradigm, only an offline dataset is provided. Although EGTA utilizes game data, it still requires a simulator to evaluate a set of strategies. The successes of existing online EF algorithms mainly depend on the continuous interaction with the game environment or an accurate simulator. Therefore, these online EF algorithms cannot be directly applied to the Offline EF paradigm.

6.3.3 Offline Reinforcement Learning

Offline reinforcement learning (Offline RL) is a *data-driven* paradigm that learns exclusively from static datasets of previously collected interactions, making it feasible to extract policies from large training datasets [108]. This paradigm can be extremely valuable in settings where online interaction is impractical, either because data collection is expensive or dangerous (e.g., in robotics [128], education [129], healthcare [130], and autonomous driving [131]). Therefore, offline RL algorithms have a much broader range of applications than online RL and are particularly appealing for real-world applications [132]. Due to its attractive characteristics, there has been a lot of recent research in this area. We can divide the research in offline RL into two categories: model-free algorithms and model-based algorithms.

Model-free offline RL algorithms learn a good policy directly from the offline dataset. There are two types of these algorithms: actor-critic methods and imitation learning methods. Actor-critic algorithms focus on implementing policy regularization and value regularization based on existing RL algorithms. Haarnoja et al. [133] propose Soft Actor-Critic (SAC) by adding an entropy regularization term to the policy gradient objective, focusing mainly on policy regularization. For value regularization, an offline RL method named Constrained Q-Learning (CQL) [109] learns a lower bound of the true Q-function by adding value regularization terms to its objective. Another line of model-free offline RL research is imitation learning, which mimics the behavior policy based on the offline dataset. Chen et al. [110] propose a method named Best-Action Imitation Learning (BAIL), which fits a value function and then uses it to select the best actions. Meanwhile, Siegel et al. [134] propose a method that learns an Advantage-weighted Behavior Model (ABM) and uses it as a prior in performing Maximum a-posteriori Policy

Optimization (MPO) [135]. This method consists of multiple iterations of policy evaluation and prior learning until finally performing a policy improvement step using the learned prior before extracting the best possible policy.

Model-based offline RL algorithms rely on the offline dataset to learn a dynamics model or a trajectory distribution used for planning. The trajectory distribution induced by these models is used to determine the best set of actions to take at each given time step. Kidambi et al. [111] propose a method named Model-based Offline Reinforcement Learning (MOReL), which measures their model's epistemic uncertainty through an ensemble of dynamics models. Meanwhile, Yu et al. [112] propose another method named Model-based Offline Policy Optimization (MOPO), which uses the maximum prediction uncertainty from an ensemble of models. Concurrently, Matsushima et al. [136] propose the Behavior REgularized Model ENsemble (BREMEN) method, which learns an ensemble of models of the behavior MDP, as opposed to a pessimistic MDP. In addition, it implicitly constrains the policy to be close to the behavior policy through trust-region policy updates. More recently, Yu et al. [137] proposed a method named Conservative Offline Model-Based Policy Optimization (COMBO), a model-based version of CQL. The advantage of COMBO over MOReL and MOPO is that it removes the need for uncertainty quantification in model-based approaches, which is challenging and often unreliable.

Certainly, offline RL algorithms do not need access to the game environment while computing the optimal strategy for one player. However, maximizing the utility of each player independently is neither sufficient nor practical for computing an equilibrium strategy because the utility of one player depends not only on its action but also on the actions of other players. If we use the offline RL algorithm to learn the best strategy for each player in the game, it may lead to significant losses, as the computed best strategies for players might be exploitable. Other players may play their best response strategies against the computed best strategy instead of their behavior strategies in the offline dataset. Therefore, offline RL is not adequate for computing the equilibrium strategy of the game, since it can only learn the best strategy for one agent independently. These limitations highlight that offline RL algorithms cannot be directly applied to the Offline EF paradigm, and our experimental results empirically verify this claim.

Finally, for a clearer comparison of these existing algorithms, we have summarized the issues of these discussed existing algorithms in Table 6.1.

Methods	Work without Environment	Converge to Equilibrium
OM	✗	✗
Online EF	✗	✓
Offline RL	✓	✗

TABLE 6.1: Issues of existing algorithms.

6.4 Offline Datasets

As delineated in the definition of the Offline EF paradigm, the offline dataset plays a pivotal role. However, there are no publicly available datasets specifically tailored for the Offline EF paradigm. Typically, the offline dataset is gathered by unknown strategies deployed in real-world scenarios. Because of this, a solitary dataset is inadequate for effectively evaluating the performance of algorithms in the Offline EF paradigm, as it may introduce bias when used alone for evaluation. Consequently, a suitable benchmark dataset for the Offline EF paradigm should consist of a diverse set of datasets that accurately reflect real-world scenarios. Generating such offline datasets presents a considerable challenge. The offline datasets should contain meaningful information and high diversity, rather than merely consisting of randomly generated datasets, despite such datasets potentially displaying high diversity. To this end, we propose several collection methods for creating offline datasets that serve as the foundation for the Offline EF paradigm.

6.4.1 Data Format

Before delving into the methods of dataset collection, it is essential to outline the data formats of the Offline EF dataset for imperfect-information extensive-form games. The Offline EF dataset can be represented by $\mathcal{D} = (s_t, a_t, s_{t+1}, u_{t+1}, d_{t+1})$, where s_t and s_{t+1} represent the game states at time step t and $t+1$ respectively from the game-level perspective. Specifically, s_t encompasses all relevant game information at time step t, which includes the information sets for each player and other game information GI out of the control of players, such as the results of a chance node $(I_1^t, I_2^t, ..., I_n^t, GI)$, the player who needs to act (p^t), and the set of available actions for the acting player $(A(I_{p^t}^t))$, i.e., $s_t = (I_1^t, I_2^t, ..., I_n^t, GI, p^t, A(I_{p^t}^t))$. Notable, p^t may represent a chance player c to include the game's stochastic events out of

all players' control. The utility for each player at time step $t+1$ is represented by $u_{t+1} = (u_1^{t+1}, u_2^{t+1}, ..., u_n^{t+1})$. Finally, the variable d_{t+1} indicates whether the game ends at state s_{t+1}, with a value of 1 if the game ends and 0 otherwise.

To better introduce the format of our Offline EF dataset, we provide an example to show the composition of the offline dataset. According to the data format introduced before, the data point would be $(s_t, a_t, s_{t+1}, u_{t+1}, d_{t+1})$ and $s_t = (I_1^t, I_2^t, ..., I_n^t, GI, p^t, A(I_{p^t}^t))$. Especially, if the s_{t+1} is the terminal state, i.e., $d_{t+1} = 1$, then we define the $p^{t+1} = -1$ to identify that no player needs to make decides at this state since the game has ends. Figure 6.3 shows one two-player imperfect-information extensive-form game $\mathcal{G}$. I_1 and I_2 are information set for Player 1 and Player 2, respectively. If an offline dataset $\mathcal{D}$ covers all state-action pairs, then $\mathcal{D}$ would include the following data points:

$$
\begin{aligned}
&((I_1, \emptyset, \emptyset, 1, \{a_1, a_2\}), a_1, (I_1a_1, I_2, \emptyset, 2, \{b_1, b_2\}), (0, 0), 0),\\
&((I_1, \emptyset, \emptyset, 1, \{a_1, a_2\}), a_2, I_1a_2, I_2, \emptyset, 2, \{b_1, b_2\}), (0, 0), 0),\\
&((I_1a_1, I_2, \emptyset, 2, \{b_1, b_2\}), b_1, (I_1a_1, I_2b_1, \emptyset, -1, \emptyset), (1, -1), 1),\\
&((I_1a_1, I_2, \emptyset, 2, \{b_1, b_2\}), b_2, (I_1a_1, I_2b_2, \emptyset, -1, \emptyset), (2, -2), 1),\\
&((I_1a_2, I_2, \emptyset, 2, \{b_1, b_2\}), b_1, (I_1a_2, I_2b_1, \emptyset, -1, \emptyset), (0, 0), 1),\\
&((I_1a_2, I_2, \emptyset, 2, \{b_1, b_2\}), b_2, (I_1a_2, I_2b_2, \emptyset, -1, \emptyset), (3, -3), 1).
\end{aligned}
$$

We can find that in states $(I_1^{t_2} = I_1a_1, I_2^{t_2} = I_2, GI = \emptyset, 2, \{b_1, b_2\})$ and $(I_1^{t_2} = I_1a_2, I_2^{t_2} = I_2, GI = \emptyset, 2, \{b_1, b_2\})$, the information set for Player 2 is the same, as shown in the game tree. Since our dataset is collected from the game-level perspective, we can still distinguish them through the game information of other players and the game information GI. Note that there is no chance node in this game, GI is always an empty set. If there is a chance node in a game, the results of the chance node would be recorded into game information GI within the game state and we can distinguish these game states through game information GI.

6.4.2 Collection Methods

Similar to the practices in the offline RL domain, datasets in the Offline EF area must be diverse to serve as effective benchmarks for developing and evaluating algorithms. Many benchmarks in the offline RL area, such as those discussed in

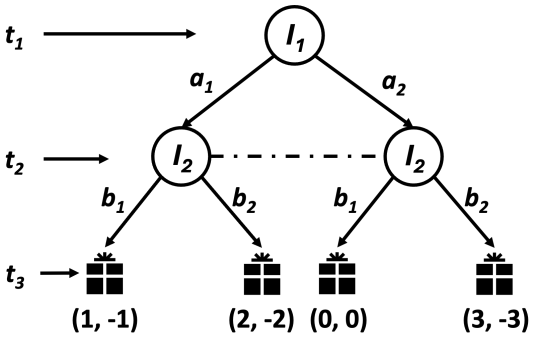


FIGURE 6.3: An IIEFG example.

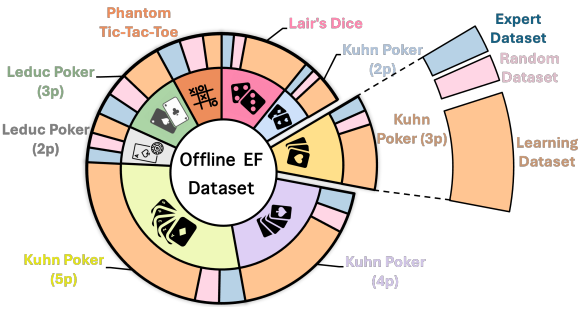


FIGURE 6.4: Offline EF dataset.

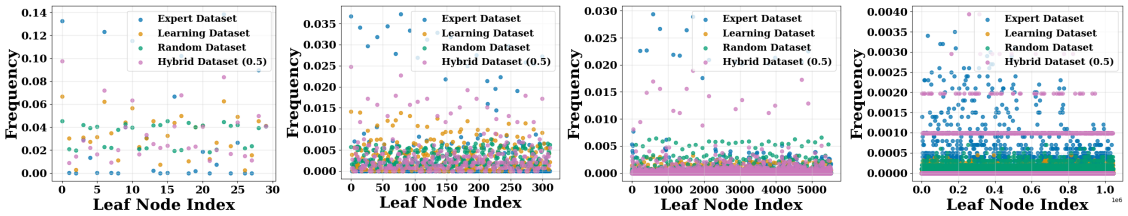
[138, 139], collected data from online RL training runs. Additionally, D4RL [140] incorporates a range of dataset collection methods inspired by real-world applications, including human demonstrations, exploratory agents, and hand-coded controllers. Inspired by these benchmarks in the offline RL area, we propose several methods for collecting offline datasets at different expert levels.

Random Dataset. The first one, referred to as the *random method*, involves each player adopting a uniform strategy and participating repeatedly in the game to collect data as the random dataset. The random method approach is inspired by the natural tendency for exploration and mimics the experience of a novice familiarizing themselves with the game for the first time.

Learning Dataset. The second method, the *learning-based method*, draws inspiration from the player skill improvement process. To mimic this process, we implement one of the existing equilibrium finding algorithms, such as CFR [12] or PSRO [13], storing intermediate game interactions to compile the learning dataset.

Expert Dataset. The final method is referred to as the *expert method*. The inspiration for this method comes from the notion that, when learning a game, we often benefit from observing the strategies of expert players. In this approach, each player follows an assigned equilibrium strategy and repeatedly engages in the game to collect the game-playing data to generate the expert dataset.

Hybrid Dataset. Additionally, to simulate more realistic scenarios and enhance dataset diversity, we propose a hybrid method that merges the random and expert datasets in different proportions. By doing so, we can create a more comprehensive and diverse collection of datasets, called hybrid datasets.



(A) Kuhn(2p) (B) Kuhn(3p) (C) Leduc(2p) (D) Leduc(3p)

FIGURE 6.5: Frequency of leaf node in different offline datasets.

6.4.3 Statistics and Visualization

We have developed a benchmark dataset for Offline EF, employing previously outlined collection methods on **eight** commonly used IIEFGs, as depicted in Figure 6.4. These games, which are prevalent in contemporary research on equilibrium finding problems, include two-player and multi-player variants of Kuhn poker and Leduc poker, Phantom Tic-Tac-Toe, and Liar's Dice. For each game, we have generated three distinct types of datasets, a random dataset, an expert dataset, and a learning dataset, following our data collection methods. The proportions of each dataset are visualized in Figure 6.4. Note that hybrid datasets can be easily obtained based on random and expert datasets. In total, our Offline EF dataset comprises approximately **3.8 million** data points, occupying about **11GB** of memory.

To validate the diversity of these collected Offline EF datasets and gain insights into them, we also introduce a visualization method to compare them. Firstly, we generate the game tree for the corresponding game. Subsequently, we traverse the game tree using depth-first search (DFS) [141] and assign an index to each leaf node based on the DFS results. Then, we count the frequency of each leaf node within the dataset. The reason why we do this is that each leaf node represents a unique sampled trajectory originating from the root node of the game tree. As a result, the frequency of leaf nodes can effectively capture the distribution of the dataset. Finally, these frequency data can be plotted to visualize. Figure 6.5 visualizes some datasets of some games. From these figures, we can find that in the random dataset, the frequency of leaf nodes is nearly uniform, whereas, in the expert dataset, the frequency of leaf nodes is uneven. The distribution of the learning dataset and the hybrid dataset falls between that of the expert dataset

and the random dataset. These observations confirm that the distribution of these datasets differs, thus validating the diversity of our proposed Offline EF datasets.

6.5 BOMB

In this section, we propose the offline algorithm, BOMB, which combines **B**ehavior cl**O**ning technique with **M**odel-**B**ased method for Offline EF paradigm. We first introduce the BOMB algorithm framework, then introduce the combination parameter estimation methods, and finally provide the theoretical analysis of BOMB.

6.5.1 Algorithm Framework

Drawing inspiration from the offline RL area, there are two possible approaches for computing equilibrium in an offline setting: model-free and model-based approaches. The model-free method learns the optimal policy *directly* from the offline dataset [110]. However, the unknown underlying behavior strategy of the offline dataset is likely not the equilibrium strategy, and equilibrium identification requires all other action tuples as references. Additionally, model-based offline RL methods [112] cannot be directly applied to the Offline EF paradigm due to their inherent reliance on the absence of strategic opponents in the environment. Therefore, neither a single model-free nor model-based method is sufficient for the Offline EF paradigm. To this end, we propose a novel framework – BOMB, which combines one model-free method and one model-based method.

Algorithm 6: BOMB Algorithm Framework

Input: An Offline EF dataset $\mathcal{D}$

1 Train policy σ_θ based on dataset $\mathcal{D}$ using behavior cloning technique ;
2 Train an environment model E_{θ_e} based on dataset $\mathcal{D}$;
3 Learn σ_{mb} policy using any existing EF algorithm on environment model E_{θ_e};
4 Select combination parameter α using a parameter estimation method ;
5 Combine σ_{mb} and σ_θ according to parameter α as final policy σ ;
6 **return** *Policy* σ.

Algorithm 6 shows the whole framework of the BOMB algorithm. Given an offline dataset $\mathcal{D}$, we first train a policy σ_θ based on the Offline EF dataset $\mathcal{D}$ using the behavior cloning technique (Line 1). The behavior cloning technique

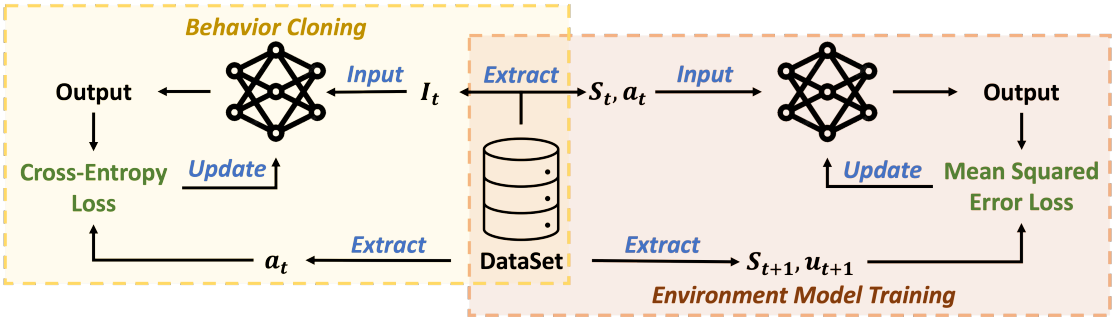


FIGURE 6.6: Processes of behavior cloning and training environment model

is a model-free approach that directly imitates the behavior policy used for generating the offline dataset and is commonly employed in offline RL [142]. Note $\mathcal{D} = (s_t, a_t, s_{t+1}, u_{t+1}, d_{t+1})$ and $s_t = (I_1^t, I_2^t, ..., I_n^t, GI, p^t, A(I_{p^t}^t))$. Since the policy network σ_θ is trained to mimic the behavior strategy, only the information set $I_{p^t}^t$ and the corresponding action a_t in dataset $\mathcal{D}$ are required for training. The cross-entropy loss is taken as the training loss, defined as

$$\mathcal{L}_{bc} = -\mathbb{E}_{(I_{p^t}^t, a_t)\sim\mathcal{D}}[a_t \cdot \log(\sigma(I_{p^t}^t; \theta))].$$

Figure 6.6 shows the training processes. After training the behavior cloning policy σ_θ, an environment model E_{θ_e} is trained based on dataset $\mathcal{D}$ and E_{θ_e} is used for learning the model-based policy σ_{mb} by any existing equilibrium finding algorithm (Lines 2-3). Inspired by model-based offline RL algorithms, where a dynamics model is trained to simulate the real environment [111, 112, 136], we train an environment model to imitate the real game environment. Specifically, we use the game state s_t and corresponding action a_t as inputs, with the subsequent game state s_{t+1}, reward u_{t+1} and the termination variable d_{t+1} serving as labels. Stochastic gradient descent (SGD) is employed as the optimizer for parameter updates, and the mean squared error (**MSE**) loss is used as the training loss, defined as

$$\mathcal{L}_{env} = \mathbb{E}_{(s_t, a_t, s_{t+1}, u_{t+1}, d_{t+1})\sim\mathcal{D}}[\mathbf{MSE}((s_{t+1}, u_{t+1}, d_{t+1}), E(s_t, a_t; \theta_e))].$$

A significant advantage of this model-based approach is adapting any existing equilibrium finding algorithm within the offline setting on the trained environment model. For example, for computing NE, we adapt PSRO [13] and Deep CFR [21] methods, referred to as MB-PSRO and MB-CFR, respectively. Additionally, we adapt the JPSRO method [29] (MB-JPSRO) for computing (C)CE. Essentially,

these adaptations involve replacing the real game environment with our trained environment model in executing these equilibrium finding algorithms, allowing them to function without requiring online interactions with the real game environment.

After learning the behavior cloning policy σ_θ and model-based policy σ_{mb}, we combine the trained behavior cloning (BC) policy and model-based (MB) policy. Let α denote the weight of the BC policy and $1-\alpha$ denote the weight of the MB policy. Specifically, the combination of these two policies occurs at the state level. For each state, the behavior cloning policy model and the model-based policy model independently generate their respective strategies, represented as distributions over the available actions for that state. These two distributions are then combined using the weight α, which determines the contribution of the behavior cloning strategy to the final strategy. We select α using a parameter estimation method, then apply it to integrate the BC and MB policies into the final policy σ (Lines 4-5).

6.5.2 Parameter Estimation Methods

Here, we introduce three estimation methods of the combination parameter α. The simplest method is randomly selecting a value from the interval $[0, 1]$ as the combination parameter α. This method can be implemented fully offline, making it suitable for scenarios where access to the real-world environment is not feasible. However, it lacks guarantees for yielding the most effective combined strategy.

The second method is the grid search method, in which we first initially preset a set of 11 candidate values for α, i.e., $\alpha = \{0, 0.1, ..., 1\}$, and these values are used to configure combined strategies, which are then tested in a real-world environment. The value of α that results in the smallest gap from the equilibrium strategy is selected as the optimal value. It can yield the best performance and similar techniques that determine offline parameters or fine-tune offline policies through online interactions are commonly employed in offline RL [143, 144]. As such, it is suitable for scenarios where access to the real-world environment is feasible. Although this method needs some online interactions, the number of required interactions is significantly less compared to computing the equilibrium strategy fully online.

To render our approach fully offline while still achieving optimal parameter values, we propose a learning-based method, depicted in Figure 6.7. In this method, a

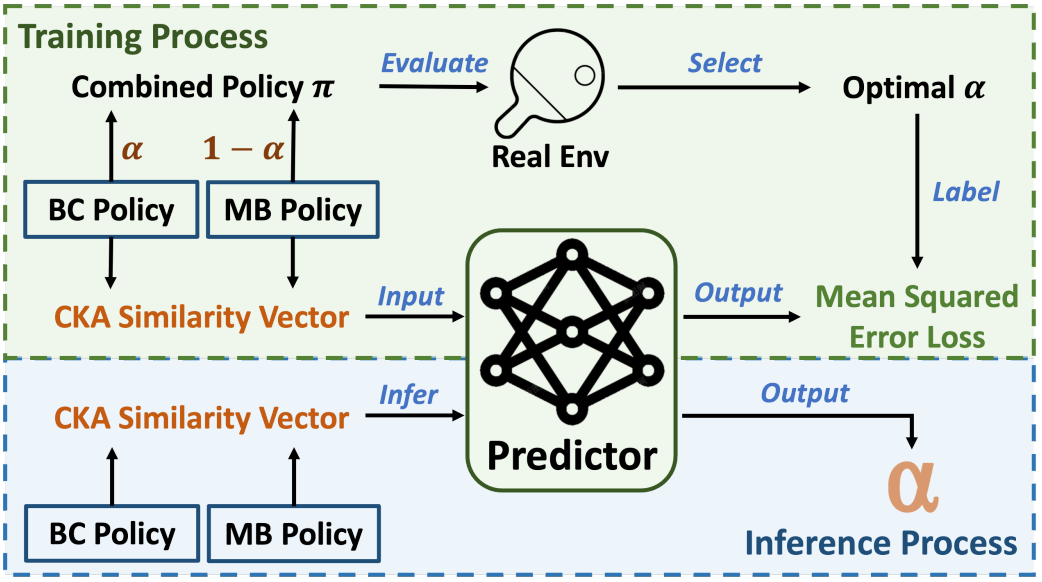


FIGURE 6.7: Learning-based estimation method.

predictor is trained to estimate the combination parameter α based on the difference between the BC and the MB policies. We first use the grid search method to get optimal parameter values as labels. The predictor takes the centered kernel alignment (CKA) [145] similarity vector between the BC and the MB policies as input and outputs the estimated parameter α. The predictor can be trained in one game where access to the real-world environment is feasible and be reused in similar games. Though the predictor can only provide an approximate optimal parameter value, it requires no further online interactions once trained.

6.5.3 Theoretical Analysis

In the offline RL area, dataset coverage over the optimal policy is sufficient for offline learning [146, 147]. However, we found a similar assumption that the dataset generated by the equilibrium strategy is not sufficient for computing equilibrium strategies in an offline manner. It can be verified by the counter-example shown in Figure 6.8. In this game, we can easily get the NE strategy, $\sigma^* = (\sigma_1^*, \sigma_2^*) = (\{I_1 : a_1\}, \{I_2 : b_2\})$. If

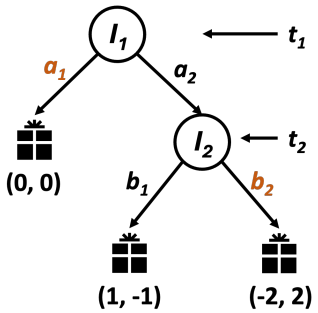


FIGURE 6.8: An IIEFG example.

we use this equilibrium strategy profile to generate the offline dataset $\mathcal{D}$, then $\mathcal{D}$ would only include the data point $((I_1^{t_1} = I_1, I_2^{t_1} = \emptyset, GI = \emptyset, 1, \{a_1, a_2\}), a_1, (I_1^{t_2} = I_1 a_1, I_2^{t_2} = \emptyset, GI = \emptyset, -1, \emptyset), (0, 0), 1)$. Clearly, the dataset $\mathcal{D}$ is not sufficient for computing the NE strategy since there is no information about Player 2. The intuition behind this is that in an offline RL setting, we can easily use the data of two actions to decide which action is better, whereas, in the Offline EF paradigm, we cannot use data from only two action pairs to know which action pair is closer to NE, because identifying NE requires other action pairs as inferences. Based on this analysis, Cui and Du [114] provide a minimal coverage assumption which is sufficient for computing an NE strategy in the two-player zero-sum Markov games, defined as follows,

Assumption 6.1. (Deterministic Unilateral Coverage) For all deterministic strategy σ_i for player i, $(\sigma_i, \sigma^*_{-i})$ are covered by the dataset, where σ^* is one NE strategy.

Assumption 6.2. (Unilateral Coverage) For all (possible stochastic) strategy σ_i for all player i, $(\sigma_i, \sigma^*_{-i})$ are covered by the dataset, where σ^* is one NE strategy.

Note that deterministic unilateral coverage assumption is equivalent to unilateral coverage assumption. The intuition behind this is that any mixed strategy can be represented by a combination of deterministic strategies. Therefore, if all deterministic strategies are covered by the dataset, then all mixed strategies are also covered. Based on this finding, in the following proof, we only consider all deterministic strategies. Previously, Cui and Du [114] established that unilateral coverage assumption is the minimal sufficient condition for computing an NE strategy in the two-player zero-sum Markov games. However, this unilateral coverage assumption over the offline dataset is **not sufficient for our model-based method** to compute the equilibrium strategy in the offline setting. We formally proved this limitation through the following lemma.

Lemma 6.1. *The unilateral coverage assumption is not sufficient for our model-based method to converge to the equilibrium strategy of the underlying game under the offline setting.*

Proof. We prove this Lemma by providing a counter-example. First, we consider an IIEFG $\mathcal{G}_3$, represented in Figure 6.9a. We can easily find that the NE strategy of game $\mathcal{G}_3$ is strategy profile $\sigma^* = (\sigma_1, \sigma_2) = (\{I_1 : a_1\}, \{I_2 : b_1\})$. To build a dataset

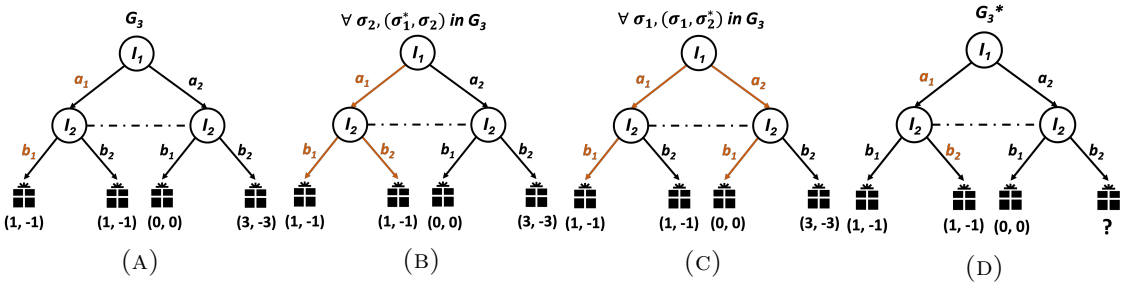


FIGURE 6.9: Counter-example for proving Lemma 6.1.

$\mathcal{D}$ satisfying the unilateral coverage assumption, the dataset needs to cover (σ_1^*, σ_2) for all deterministic strategy σ_2 and (σ_1, σ_2^*) for all deterministic strategy σ_1. We show the state-action pairs covered by these strategy profiles in Figures 6.9b-6.9c. It means that if the dataset $\mathcal{D}$ satisfies the unilateral coverage assumption, then the dataset $\mathcal{D}$ would cover these state-action pairs marked by these orange lines.

When applying our model-based method on the dataset $\mathcal{D}$, the first step is to train an environment model based on the dataset $\mathcal{D}$. Assume that the environment model can be trained well which means that the environment model can precisely represent all game information in the dataset. Therefore, the game represented by the trained environment model would be $\mathcal{G}_3^*$ in Figure 6.9d. Note that there is some missing data to rebuild the game tree. Although our trained environment model can give approximate results for these missing data, it may result in a different equilibrium strategy. For example, if the missing value in $\mathcal{G}_3^*$ is $(0, 0)$ or $(-1, 1)$, then the strategy profile $(\sigma_1, \sigma_2') = (\{I_1 : a_1\}, \{I_2 : b_2\})$ would be the NE strategy of game $\mathcal{G}_3^*$. However, σ is not the NE strategy for the original game $\mathcal{G}_3$. Therefore, the unilateral coverage assumption is not sufficient for our model-based method to converge to the equilibrium strategy of the underlying game. □

6.5.3.1 Minimal Dataset Coverage Assumption

Next, we move forward to provide the minimal coverage assumptions required to ensure the convergence of our methods in IIEFGs with perfect recall. First, we introduce two dataset coverage definitions: *uniform and equilibrium coverage*.

Definition 6.2. An offline dataset $\mathcal{D}$ is said to be a *uniform cover* of an IIEFG $\mathcal{G}$ if and only if $\mathcal{D}$ covers all state-action pairs, i.e., $(s_t, a_t, s_{t+1}, u_{t+1}, d_{t+1}), \forall s_t, a_t$.

Definition 6.3. An offline dataset $\mathcal{D}$ is said to be an *equilibrium cover* of an IIEFG $\mathcal{G}$ if and only if $\mathcal{D}$ is created by a fully mixed equilibrium strategy σ^* of $\mathcal{G}$.

Now, drawing on the dataset coverage definitions previously introduced, we present our results on **the minimal dataset coverage assumptions required for our model-based framework and behavior cloning technique** to converge to the equilibrium strategy of the underlying game.

Theorem 6.2. *For an IIEFG $\mathcal{G}$, the uniform coverage of $\mathcal{G}$ is the minimal dataset coverage assumption that guarantees the convergence of our model-based algorithm.*

Proof. From the example in the proof of Lemma 6.1, we find that a slight violation of the uniform coverage assumption, i.e., only one state-action pair is missing, will impede the computation of the equilibrium strategy using our model-based method. In other words, any state-action pair that is not covered by the dataset may cause failure in computing the equilibrium strategy of the original game using our model-based method. Next, we need to prove that the dataset satisfying the uniform coverage assumption can guarantee the convergence to the equilibrium strategy of the original game using our model-based method.

In our model-based method, we need to train an environment model based on the offline dataset. Therefore, to prove the convergence guarantee under the uniform coverage dataset assumption, we need to verify whether the game reconstructed from the dataset satisfying the uniform coverage assumption is the same as the original game. Here, we reuse the example in Figure 6.3. In that example, the offline dataset $\mathcal{D}$ of the IIEFG $\mathcal{G}$ covers all state-action pairs, i.e., the offline dataset $\mathcal{D}$ satisfies the uniform coverage dataset assumption. From the offline dataset $\mathcal{D}$, we can easily rebuild the game G', as shown in Figure 6.10.

$$S_1 = (I_1, \emptyset, \emptyset, 1, \{a_1, a_2\}), S_2 = (I_1a_1, I_2, \emptyset, 2, \{b_1, b_2\}), S_3 = (I_1a_2, I_2, \emptyset, 2, \{b_1, b_2\}),$$
$$S_4 = (I_1a_1, I_2b_1, \emptyset, -1, \emptyset), S_5 = (I_1a_1, I_2b_2, \emptyset, -1, \emptyset), S_6 = (I_1a_2, I_2b_1, \emptyset, -1, \emptyset),$$
$$S_7 = (I_1a_2, I_2b_2, \emptyset, -1, \emptyset).$$

Especially, for game states S_2 and S_3, the player acting is both Player 2, and the information set for Player 2 is the same. Therefore, these two game states correspond to different game nodes under the same information set. Although Player

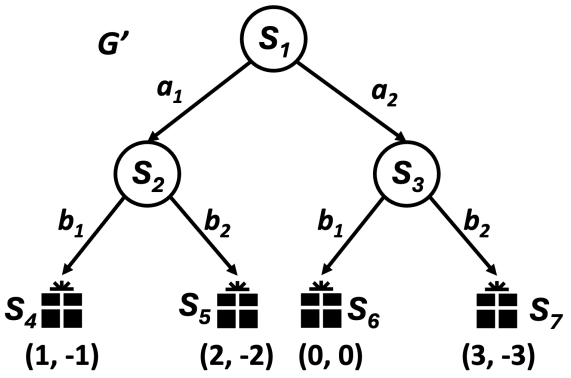


FIGURE 6.10: An IIEFG G'.

2 cannot distinguish these two game states, from the perspective of the game, we can still distinguish them by the information set of Player 1. Particularly, if there is a chance node in the game, the result of the chance node would be recorded in GI within the game state S. Therefore, we can still distinguish these game states by game information GI. Since the dataset satisfying the uniform coverage assumption covers all state-action pairs, the links between game states can be built following these data points in the dataset. Finally, we can find that the reconstructed game tree has the same game states and the same transition function as the original game, thereby the same equilibrium strategy. Therefore, our reconstructed game model can provide the same information as the underlying game of the offline dataset. Then applying our model-based equilibrium finding algorithm to the reconstructed game model definitely can converge to the equilibrium strategy of the underlying game in the offline setting. □

So far, we have demonstrated that the uniform dataset coverage assumption is sufficient for our model-based method to converge to the equilibrium strategy in the offline setting. While this assumption may seem strict, as it requires full coverage of all game states, it is essential for the construction of an accurate environment model, which depends on complete state transition information. Any missing game state data can significantly impact the performance of the model-based method. Therefore, the uniform dataset coverage assumption represents the minimal dataset coverage requirement for the model-based method. In scenarios where random exploration is effective, the offline dataset is more likely to satisfy this assumption. For instance, datasets generated using our random collection method naturally meet the uniform coverage criteria, as discussed in the next section.

For our behavior cloning method, these dataset coverage assumptions may not be sufficient to converge to the equilibrium strategy since its performance mainly depends on the underlying behavior strategy of the dataset. In the following theorem, we provide a minimal dataset coverage assumption for our behavior cloning method to converge to the equilibrium strategy in the offline setting.

Theorem 6.3. *For an IIEFG $\mathcal{G}$, the equilibrium coverage of $\mathcal{G}$ is the minimal dataset coverage assumption that guarantees the convergence of the behavior cloning algorithm.*

Proof. Note that the behavior cloning method can mimic the underlying behavior strategy represented by the offline dataset. According to the definition of equilibrium coverage, the offline dataset $\mathcal{D}$ satisfying the equilibrium coverage is created by a fully mixed equilibrium strategy σ^*. Therefore, $\mathcal{D}$ includes all decision points of players since the equilibrium strategy σ is fully mixed. When applying the behavior cloning method to the offline dataset $\mathcal{D}$, then it would converge to the equilibrium strategy σ^* since it can imitate the underlying behavior strategy σ^* of the offline dataset $\mathcal{D}$. Next, we prove that a slight violation of the equilibrium coverage assumption will impede the convergence of our behavior cloning method.

Here, we reuse the game example in Figure 6.8. Note that the NE strategy of the game is a pure strategy, i.e, $\sigma^* = (\sigma_1^*, \sigma_2^*) = (\{I_1 : a_1\}, \{I_2 : b_2\})$. If we use this equilibrium strategy to generate the offline dataset $\mathcal{D}$, then $\mathcal{D}$ would only include the data point $((I_1^{t_1} = I_1, I_2^{t_1} = \emptyset, GI = \emptyset, 1, \{a_1, a_2\}), a_1, (I_1^{t_2} = I_1 a_1, I_2^{t_2} = \emptyset, GI = \emptyset, -1, \emptyset), (0, 0), 1)$. We cannot get the equilibrium strategy only from $\mathcal{D}$. In this example game, the offline dataset $\mathcal{D}$ is generated by a pure equilibrium strategy instead of a fully mixed equilibrium strategy, and the behavior cloning method cannot get the equilibrium strategy from the offline dataset $\mathcal{D}$ since there is no information about Player 2. Another example is the dataset $\mathcal{D}'$ covering the equilibrium strategy σ^*, i.e., the $\mathcal{D}'$ includes these data points, as follows,

$$
\begin{aligned}
&((I_1, \emptyset, \emptyset, 1, \{a_1, a_2\}), a_1, (I_1 a_1, \emptyset, \emptyset, -1, \emptyset), (0, 0), 1),\\
&((I_1, \emptyset, \emptyset, 1, \{a_1, a_2\}), a_2, (I_1 a_2, I_2, \emptyset, 2, \{b_1, b_2\}), (0, 0), 0),\\
&((I_1 a_2, I_2, \emptyset, 2, \{b_1, b_2\}), b_2, (I_1 a_2, I_2 b_2, \emptyset, -1, \emptyset), (-2, 2), 1).
\end{aligned}
$$

To cover the equilibrium strategy of Player 2, the data point $((I_1^{t_1} = I_1, I_2^{t_1} = \emptyset, GI = \emptyset, 1, \{a_1, a_2\}), a_2, (I_1^{t_2} = I_1 a_2, I_2^{t_2} = I_2, GI = \emptyset, 2, \{b_1, b_2\}), (0, 0), 0)$ should

also be visited. Although $\mathcal{D}'$ covers the equilibrium strategy, $\mathcal{D}'$ does not satisfy the equilibrium coverage assumption since $\mathcal{D}'$ is not created by the equilibrium strategy. Then the behavior cloning method cannot converge to the equilibrium strategy σ^* based on $\mathcal{D}'$ since BC cannot get the pure strategy for Player 1 under the influence of the data point $((I_1^{t_1} = I_1, I_2^{t_1} = \emptyset, GI = \emptyset, 1, \{a_1, a_2\}), a_2, (I_1^{t_2} = I_1 a_2, I_2^{t_2} = I_2, GI = \emptyset, 2, \{b_1, b_2\}), (0, 0), 0)$. Therefore, a slight violation of the equilibrium coverage assumption would cause failure in computing the equilibrium strategy of the original game using our behavior cloning method. In conclusion, the equilibrium coverage assumption is the minimal dataset coverage assumption that guarantees the convergence to the equilibrium strategy of the original game using our behavior cloning method. □

We have demonstrated that the equilibrium dataset coverage assumption is sufficient for the behavior cloning method to converge to the equilibrium strategy in the offline setting. Since behavior cloning can only replicate the behavior strategy underlying the offline dataset, satisfying the equilibrium dataset coverage assumption is essential to guarantee its convergence. In real-world scenarios, the dataset including the game data generated by experts typically meets this assumption. Similarly, in our offline dataset, the data collected using the expert collection method naturally satisfies the equilibrium coverage criteria, as discussed in the next section.

6.5.3.2 Performance Guarantee for BOMB

Next, we turn our attention to analyzing the performance of our methods under different Offline EF datasets. Initially, we focus on the datasets we previously generated. According to our data collection methods, we can establish the following two assumptions regarding the random and expert datasets.

Assumption 6.3. Random dataset of an IIEFG $\mathcal{G}$ is a uniform cover of $\mathcal{G}$.

It is straightforward to get this assumption since the random dataset is sampled by the uniform strategy. As long as we sample enough data, the dataset would involve all the game state-action pairs. Thus, the dataset satisfies the uniform coverage.

Assumption 6.4. Expert dataset of an IIEFG $\mathcal{G}$ is an equilibrium cover of $\mathcal{G}$.

This assumption is built based on the generation process of the expert dataset. In our selected game, we assume that the equilibrium strategies used to generate expert datasets are fully mixed. Since we use the equilibrium strategy to generate the expert dataset, the expert dataset would only cover the data deduced by the equilibrium strategy. In other words, the expert dataset and its corresponding equilibrium strategy profile are bijective relationships.

In our BOMB framework, both the MB and BC algorithms involve training neural networks through supervised learning. The training error, denoted as ϵ can be reduced with sufficient data [148]. Building on assumptions of random and expert datasets and previously discussed minimal dataset coverage assumptions, we first offer the performance guarantee for our BC and MB methods, respectively.

Lemma 6.4. *Suppose the training error ϵ for training the environment model approaches zero, then our model-based method can converge under random datasets and cannot converge under expert datasets.*

Proof. Since the training error, ϵ for training the environment model approaches zero, we can ignore the approximation error. Based on Assumption 6.3, the random dataset satisfies the uniform coverage assumption. The uniform coverage assumption is sufficient for our model-based method to compute the equilibrium strategy according to Theorem 6.2. Thus, our model-based method can converge to an equilibrium strategy under random datasets. The expert dataset satisfies the equilibrium coverage assumption according to Assumption 6.4 while this dataset may not satisfy the uniform coverage assumption. Therefore there is no guarantee that our MB method can converge to an equilibrium strategy under expert datasets. □

Lemma 6.5. *Suppose the training error ϵ for training the behavior cloning policy approaches zero, then our behavior cloning method can converge under expert datasets, and cannot converge under random datasets.*

Proof. The assumption that the training error ϵ for training the behavior cloning policy approaches zero means that the behavior cloning policy can precisely mimic the behavior strategy used to generate the offline dataset. For the expert dataset, according to Assumption 6.4, the expert dataset satisfies the equilibrium coverage assumption. Then the expert dataset is sufficient for the behavior cloning technique to converge to the equilibrium strategy according to Theorem 6.3. However, for

Methods	Dataset	
	Random (Assumpation 6.3)	Expert (Assumption 6.4)
Model-Based	✓(Lemma 6.4)	✗ (Lemma 6.4)
Behavior-Cloning	✗ (Lemma 6.5)	✓ (Lemma 6.5)
BOMB	✓(Theorem 6.6)	✓ (Theorem 6.6)

TABLE 6.2: Summary of theoretical results. (✓ represents that the method can converge.)

the random dataset, according to the generation process of the random dataset and Assumption 6.3, the behavior strategy used to generate the random dataset is a uniform strategy. Therefore, the behavior cloning algorithm can only get a uniform strategy instead of the equilibrium strategy under the random dataset. □

Based on these two lemmas, we can readily derive the below theorem which describes the performance of our BOMB framework when applied to random and expert datasets, thanks to the combination property of BOMB. Table 6.2 summarizes these results for a clear presentation.

Theorem 6.6. *Suppose that the training error ϵ for training the environment model and the behavior cloning policy approaches zero, then our BOMB framework can converge under both the random dataset and the expert dataset.*

Proof. In the BOMB method, the weight of the BC policy is represented by α and the MB policy's weight is $1-\alpha$. The α ranges from 0 to 1. When under the random dataset, let α be equal to 0. Then the policy of our BOMB framework would be reduced to the MB policy, i.e., the policy trained using the MB method. According to Theorem 6.4, the model-based method can converge to an equilibrium strategy under the random dataset. Therefore, the BOMB framework can converge to an equilibrium strategy under the random dataset. Under the expert dataset, let α be equal to 1. Then the policy of BOMB would be reduced to BC policy, i.e., the policy trained by the BC method. Similarly, according to Theorem 6.5, BOMB can converge to an equilibrium strategy under the expert dataset. □

Finally, we extend our analysis to a more general scenario where the offline dataset is collected by an unknown behavior strategy σ. In this context, we first present the following two lemmas describing the performance of our MB and BC methods.

Lemma 6.7. *Suppose that the offline dataset $\mathcal{D}_\sigma$ generated by a behavior strategy σ covers $(s_t, a_t, s_{t+1}, u_{t+1}, d_{t+1}), \forall s_t, a_t$ and the training error ϵ for training the environment model approaches zero, then our model-based method can converge to the equilibrium strategy.*

Proof. According to the definition of uniform coverage assumption and the assumption in this lemma, the dataset $\mathcal{D}_\sigma$ satisfies the uniform coverage assumption. Then according to Theorem 6.2, the dataset $\mathcal{D}_\sigma$ is sufficient for the model-based method for the convergence to the equilibrium strategy. □

Lemma 6.8. *Suppose that the training error ϵ for training the BC policy on the offline dataset $\mathcal{D}_\sigma$, which is generated by a fully mixed behavior strategy σ, approaches zero, then the performance of our trained BC policy σ_θ would be as good as the performance of the behavior strategy σ.*

Proof. According to the assumption in this lemma, the training error can be ignored and the behavior cloning technique can precisely mimic the underlying behavior strategy σ in the offline dataset. Therefore, the trained behavior cloning policy σ_θ would be the same as σ. Consequently, the performance of σ_θ would have the same performance as behavior strategy σ. □

Building on the insights provided by the preceding two lemmas, we propose the following theorem concerning the performance of BOMB under a general case where the offline dataset is generated by an unknown strategy profile.

Theorem 6.9. *Suppose that offline dataset $\mathcal{D}_\sigma$ generated by a fully mixed behavior strategy σ and the training error ϵ for both the environment model and the BC policy approaches zero, then our BOMB framework can converge to a strategy profile σ^* achieving an equal or smaller gap from the equilibrium strategy than the gap between σ and the equilibrium strategy.*

Proof. Following the proof of Theorem 6.6, let α equal 1. Then the BOMB policy would be reduced to the BC policy. According to Theorem 6.8, the performance of the BC policy is as good as behavior strategy σ. Therefore, the BOMB framework can get a strategy that can achieve an equal gap between the behavior strategy σ and the equilibrium strategy. In another extreme case in which $\mathcal{D}_\sigma$ covers $(s_t, a, s_{t+1}), \forall s_t, a \in A(s_t)$, let α be equal to 0. Then the BOMB policy would be

reduced to the MB policy. According to Theorem 6.7, the MB policy can converge to an equilibrium strategy, which would achieve an equal gap from the equilibrium strategy when σ is an equilibrium strategy and a smaller gap from the equilibrium strategy when σ is not an equilibrium strategy. Therefore, in the general case, the BOMB framework can converge to σ^* achieving an equal or a smaller gap from the equilibrium strategy than the gap between σ and the equilibrium strategy. □

6.6 Empirical Evalution

To assess the performance of our proposed algorithm – BOMB, we conduct the following experiments: i) we compare two offline RL algorithms to our BOMB algorithm; ii) we evaluate the performance of different estimation methods; and iii) we run the BOMB framework on various offline datasets to evaluate its performance in computing different equilibrium strategies.

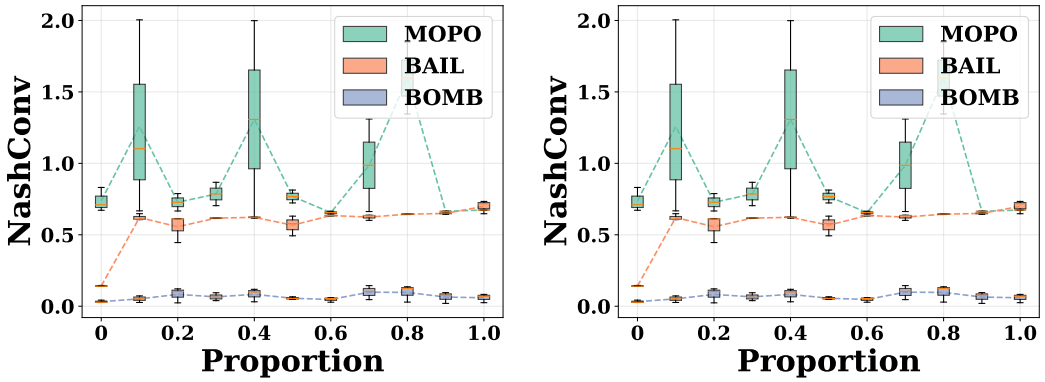
6.6.1 Experimental Setting

OpenSpiel [2] [149] is a collection of environments and algorithms for research in general reinforcement learning and search/planning in games. It is widely accepted and implements many different games. We use it as our experimental platform and opt for several poker games, Liar's Dice, and Phantom Tic-Tac-Toe, which are all widely used in previous works [53, 54], as experimental domains. As introduced in Chapter 2, NashConv (exploitability) is adopted for measuring the strategy of how close to NEs, and (C)CE Gap Sum is used as a measurement of the closeness to (C)CEs. All the experiments are performed on a server with a 10-core 3.3GHz Intel i9-9820X CPU and an NVIDIA RTX 2080 Ti GPU. All results are averaged over three seeds, and the error bars are also reported.

6.6.2 Experimental Results

To better demonstrate the performance of our algorithm, we present our results by answering the following research questions (RQs).

[2] `https://github.com/deepmind/open_spiel`



(A) Kuhn poker (2p) (B) Leduc poker (2p)

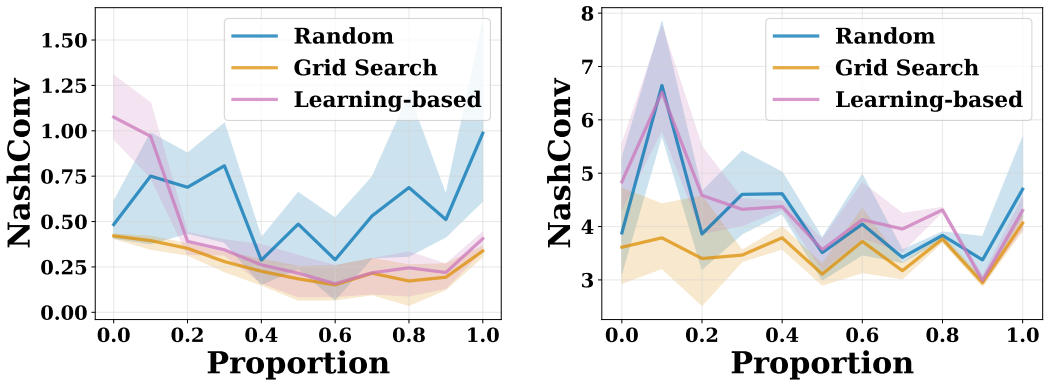
FIGURE 6.11: Comparison with offline RL.

RQ1: *Can the BOMB framework outperform offline RL methods?*

To support this claim that offline RL algorithms are insufficient for the Offline EF paradigm, we choose one model-free algorithm – Best-Action Imitation Learning (BAIL) [110] and one model-based algorithm – Model-based Offline Policy Optimation (MOPO) [112] as the representative of offline RL algorithms. Figure 6.11 shows the comparison results in two-player Kuhn poker and Leduc poker games under the hybrid dataset. The x-axis represents the proportion of data from the random datasets in the hybrid dataset. When the ratio is zero, the hybrid dataset is equivalent to the expert dataset; conversely, when the ratio is one, the hybrid dataset is reduced to the random dataset. We find that the BOMB framework outperforms both offline RL algorithms in all cases. It means that neither of these offline RL algorithms can produce a strategy profile close enough to the equilibrium strategy, which might be attributed to the players' policies being optimized independently.

RQ2: *How do different parameter estimation methods perform?*

As introduced previously, we propose three combination methods: random method, grid search method, and learning-based method. To evaluate the performance of different combination methods, we conduct experiments on poker games. For the learning-based method, we train the parameter predictor on the two-player Kuhn poker game. And then, we test the parameter predictor in other poker games. Figure 6.12 shows the performance results of three combination methods on three-player Kuhn poker and two-player Leduc poker games. We can find that the grid search method achieves the best performance and the learning-based method



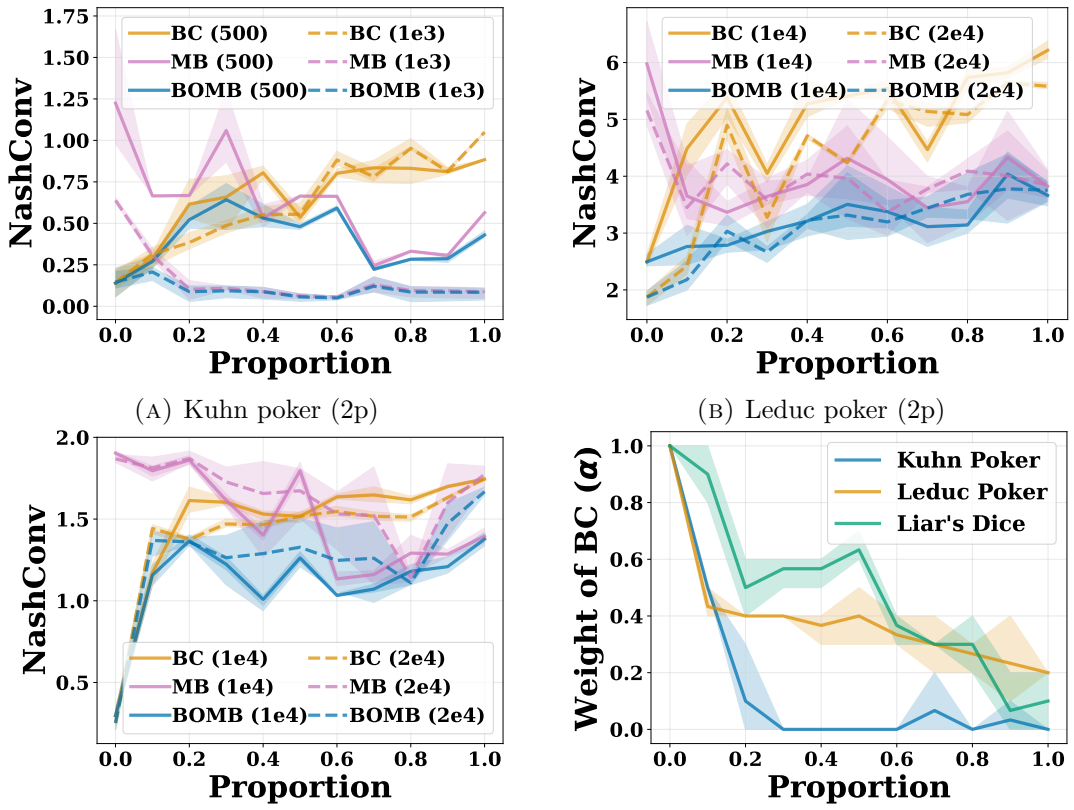
(A) Kuhn poker (3p) (B) Leduc poker (2p)

FIGURE 6.12: Results of different estimation methods.

performs similarly to the grid search method on the three-player Kuhn poker while it performs slightly worse on the two-player Leduc poker game. It implies that the performance of the parameter predictor mainly depends on the difference between the test game and the game used to train the predictor. The most interesting result is that the random method performs well in many cases, which means that even a simple combination can also work well. In the rest of the experiments, we use the grid search method as the parameter estimation method.

RQ3: *Can the BOMB framework compute NE?*

We move to evaluate the performance of our algorithm, BOMB, in computing NE strategy. In addition to performing the BOMB framework, we also assess the individual performance of the behavior cloning technique and the model-based algorithm. It not only helps in understanding the strengths and weaknesses of each component but also provides a comprehensive insight into the efficacy of the BOMB framework in computing the equilibrium offline. Figures 6.13a-6.13c show results on some two-player games under different sizes of offline datasets. Here, we use MB-CFR or MB-PSRO to compute the NE strategy. The MB framework's performance is independent of the algorithm used to compute the NE strategy. As the proportion of the random dataset increases, we observe that the performance of BC decreases while the performance of MB slightly increases. Additionally, we notice that as the size of offline data increases, the improvement of the BC's performance is not significant and the MB's performance improves. It means that the performance of BC mainly depends on the quality of datasets, i.e., the quality of the behavior policy generating the dataset, and the performance of MB relies on the similarity between the environment model and the actual environment. These



(A) Kuhn poker (2p)

(B) Leduc poker (2p)

(C) Liar's dice

(D) Proper weight

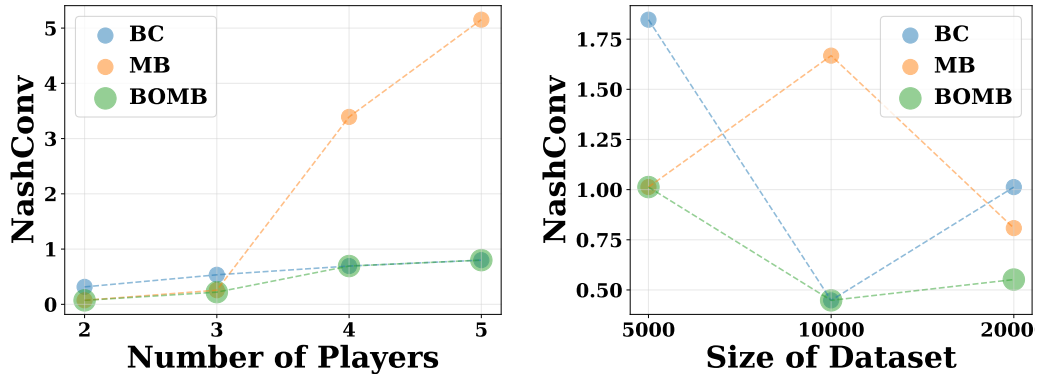
FIGURE 6.13: Experimental results on computing NE.

figures show that our algorithm, BOMB, outperforms both BC and MB methods in all cases, demonstrating its effectiveness in computing NE strategy.

To further analyze the BOMB method, we plot the parameter α for these combined policies, as illustrated in Figure 6.13d, showing that as the proportion of the random dataset increases, the weight of the BC policy decreases. It confirms that the BC policy performs better under the expert dataset while the MB policy performs better under the random dataset. We also conduct experiments on the learning dataset, which closely approximates real-world conditions. Figure 6.14a shows the results of Kuhn poker games with different numbers of players and Figure 6.14b shows the results of Phantom Tic-Tac-Toe under different numbers of offline data. It indicates that given an offline dataset generated by an unknown strategy, our algorithm can consistently obtain a satisfactory approximate NE strategy.

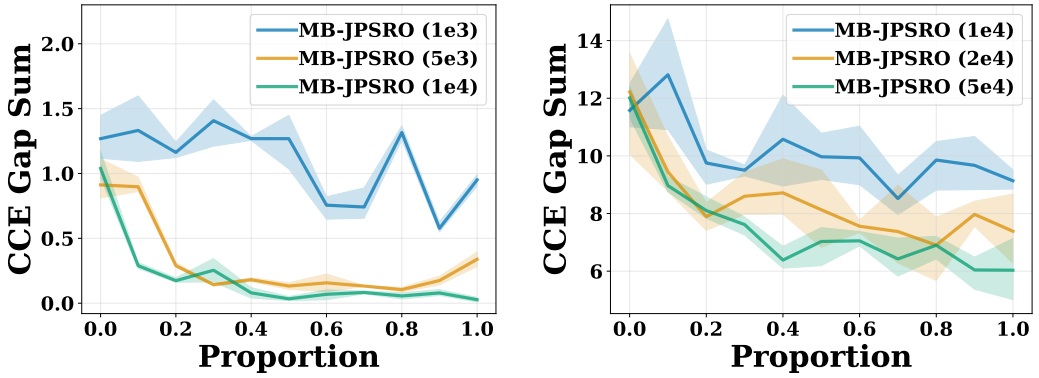
RQ4: *Can the BOMB framework compute CCE?*

We proceed to evaluate the performance of the model-based method in computing the CCE strategy. Here, we do not perform the BC technique or use the BOMB



(A) Kuhn poker (B) Phantom Tic-Tac-Toe

FIGURE 6.14: Results on learning dataset.



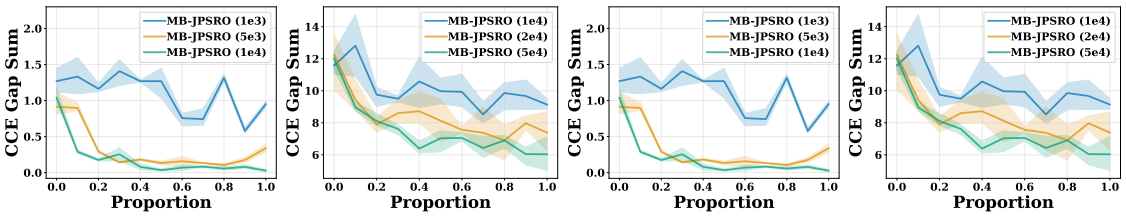
(A) Kuhn poker (3p) (B) Leduc poker (3p)

FIGURE 6.15: Results on computing CCE.

framework to compute the CCE strategy since our Offline EF datasets are collected using an independent strategy for each player, rather than a joint strategy. Figure 6.15 shows the results of performing the MB-JPSRO algorithm on three-player Kuhn poker and Leduc poker games. We can observe that as the size of the offline data increases, the performance of the MB-JPSRO algorithm improves. This further supports the notion that the performance of the model-based method primarily depends on the quality of the trained environment model and also highlights its significance in computing equilibrium strategies in an offline manner.

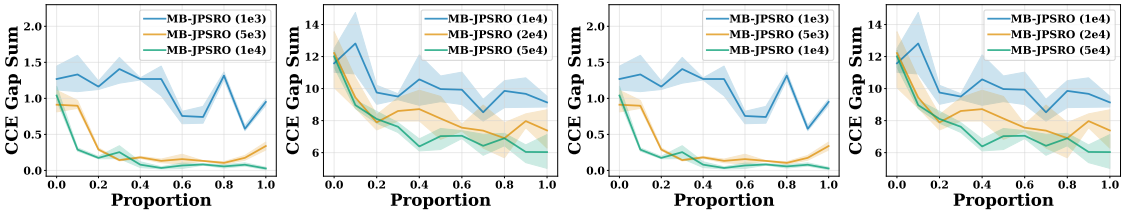
6.6.3 Ablation Study

To investigate the influence of hyperparameters, we conduct several ablation experiments on two-player Kuhn poker and Leduc poker games. We consider different



(A) Kuhn(2p) (B) Kuhn(2p) (C) Leduc(2p) (D) Leduc(2p)

FIGURE 6.16: Abalation results for different hidden layer sizes.



(A) Kuhn(2p) (B) Kuhn(2p) (C) Leduc(2p) (D) Leduc(2p)

FIGURE 6.17: Abalation results for different train epochs.

model structures with various numbers of hidden layers. Specifically, for the two-player Kuhn poker game, we use different environment models with 16, 32, and 64 hidden layers. For the two-player Leduc poker game, which is more complex, the numbers of hidden layers for different models are 32, 64, and 128. In addition, we train the environment models for different epochs to evaluate the robustness of our approach. Figures 6.16-6.17 show these ablation results.

We find that the number of hidden layers and the number of training epochs have little effect on the performance of the behavior cloning (BC) algorithm. These results further verify that the performance of the BC algorithm primarily depends on the quality of the dataset. As we know, the performance of the model-based method mainly depends on the trained environment model. Since the number of hidden layers and the number of training epochs influence the training phase of the environment model, they have a slight impact on the performance of the model-based method. As long as the size of the hidden layers and the number of training epochs are sufficient to ensure that the environment model is trained accurately, the performance of the model-based method will not be affected.

6.7 Chapter Summary

In this chapter, we propose the offline equilibrium finding (Offline EF) paradigm for imperfect-information extensive-form games, which focuses on computing equilibrium strategies solely from offline datasets. Specifically, we first created Offline EF datasets using several established data-collection methods. Then, we proposed a novel algorithm, BOMB, which combines behavior cloning with a model-based approach that adapts existing online equilibrium finding algorithms to the offline setting by introducing an environment model. To better understand the algorithm, we examined its convergence properties and provided a guarantee of the solution quality. Finally, empirical results further validated the superiority of the BOMB framework over existing offline RL algorithms, affirming its efficacy for computing equilibrium strategies offline. We hope our efforts can open up new avenues in equilibrium finding and accelerate research in large-scale game theory.

Chapter 7

Conclusions and Future Directions

In this chapter, we summarize the work conducted in this thesis and then discuss future research directions based on our current results.

7.1 Conclusions

The imperfect-information extensive-form game model is widely used for simulating many real-world problems, as it is well-suited for modeling sequential decision-making processes involving multiple agents. In this thesis, we develop *scalable*, *generalizable*, and *offline* algorithms for solving large-scale imperfect-information extensive-form games, with a particular focus on pursuit-evasion games.

Firstly, we propose two algorithms focusing on improving scalability. Specifically, in Chapter 3, we introduce a novel framework called CFR-MIX to tackle the exponential combinatorial action space problem in Team-Adversary Games. Drawing inspiration from the idea of centralized training with decentralized execution, we propose to represent the team's joint action strategy through the individual strategies of each agent. To ensure cooperation among agents, we define a consistency relationship between these two strategy representations. We then extend this consistency relationship to the cumulative regret values within the CFR framework,

proposing a novel decomposition method to maintain this consistency. This approach is implemented using a mixing layer, which effectively applies the decomposition method. Experimental results demonstrate that our CFR-MIX algorithm significantly outperforms other state-of-the-art CFR-based algorithms.

In Chapter 4, we further enhance scalability by integrating the pre-training and fine-tuning paradigm into the PSRO framework. We begin by modifying the graph embedding algorithm, LINE, to learn a good state representation that retains both the structure information of the graph and the node information. Subsequently, we employ a multi-task reinforcement learning algorithm to train the policy base model for the pursuer. Once the policy base model is pre-trained, we develop a novel best response oracle within the PSRO framework by fine-tuning the pre-trained pursuer's policy. Experimental results in pursuit-evasion games on both synthetic and real-world graphs demonstrate that our algorithm outperforms other baselines in terms of speed and scalability.

Next, we introduce an algorithm aimed at enhancing generalizability. In Chapter 5, we propose a versatile framework called Grasper, designed to efficiently provide high-quality solutions for various pursuit-evasion games under different initial conditions. To implement this framework, we propose a three-stage training method: a pre-pretraining stage where the GNN is trained using GraphMAE, a pre-training stage that involves training the hypernetwork through heuristic-guided multi-task pre-training (HMP), and a fine-tuning stage where the best response strategy for the pursuer is refined. Experimental results demonstrate Grasper's superiority over existing baselines in terms of solution quality and generalizability.

Finally, we propose an offline algorithm for solving imperfect-information extensive-form games. In Chapter 6, we explore the offline equilibrium finding (Offline EF) paradigm, which computes equilibrium strategies solely from offline datasets. We begin by creating a collection of diverse offline datasets using several established data-collection methods. We then introduce a novel framework called BOMB, which integrates behavior cloning with a model-based approach along with a novel parameter estimation method. Notably, the model-based approach can adapt standard online equilibrium finding algorithms to the offline setting by incorporating an environment model. To provide a deeper understanding of our algorithm, we present a comprehensive theoretical analysis of the BOMB framework, offering performance guarantees across various datasets. Experimental results validate the

superiority of the BOMB framework over existing offline RL algorithms, demonstrating its effectiveness in computing equilibrium strategies in an offline setting.

7.2 Future Directions

In the final section of the thesis, we discuss several potential future directions that can further advance the research presented.

Firstly, although our proposed algorithms can improve scalability and solve large-scale games, they are tailored to special games, such as team-adversary games and pursuit-evasion games. For more general IIEFGs, these algorithms may still need further refinement and adaptation. Therefore, the potential research direction could be exploring scalable algorithms for solving general IIEFGs. Specifically, this involves investigating scalable algorithms that can be applied to a wider range of IIEFGs, ensuring that the scalability improvements are not limited to specific games. This would entail developing flexible and robust algorithms capable of handling various game structures and dynamics.

Secondly, while our work has demonstrated significant improvements in generalizability with the Grasper framework for PEGs, it is still designed for the specific game and solution concept. For the game, except for the NE solution concept, there are many other equilibrium strategies. Therefore, one potential future research direction would be investigating generalizable algorithms for computing different equilibrium strategies. Specifically, this involves exploring algorithms that can compute various equilibrium strategies based on different user requirements, utilizing techniques such as hypernetworks. Recently, the large language model (LLM) has achieved significant achievement in solving multiple tasks. Therefore, another potential future research direction could focus on exploring the integration of advanced LLMs to enhance generalizability. By leveraging the capabilities of LLMs, we can improve the flexibility and robustness of our framework, enabling it to handle a broader range of equilibrium concepts and adapt to diverse game-theoretic scenarios more effectively.

Finally, although we have researched exploring the application of offline learning to the domain of game theory, this research is still in its preliminary stages. While our BOMB framework has proven successful in computing equilibrium strategy in

an offline setting, there is still room for improvement. Therefore, several potential research directions could further enhance the file of Offline EF: i) Investigating more advanced offline methods and parameter estimation techniques to enhance the effectiveness and efficiency of solving more complicated games within the offline equilibrium finding paradigm; ii) Exploring the offline learning paradigm in other game models, such as Stochastic games and Bayesian games, to bridge the gap between theoretical research and real-world applications; iii) Incorporating real-world data into the offline learning process to validate and refine the algorithms in practical applications, ensuring their applicability in real-world scenarios.

List of Publications

- **Shuxin Li**, Youzhi Zhang, Xinrun Wang, Wanqi Xue, Bo An. CFR-MIX: Solving imperfect information extensive-form games with combinatorial action space. *In Proceedings of the 30th International Joint Conference on Artificial Intelligence.* **(IJCAI'21)**

- **Shuxin Li**, Xinrun Wang, Youzhi Zhang, Wanqi Xue, Jakub Cerny, Bo An. Solving large-scale pursuit-evasion games using pre-trained strategies.*In Proceedings of the 37th AAAI Conference on Artificial Intelligence.***(AAAI'23)**

- Pengdeng Li*, **Shuxin Li***, Xinrun Wang, Jakub Cerny, Youzhi Zhang, Stephen McAleer, Hau Chan, Bo An. Grasper: A generalist pursuer for pursuit-evasion problems. *In Proceedings of the 23rd International Conference on Autonomous Agents and Multi-Agent Systems.* **(AAMAS'24)**

- Pengdeng Li, **Shuxin Li***, Chang Yang*, Xinrun Wang, Shuyue Hu, Xiao Huang, Hau Chan, Bo An. Configurable mirror descent: Towards a unification of decision making. *In Proceedings of the 41st International Conference on Machine Learning.* **(ICML'24)**

- Pengdeng Li, **Shuxin Li**, Chang Yang, Xinrun Wang, Xiao Huang, Hau Chan, Bo An. Self-adaptive PSRO: Towards an automatic population-based game solver. *In Proceeding of the 33rd International Joint Conference on Artificial Intelligence.* **(IJCAI'24)**

- Xinrun Wang, Chang Yang, **Shuxin Li**, Pengdeng Li, Xiao Huang, Hau Chan, Bo An. Reinforcement Nash equilibrium solver. *In Proceeding of the 33rd International Joint Conference on Artificial Intelligence.* **(IJCAI'24)**

- Pengdeng Li, Xinrun Wang, **Shuxin Li**, Hau Chan, Bo An. Population-size-aware policy optimization for mean-field games. *In Proceedings of the 2023 International Conference on Learning Representations.* **(ICLR'23)**

- Wanqi Xue, Youzhi Zhang, **Shuxin Li**, Bo An, Chai Kiat Yeo. Solving large-scale extensive-form network security games via neural fictitious self-play. *In Proceedings of the 30th International Joint Conference on Artificial Intelligence.* **(IJCAI'21)**

Bibliography

[1] Noam Nisan, Tim Roughgarden, Eva Tardos, and Vijay V Vazirani. *Algorithmic Game Theory*. Cambridge University Press, 2007. 1, 7, 87

[2] John F Nash Jr. Equilibrium points in n-person games. *Proceedings of the National Academy of Sciences*, 36(1):48–49, 1950. 1, 11, 18, 26, 48, 90

[3] Paul Klemperer. Auction theory: A guide to the literature. *Journal of Economic Surveys*, 13(3):227–286, 1999. 1

[4] Gangshu Cai and Peter R Wurman. Monte carlo approximation in incomplete information, sequential auction games. *Decision Support Systems*, 39(2):153–168, 2005.

[5] Vincent P Crawford. Experiments on cognition, communication, coordination, and cooperation in relationships. *Annual Review of Economics*, 11: 167–191, 2019. 1

[6] Ken Binmore and Nir Vulkan. Applying game theory to automated negotiation. *Netnomics*, 1(1):1–9, 1999. 1

[7] Noam Brown and Tuomas Sandholm. Superhuman AI for multiplayer poker. *Science*, 365(6456):885–890, 2019. 87

[8] Matej Moravčík, Martin Schmid, Neil Burch, Viliam Lisỳ, Dustin Morrill, Nolan Bard, Trevor Davis, Kevin Waugh, Michael Johanson, and Michael Bowling. DeepStack: Expert-level artificial intelligence in heads-up no-limit poker. *Science*, 356(6337):508–513, 2017. 1, 2

[9] Yoav Shoham and Kevin Leyton-Brown. *Multiagent Systems: Algorithmic, Game-theoretic, and Logical Foundations*. Cambridge University Press, 2008. 1, 8, 10, 87

[10] H Brendan McMahan, Geoffrey J Gordon, and Avrim Blum. Planning in the presence of cost functions controlled by an adversary. In *Proceedings of the 20th International Conference on Machine Learning*, pages 536–543, 2003. 1, 17, 87, 93

[11] Manish Jain, Dmytro Korzhyk, Ondřej Vaněk, Vincent Conitzer, Michal Pěchouček, and Milind Tambe. A double oracle algorithm for zero-sum security games on graphs. In *Proceeding of the 10th International Conference on Autonomous Agents and Multiagent Systems*, pages 327–334, 2011. 1, 49

[12] Martin Zinkevich, Michael Johanson, Michael Bowling, and Carmelo Piccione. Regret minimization in games with incomplete information. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, pages 1729–1736, 2007. 1, 3, 13, 16, 23, 45, 70, 87, 92, 97

[13] Marc Lanctot, Vinicius Zambaldi, Audrūnas Gruslys, Angeliki Lazaridou, Karl Tuyls, Julien Pérolat, David Silver, and Thore Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 4193–4206, 2017. 1, 3, 17, 18, 45, 46, 49, 57, 70, 87, 92, 97, 100

[14] Michael Bowling, Neil Burch, Michael Johanson, and Oskari Tammelin. Heads-up limit Hold'em poker is solved. *Science*, 347(6218):145–149, 2015. 1, 16

[15] Noam Brown, Tuomas Sandholm, and Strategic Machine. Libratus: The superhuman AI for no-limit poker. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 5226–5228, 2017. 2

[16] Youzhi Zhang, Bo An, Long Tran-Thanh, Zhen Wang, Jiarui Gan, and Nicholas R Jennings. Optimal escape interdiction on transportation networks. In *Proceedings of the 26th International Joint Conference on Artificial Intelligence*, pages 3936–3944, 2017. 2, 42, 65, 70

[17] Youzhi Zhang, Qingyu Guo, Bo An, Long Tran-Thanh, and Nicholas R Jennings. Optimal interdiction of urban criminals with the aid of real-time information. In *Proceedings of the 33rd AAAI conference on artificial intelligence*, pages 1262–1269, 2019. 2, 42, 48, 65, 70

[18] Arunesh Sinha, Fei Fang, Bo An, Christopher Kiekintveld, and Milind Tambe. Stackelberg security games: Looking beyond a decade of success. In *Proceedings of the 27th International Joint Conference on Artificial Intelligence*, pages 5494–5501, 2018. 2, 45

[19] Frederick P Rivara and Christopher D Mack. Motor vehicle crash deaths related to police pursuits in the united states. *Injury Prevention*, 10(2): 93–95, 2004. 2, 45

[20] Marc Lanctot, Kevin Waugh, Martin Zinkevich, and Michael Bowling. Monte Carlo sampling for regret minimization in extensive games. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, pages 1078–1086, 2009. 3, 17, 23

[21] Noam Brown, Adam Lerer, Sam Gross, and Tuomas Sandholm. Deep counterfactual regret minimization. In *Proceedings of the 36th International Conference on Machine Learning*, pages 793–802, 2019. 3, 17, 24, 36, 38, 40, 100

[22] Hui Li, Kailiang Hu, Shaohua Zhang, Yuan Qi, and Le Song. Double neural counterfactual regret minimization. In *Proceedings of the 8th International Conference on Learning Representations*, pages 1–13, 2020. 3, 17, 24, 40

[23] Christian Gourieroux and Joann Jasiak. *Financial Econometrics: Problems, Models, and Methods*. Princeton University Press, 2022. 4, 88

[24] Ansam Khraisat, Iqbal Gondal, Peter Vamplew, and Joarder Kamruzzaman. Survey of intrusion detection systems: Techniques, datasets and challenges. *Cybersecurity*, 2(1):1–22, 2019. 4, 88

[25] Usha Kiruthika, Thamarai Selvi Somasundaram, and S Kanaga Suba Raja. Lifecycle model of a negotiation agent: A survey of automated negotiation techniques. *Group Decision and Negotiation*, 29(6):1239–1262, 2020. 4, 88

[26] Tabish Rashid, Mikayel Samvelyan, Christian Schroeder, Gregory Farquhar, Jakob Foerster, and Shimon Whiteson. QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning. In *Proceedings of the 35th International Conference on Machine Learning*, pages 4295–4304, 2018. 5

[27] Zhenyu Hou, Xiao Liu, Yukuo Cen, Yuxiao Dong, Hongxia Yang, Chunjie Wang, and Jie Tang. GraphMAE: self-supervised masked graph autoencoders. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 594–604, 2022. 6, 67, 71, 75

[28] Robert J Aumann. Correlated equilibrium as an expression of Bayesian rationality. *Econometrica: Journal of the Econometric Society*, 55(1):1–18, 1987. 12, 90

[29] Luke Marris, Paul Muller, Marc Lanctot, Karl Tuyls, and Thore Graepel. Multi-agent training beyond zero-sum with correlated equilibrium meta-solvers. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 7480–7491, 2021. 13, 19, 100

[30] Sergiu Hart and Andreu Mas-Colell. A simple adaptive procedure leading to correlated equilibrium. *Econometrica*, 68(5):1127–1150, 2000. 15

[31] Richard Gibson, Marc Lanctot, Neil Burch, Duane Szafron, and Michael Bowling. Generalized sampling and variance in counterfactual regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1355–1361, 2012. 17, 23

[32] Martin Schmid, Neil Burch, Marc Lanctot, Matej Moravcik, Rudolf Kadlec, and Michael Bowling. 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*, pages 2157–2164, 2019. 17

[33] Eric Steinberger, Adam Lerer, and Noam Brown. Dream: Deep regret minimization with advantage baselines and model-free learning. *arXiv preprint arXiv:2006.10410*, 2020. 17

[34] Eric Steinberger. Single deep counterfactual regret minimization. *arXiv preprint arXiv:1901.07621*, 2019. 17, 24

[35] Paul Muller, Shayegan Omidshafiei, Mark Rowland, Karl Tuyls, Julien Perolat, Siqi Liu, Daniel Hennes, Luke Marris, Marc Lanctot, Edward Hughes, et al. A generalized training approach for multiagent learning. In *Proceedings of the 7th International Conference on Learning Representations*, 2019. 17

[36] Karl Tuyls and Ronald Westra. Replicator dynamics in discrete and continuous strategy spaces. *Multi-Agent Systems: Simulation and Applications*, pages 215–240, 2009. 18

[37] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A Rusu, Joel Veness, Marc G Bellemare, Alex Graves, Martin Riedmiller, Andreas K Fidjeland, Georg Ostrovski, et al. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015. 19

[38] David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, and Martin Riedmiller. Deterministic policy gradient algorithms. In *Proceedings of the 31st International Conference on Machine Learning*, pages 387–395, 2014. 19

[39] George W. Brown. Iterative solution of games by fictitious play. *Activity Analysis of Production and Allocation*, pages 374–376, 1951. 19

[40] Johannes Heinrich, Marc Lanctot, and David Silver. Fictitious self-play in extensive-form games. In *Proceedings of the 32rd International Conference on Machine Learning*, pages 805–813, 2015. 19

[41] Johannes Heinrich and David Silver. Deep reinforcement learning from self-play in imperfect-information games. *arXiv preprint arXiv:1603.01121*, 2016. 19

[42] Stephen McAleer, John Lanier, Roy Fox, and Pierre Baldi. Pipeline psro: a scalable approach for finding approximate nash equilibria in large games. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 20238–20248, 2020. 19, 46, 50

[43] Stephen Marcus McAleer, John Banister Lanier, Kevin Wang, Pierre Baldi, and Roy Fox. XDO: A double oracle algorithm for extensive-form games. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, 2021. 19

[44] Stephen McAleer, Kevin Wang, Marc Lanctot, John Lanier, Pierre Baldi, and Roy Fox. Anytime optimal PSRO for two-player zero-sum games. *arXiv preprint arXiv:2201.07700*, 2022. 19

[45] Shuxin Li, Youzhi Zhang, Xinrun Wang, Wanqi Xue, and Bo An. CFR-MIX: Solving imperfect information extensive-form games with combinatorial action space. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, page 3663–3669, 2021. 23, 46, 49

[46] Nicola Basilico, Andrea Celli, Giuseppe De Nittis, and Nicola Gatti. Coordinating multiple defensive resources in patrolling games with alarm systems. In *Proceedings of the 16th International Conference on Autonomous Agents and Multiagent Systems*, pages 678–686, 2017. 23

[47] Nicholas Abou Risk, Duane Szafron, et al. Using counterfactual regret minimization to create competitive multiplayer poker agents. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 159–166, 2010. 24

[48] Andrea Celli and Nicola Gatti. Computational results for extensive-form adversarial team games. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 30

[49] Oskari Tammelin, Neil Burch, Michael Johanson, and Michael Bowling. Solving heads-up limit Texas hold'em. In *Proceedings of the 24th International Conference on Artificial Intelligence*, pages 645–652, 2015. 32

[50] Dustin Morrill. *Using Regret Estimation to Solve Games Compactly*. PhD thesis, University of Alberta, 2016. 37

[51] Marc Lanctot. *Monte Carlo Sampling and Regret Minimization for Equilibrium Computation and Decision-Making in Large Extensive Form Games*. PhD thesis, University of Alberta, 2013. 38

[52] Sheldon M. Ross. Goofspiel — the game of pure strategy. *Journal of Applied Probability*, 8(3):621–625, 1971. 40

[53] Viliam Lisỳ, Marc Lanctot, and Michael Bowling. Online monte carlo counterfactual regret minimization for search in imperfect information games. In *Proceedings of the 14th International Conference on Autonomous Agents and Multiagent Systems*, pages 27–36, 2015. 41, 112

[54] Noam Brown and Tuomas Sandholm. Solving imperfect-information games via discounted regret minimization. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1829–1836, 2019. 41, 112

[55] Karel Horák and Branislav Bošanskỳ. Dynamic programming for one-sided partially observable pursuit-evasion games. In *Proceedings of the International Conference on Agents and Artificial Intelligence*, pages 503–510, 2017. 42, 70, 71

[56] Shuxin Li, Xinrun Wang, Youzhi Zhang, Wanqi Xue, Jakub Černý, and Bo An. Solving large-scale pursuit-evasion games using pre-trained strategies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 11586–11594, 2023. 45, 65, 66, 71, 73, 87

[57] Wanqi Xue, Youzhi Zhang, Shuxin Li, Xinrun Wang, Bo An, and Chai Kiat Yeo. Solving large-scale extensive-form network security games via neural fictitious self-play. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, page 3713–3720, 2021. 46, 48, 54, 56, 65, 69, 70, 73, 79, 87

[58] Wanqi Xue, Bo An, and Chai Kiat Yeo. Nsgzero: Efficiently learning non-exploitable policy in large-scale network security games with neural monte carlo tree search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 4646–4653, 2022. 46, 48, 54, 56, 57, 65, 69

[59] David Balduzzi, Marta Garnelo, Yoram Bachrach, Wojciech Czarnecki, Julien Perolat, Max Jaderberg, and Thore Graepel. Open-ended learning in symmetric zero-sum games. In *Proceedings of the 36th International Conference on Machine Learning*, pages 434–443, 2019. 46

[60] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4171–4186, 2019. 46

[61] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. `https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf`, 2018. 47

[62] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al. Mastering the game of Go with deep neural networks and tree search. *Nature*, 529(7587):484–489, 2016. 47

[63] Oriol Vinyals, Igor Babuschkin, Wojciech M Czarnecki, Michaël Mathieu, Andrew Dudzik, Junyoung Chung, David H Choi, Richard Powell, Timo Ewalds, Petko Georgiev, et al. Grandmaster level in starcraft ii using multi-agent reinforcement learning. *Nature*, 575(7782):350–354, 2019. 47, 50

[64] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9, 2015. 47

[65] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016. 47

[66] Xu Han, Zhengyan Zhang, Ning Ding, Yuxian Gu, Xiao Liu, Yuqi Huo, Jiezhong Qiu, Yuan Yao, Ao Zhang, Liang Zhang, et al. Pre-trained models: Past, present and future. *AI Open*, 2:225–250, 2021. 47

[67] Branislav Bosansky, Christopher Kiekintveld, Viliam Lisy, and Michal Pechoucek. An exact double-oracle algorithm for zero-sum extensive-form games with imperfect information. *Journal of Artificial Intelligence Research*, 51:829–866, 2014. 48

[68] Branislav Bosansky, Albert Xin Jiang, Milind Tambe, and Christopher Kiekintveld. Combining compact representation and incremental generation in large games with sequential strategies. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 812–818, 2015. 48

[69] Yu Zhang and Qiang Yang. A survey on multi-task learning. *IEEE Transactions on Knowledge and Data Engineering*, 34(12):5586–5609, 2021. 50, 66, 71

[70] Sebastian Ruder. An overview of multi-task learning in deep neural networks. *arXiv preprint arXiv:1706.05098*, 2017. 50, 71

[71] Aaron Wilson, Alan Fern, Soumya Ray, and Prasad Tadepalli. Multi-task reinforcement learning: A hierarchical bayesian approach. In *Proceedings of the 24th International Conference on Machine Learning*, pages 1015–1022, 2007. 50, 71

[72] Tung-Long Vuong, Do-Van Nguyen, Tai-Long Nguyen, Cong-Minh Bui, Hai-Dang Kieu, Viet-Cuong Ta, Quoc-Long Tran, and Thanh-Ha Le. Sharing experience in multitask reinforcement learning. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, pages 3642–3648, 2019. 50

[73] Mandi Zhao, Pieter Abbeel, and Stephen James. On the effectiveness of fine-tuning versus meta-reinforcement learning. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 26519–26531, 2022. 50, 71

[74] Jian Tang, Meng Qu, Mingzhe Wang, Ming Zhang, Jun Yan, and Qiaozhu Mei. LINE: Large-scale information network embedding. In *Proceedings of the 24th International Conference on World Wide Web*, pages 1067–1077, 2015. 51

[75] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017. 55

[76] Chao Yu, Akash Velu, Eugene Vinitsky, Jiaxuan Gao, Yu Wang, Alexandre Bayen, and Yi Wu. The surprising effectiveness of PPO in cooperative multi-agent games. In *Proceedings of the 36th International Conference on Neural Information Processing Systems, Datasets and Benchmarks Track*, pages 24611–24624, 2022. 55, 76

[77] Geoff Boeing. OSMnx: New methods for acquiring, constructing, analyzing, and visualizing complex street networks. *Computers, Environment and Urban Systems*, 65:126–139, 2017. 57

[78] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, pages 1126–1135, 2017. 61

[79] Pengdeng Li, Shuxin Li, Xinrun Wang, Jakub Cernỳ, Youzhi Zhang, Stephen McAleer, Hau Chan, and Bo An. Grasper: A generalist pursuer for pursuit-evasion problems. In *Proceedings of the 23rd International Conference on Autonomous Agents and Multiagent Systems*, pages 1147–1155, 2024. 65

[80] Shuxin Li, Youzhi Zhang, Xinrun Wang, Wanqi Xue, and Bo An. CFR-MIX: Solving imperfect information extensive-form games with combinatorial action space. In *Proceedings of the 30th International Joint Conference on Artificial Intelligence*, pages 3663–3669, 2021. 65, 70

[81] Pengdeng Li, Xinrun Wang, Shuxin Li, Hau Chan, and Bo An. Population-size-aware policy optimization for mean-field games. In *Proceedings of the 11th International Conference on Learning Representations*, 2023. 66

[82] Jakob Foerster, Gregory Farquhar, Triantafyllos Afouras, Nantas Nardelli, and Shimon Whiteson. Counterfactual multi-agent policy gradients. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 2974–2982, 2018. 68

[83] Rene Vidal, Omid Shakernia, H Jin Kim, David Hyunchul Shim, and Shankar Sastry. Probabilistic pursuit-evasion games: Theory, implementation, and experimental evaluation. *IEEE Transactions on Robotics and Automation*, 18(5):662–669, 2002. 70

[84] Shaunak D Bopardikar, Francesco Bullo, and Joao P Hespanha. On discrete-time pursuit-evasion games with sensing limitations. *IEEE Transactions on Robotics*, 24(6):1429–1439, 2008.

[85] Victor G Lopez, Frank L Lewis, Yan Wan, Edgar N Sanchez, and Lingling Fan. Solutions for multiagent pursuit-evasion games on communication graphs: Finite-time capture and asymptotic behaviors. *IEEE Transactions on Automatic Control*, 65(5):1911–1923, 2019.

[86] Yuanda Wang, Lu Dong, and Changyin Sun. Cooperative control for multi-player pursuit-evasion games with reinforcement learning. *Neurocomputing*, 412:101–114, 2020.

[87] Linan Huang and Quanyan Zhu. A dynamic game framework for rational and persistent robot deception with an application to deceptive pursuit-evasion. *IEEE Transactions on Automation Science and Engineering*, 19 (4):2918–2932, 2021.

[88] Xiuxian Li, Min Meng, Yiguang Hong, and Jie Chen. A survey of decision making in adversarial games. *Science China Information Sciences*, 67(4): 141–201, 2024. 70, 71

[89] Jason Tsai, Zhengyu Yin, Jun-young Kwak, David Kempe, Christopher Kiekintveld, and Milind Tambe. Urban security: Game-theoretic resource allocation in networked domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 881–886, 2010. 71

[90] Zhijian Duan, Wenhan Huang, Dinghuai Zhang, Yali Du, Jun Wang, Yaodong Yang, and Xiaotie Deng. Is Nash equilibrium approximator learnable? In *Proceedings of the 22nd International Conference on Autonomous Agents and Multiagent Systems*, pages 233–241, 2023. 71

[91] Zhijian Duan, Yunxuan Ma, and Xiaotie Deng. Are equivariant equilibrium approximators beneficial? In *Proceedings of the 40th International Conference on Machine Learning*, pages 8747–8778, 2023. 71

[92] Xidong Feng, Oliver Slumbers, Ziyu Wan, Bo Liu, Stephen McAleer, Ying Wen, Jun Wang, and Yaodong Yang. Neural auto-curricula in two-player zero-sum games. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pages 3504–3517, 2021.

[93] Luke Marris, Ian Gemp, Thomas Anthony, Andrea Tacchetti, Siqi Liu, and Karl Tuyls. Turbocharging solution concepts: Solving NEs, CEs and CCEs with neural equilibrium solvers. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 5586–5600, 2022. 71

[94] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. 71

[95] Jiezhong Qiu, Qibin Chen, Yuxiao Dong, Jing Zhang, Hongxia Yang, Ming Ding, Kuansan Wang, and Jie Tang. GCC: Graph contrastive coding for graph neural network pre-training. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1150–1160, 2020. 71

[96] Shantanu Thakoor, Corentin Tallec, Mohammad Gheshlaghi Azar, Mehdi Azabou, Eva L Dyer, Remi Munos, Petar Veličković, and Michal Valko. Large-scale representation learning on graphs via bootstrapping. In *Proceedings of the 10th International Conference on Learning Representations*, 2022. 71

[97] Hengrui Zhang, Qitian Wu, Junchi Yan, David Wipf, and Philip S Yu. From canonical correlation analysis to self-supervised graph neural networks. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pages 76–89, 2021. 71

[98] Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the web conference 2021*, pages 2069–2080, 2021. 71

[99] Ronan Collobert and Jason Weston. A unified architecture for natural language processing: Deep neural networks with multitask learning. In *Proceedings of the 25th International Conference on Machine Learning*, pages 160–167, 2008. 71

[100] Ross Girshick. Fast R-CNN. In *ICCV*, pages 1440–1448, 2015. 71

[101] Sihan Zeng, Malik Aqeel Anwar, Thinh T Doan, Arijit Raychowdhury, and Justin Romberg. A decentralized policy gradient approach to multi-task reinforcement learning. In *Uncertainty in Artificial Intelligence*, pages 1002–1012, 2021. 71

[102] David Ha, Andrew M. Dai, and Quoc V. Le. Hypernetworks. In *Proceedings of the 5th International Conference on Learning Representations*, 2017. 73

[103] Yuri Burda, Harrison Edwards, Amos Storkey, and Oleg Klimov. Exploration by random network distillation. In *Proceedings of the 7th International Conference on Learning Representations*, 2019. 77

[104] Martin Schmid, Matej Moravčík, Neil Burch, Rudolf Kadlec, Josh Davidson, Kevin Waugh, Nolan Bard, Finbarr Timbers, Marc Lanctot, G Zacharias Holland, Davoodi Davoodi, Alden Christianson, and Michael Bowling. Student of games: a unified learning algorithm for both perfect and imperfect information games. *Science Advances*, 9(46):eadg3256, 2023. 79

[105] Noam Brown and Tuomas Sandholm. Superhuman AI for heads-up no-limit poker: Libratus beats top professionals. *Science*, 359(6374):418–424, 2018. 87

[106] Daochen Zha, Jingru Xie, Wenye Ma, Sheng Zhang, Xiangru Lian, Xia Hu, and Ji Liu. Douzero: Mastering doudizhu with self-play deep reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, pages 12333–12344, 2021. 87

[107] Julien Perolat, Bart De Vylder, Daniel Hennes, Eugene Tarassov, Florian Strub, Vincent de Boer, Paul Muller, Jerome T Connor, Neil Burch, Thomas Anthony, et al. Mastering the game of stratego with model-free multiagent reinforcement learning. *Science*, 378(6623):990–996, 2022. 87

[108] Sergey Levine, Aviral Kumar, George Tucker, and Justin Fu. Offline reinforcement learning: Tutorial, review, and perspectives on open problems. *arXiv preprint arXiv:2005.01643*, 2020. 88, 90, 93

[109] Aviral Kumar, Aurick Zhou, George Tucker, and Sergey Levine. Conservative q-learning for offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 1179–1191, 2020. 88, 93

[110] Xinyue Chen, Zijian Zhou, Zheng Wang, Che Wang, Yanqiu Wu, and Keith Ross. Bail: Best-action imitation learning for batch deep reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 18353–18363, 2020. 88, 93, 99, 113

[111] Rahul Kidambi, Aravind Rajeswaran, Praneeth Netrapalli, and Thorsten Joachims. MOReL: Model-based offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 21810–21823, 2020. 88, 94, 100

[112] Tianhe Yu, Garrett Thomas, Lantao Yu, Stefano Ermon, James Y Zou, Sergey Levine, Chelsea Finn, and Tengyu Ma. MOPO: Model-based offline policy optimization. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 14129–14142, 2020. 88, 94, 99, 100, 113

[113] Michael Mathieu, Sherjil Ozair, Srivatsan Srinivasan, Caglar Gulcehre, Shangtong Zhang, Ray Jiang, Tom Le Paine, Konrad Zolna, Richard Powell, Julian Schrittwieser, et al. StarCraft II unplugged: Large scale offline reinforcement learning. In *Deep RL Workshop NeurIPS 2021*, 2021. 88

[114] Qiwen Cui and Simon S Du. When are offline two-player zero-sum markov games solvable? In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 25779–25791, 2022. 89, 91, 103

[115] Han Zhong, Wei Xiong, Jiyuan Tan, Liwei Wang, Tong Zhang, Zhaoran Wang, and Zhuoran Yang. Pessimistic minimax value iteration: Provably efficient equilibrium learning from offline datasets. In *Proceedings of the 39th International Conference on Machine Learning*, pages 27117–27142, 2022. 89

[116] He He, Jordan Boyd-Graber, Kevin Kwok, and Hal Daumé III. Opponent modeling in deep reinforcement learning. In *Proceedings of the 33rd International Conference on Machine Learning*, pages 1804–1813, 2016. 91

[117] Yan Zheng, Zhaopeng Meng, Jianye Hao, Zongzhang Zhang, Tianpei Yang, and Changjie Fan. A deep bayesian policy reuse approach against non-stationary agents. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems*, pages 962–972, 2018. 91

[118] Aditya Grover, Maruan Al-Shedivat, Jayesh Gupta, Yuri Burda, and Harrison Edwards. Learning policy representations in multiagent systems. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1802–1811, 2018. 91

[119] Zhang-Wei Hong, Shih-Yang Su, Tzu-Yun Shann, Yi-Hsiang Chang, and Chun-Yi Lee. A deep policy inference Q-network for multi-agent systems. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 1388–1396, 2018. 91

[120] Jakob Foerster, Richard Y Chen, Maruan Al-Shedivat, Shimon Whiteson, Pieter Abbeel, and Igor Mordatch. Learning with opponent-learning awareness. In *Proceedings of the 17th International Conference on Autonomous Agents and MultiAgent Systems*, pages 122–130, 2018. 91

[121] Maruan Al-Shedivat, Trapit Bansal, Yura Burda, Ilya Sutskever, Igor Mordatch, and Pieter Abbeel. Continuous adaptation via meta-learning in non-stationary and competitive environments. In *Proceedings of the 6th International Conference on Learning Representations*, 2018. 91

[122] Dong Ki Kim, Miao Liu, Matthew D Riemer, Chuangchuang Sun, Marwa Abdulhai, Golnaz Habibi, Sebastian Lopez-Cot, Gerald Tesauro, and Jonathan How. A policy gradient algorithm for learning to learn in multiagent reinforcement learning. In *Proceedings of the 38th International Conference on Machine Learning*, pages 5541–5550, 2021. 92

[123] Xiaopeng Yu, Jiechuan Jiang, Wanpeng Zhang, Haobin Jiang, and Zongqing Lu. Model-based opponent modeling. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*, pages 28208–28221, 2022. 92

[124] Michael P Wellman. Methods for empirical game-theoretic analysis. In *Proceedings of the AAAI Conference on Artificial Intelligence*, pages 1552–1556, 2006. 92

[125] Patrick R Jordan, L Julian Schvartzman, and Michael P Wellman. Strategy exploration in empirical games. In *Proceedings of the 9th International Conference on Autonomous Agents and Multiagent Systems*, pages 1131–1138, 2010. 92

[126] L Julian Schvartzman and Michael P Wellman. Stronger cda strategies through empirical game-theoretic analysis and reinforcement learning. In *Proceedings of the 8th International Conference on Autonomous Agents and Multiagent Systems*, pages 249–256, 2009. 92

[127] L Julian Schvartzman and Michael P Wellman. Exploring large strategy spaces in empirical game modeling. *Agent Mediated Electronic Commerce (AMEC 2009)*, page 139, 2009. 92

[128] Bharat Singh, Rajesh Kumar, and Vinay Pratap Singh. Reinforcement learning in robotic applications: A comprehensive survey. *Artificial Intelligence Review*, pages 1–46, 2021. 93

[129] Adish Singla, Anna N Rafferty, Goran Radanovic, and Neil T Heffernan. Reinforcement learning for education: Opportunities and challenges. *arXiv preprint arXiv:2107.08828*, 2021. 93

[130] Siqi Liu, Kay Choong See, Kee Yuan Ngiam, Leo Anthony Celi, Xingzhi Sun, and Mengling Feng. Reinforcement learning for clinical decision support in critical care: Comprehensive review. *Journal of Medical Internet Research*, 22(7):e18477, 2020. 93

[131] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, pages 4909–4926, 2022. 93

[132] Rafael Figueiredo Prudencio, Marcos ROA Maximo, and Esther Luna Colombini. A survey on offline reinforcement learning: Taxonomy, review, and open problems. *IEEE Transactions on Neural Networks and Learning Systems*, 2023. 93

[133] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor. In *Proceedings of the 35th International Conference on Machine Learning*, pages 1861–1870, 2018. 93

[134] Noah Siegel, Jost Tobias Springenberg, Felix Berkenkamp, Abbas Abdolmaleki, Michael Neunert, Thomas Lampe, Roland Hafner, Nicolas Heess, and Martin Riedmiller. Keep doing what worked: Behavior modelling priors for offline reinforcement learning. In *Proceedings of the 8th International Conference on Learning Representations*, 2020. 93

[135] Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. In *Proceedings of the 6th International Conference on Learning Representations*, 2018. 94

[136] Tatsuya Matsushima, Hiroki Furuta, Yutaka Matsuo, Ofir Nachum, and Shixiang Gu. Deployment-efficient reinforcement learning via model-based offline optimization. In *Proceedings of the 9th International Conference on Learning Representations*, 2021. 94, 100

[137] Tianhe Yu, Aviral Kumar, Rafael Rafailov, Aravind Rajeswaran, Sergey Levine, and Chelsea Finn. COMBO: Conservative offline model-based policy optimization. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pages 28954–28967, 2021. 94

[138] Scott Fujimoto, Edoardo Conti, Mohammad Ghavamzadeh, and Joelle Pineau. Benchmarking batch deep reinforcement learning algorithms. *arXiv preprint arXiv:1910.01708*, 2019. 97

[139] Caglar Gulcehre, Ziyu Wang, Alexander Novikov, Tom Le Paine, Sergio Gómez Colmenarejo, Konrad Zołna, Rishabh Agarwal, Josh Merel, Daniel Mankowitz, Cosmin Paduraru, et al. RL unplugged: a suite of benchmarks for offline reinforcement learning. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, pages 7248–7259, 2020. 97

[140] Justin Fu, Aviral Kumar, Ofir Nachum, George Tucker, and Sergey Levine. D4RL: Datasets for deep data-driven reinforcement learning. *arXiv preprint arXiv:2004.07219*, 2020. 97

[141] Robert Tarjan. Depth-first search and linear graph algorithms. *SIAM journal on computing*, 1(2):146–160, 1972. 98

[142] Scott Fujimoto and Shixiang Gu. A minimalist approach to offline reinforcement learning. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pages 20132–20145, 2021. 100

[143] Dmitry Kalashnikov, Alex Irpan, Peter Pastor, Julian Ibarz, Alexander Herzog, Eric Jang, Deirdre Quillen, Ethan Holly, Mrinal Kalakrishnan, Vincent Vanhoucke, et al. Scalable deep reinforcement learning for vision-based robotic manipulation. In *Proceedings of the Conference on Robot Learning*, pages 651–673, 2018. 101

[144] Seunghyun Lee, Younggyo Seo, Kimin Lee, Pieter Abbeel, and Jinwoo Shin. Offline-to-online reinforcement learning via balanced replay and pessimistic Q-ensemble. In *Proceedings of the Conference on Robot Learning*, pages 1702–1712, 2022. 101

[145] Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. In *Proceedings of the 36th International Conference on Machine Learning*, pages 3519–3529, 2019. 102

[146] Paria Rashidinejad, Banghua Zhu, Cong Ma, Jiantao Jiao, and Stuart Russell. Bridging offline reinforcement learning and imitation learning: a tale of pessimism. In *Proceedings of the 35th International Conference on Neural Information Processing Systems*, pages 11702–11716, 2021. 102

[147] Tengyang Xie, Ching An Cheng, Nan Jiang, Paul Mineiro, and Alekh Agarwal. Bellman-consistent pessimism for offline reinforcement learning. In *Proceedings of the 35th Conference on Neural Information Processing Systems*, pages 6683–6694, 2021. 102

[148] Shai Shalev-Shwartz and Shai Ben-David. *Understanding machine learning: From theory to algorithms*. Cambridge university press, 2014. 109

[149] Marc Lanctot, Edward Lockhart, Jean-Baptiste Lespiau, Vinicius Zambaldi, Satyaki Upadhyay, Julien Pérolat, Sriram Srinivasan, Finbarr Timbers, Karl Tuyls, Shayegan Omidshafiei, et al. OpenSpiel: A framework for reinforcement learning in games. *arXiv preprint arXiv:1908.09453*, 2019. 112