A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning

Parth Kachhadia*

April 30, 2023

Abstract

Recent advances in Game Theory have applications in Economics, Smart Power Grids, Intelligent Transportation, Smart Cities, Self-Driving, Multi-agent Robotics, etc. As all of these systems would require a multi-agent rational decision making to derive optimal outcomes. These procedure of achieving maximum payoff, would often lead towards competing or cooperative strategy outcomes for an independent rational agent. One similar rational agent is the learning agent using reinforcement learning to explore and exploit the environment, in order to learn optimal policy. However, similar approach does not work as effectively for multi-agent setting, where since all the agents have shared state-space, one agent's learned policy might over-fit to other agent's policies or may fail to generalize during execution. One reason is, reinforcement learning agent is an independent agent, which interacts only with Markovian and static environment. But, These assumptions does not hold for Multi-agent setting, where agent has to react dynamically based on other agent's behaviours. To accommodate such issues and effectively learn best responses for each agent, authors in [2] introduces new metric Joint-policy correlation and describes a meta-algorithm for general Multi-agent Reinforcement Learning. Based on economic reasoning, this algorithm uses deep reinforcement learning to compute best responses to a distribution over polices and uses empirical game-theoretic analysis to compute new meta-strategy from learned distributions. Approach presented, rather than computing exact best response strategy, computes approximate best response using reinforcement learning. Objective of project is to gain better understanding of the new Joint-policy metric, model the hierarchical approach defined and replicate the experimental results presented of Meta-strategy learning algorithm.

Key terms: Game Theory, Multi-agent Reinforcement Learning, Deep reinforcement learning, Meta-strategy learning, Approximate Best-Response, PSRO, DCH, Empirical Game Theorectic Analysis.

1 Introduction

In recent years, the field of Artificial Intelligence has seen many remarkable successes, from visual systems to Language generation, these Intelligent agents have a capacity to mimic real-world representations and generate optimal outcomes. Similarly, there have been some progresses in the field of game playing, where these rational agents have been given a shot at learning different classical games such as Chess, Go, Atari , etc.

AlphaGo, is a very successful game playing rational agent, who won against a world Go champion, which is a two-player zero-sum game. Authors of AlphaGo, trained their agent, using Reinforcement Learning with Monte-Carlo Tree Search, where agent will learn to make rational decisions by playing millions of games against human expert systems. Studies reveal since it got to play millions of games, It's a lot more than Humanity combined have played till today and this trait allowed AlphaGo agent to come up with strategies which were never discovered before by us. One reason for such advancements is RL, which accounts for α - exploration factor while learning the Optimal Policy for the agent. (Maximizing the total payoff for the player)

Reinforcement Learning has been proved fruitful and many existing algorithms supports these claims. However, this RL agent is an Independent learner. Which means, If our agent is operating in an environment

*MS Computer Science, 2nd year

where there are other rational agents present, we encounter problems. Since, RL is constrained to converge in Static MDP environmets. In Multi-agent settings, Independent RL looses it's convergence guarantees and might overfit to other agent's learned policy and do not generalise well. To overcome such issues in RL, several methods have been applied with different modifications to algorithms, where agents may collect or approximate extra information about their surroundings, compute Joint values, adjust learning rates, etc. These are limited to fully-observable or Repeated Matrix Games.

On key insight from basic RL, is in Multi-agent setting, by ignoring all the other agent's policies, RL agent may have been loosing some Rewards. This applies to both Competitive and Cooperative settings between players. Authors of the original paper, presented a novel matrix, Objective of this matrix is to allow us more insights into the inner workings of RL agent's rational decision making process. There is significant amount of work done on extending the notion of belief states and Bayesian Updates from POMDPs to apply in a n-player general case for Cooperation/Competition.

In this paper, new algorithm has been proposed to leverage several Game Theoretic concepts combined with RL, to first compute approximated Best-Response against distribution over learned policies using deep RL and then using Empirical Game Theoretic Analysis, compute new Meta-strategy for each player of the game.

2 Background - Game Theoretic Concepts & Previous Work

This section discusses, the basic building blocks borrowed from Game Theory, which are necessary for understanding. Several components of the general idea have been rediscovered many times across different research (i.e. AI vs Game Theory) communities, each with slightly different but similar motivations. One aim here is therefore to unify the algorithms and terminology.

A normal-form game is a tuple (π, U, n) where n is the number of players, $\pi = (\pi_1, \ldots, \pi_n)$ is the set of policies (or strategies, one for each player $i \in [[n]]$, where $[[n]] = 1, \ldots, n)$, and $U : \pi \to n$ is a payoff table of utilities for each joint policy played by all players. Extensive-form games extend these formalisms to the multistep sequential case (e.g. poker).

In a multi-agent setting, players aim to maximize their expected utility by either selecting a policy from a set of options π_i or by randomly selecting one from a distribution $\sigma_i \in \delta(\pi_i)$. However, since the quality of these policies σ_i depends on the strategies of other players σ_i, it cannot be determined or evaluated independently. Although every finite extensive-form game can be transformed into an equivalent normal-form, the latter is much larger, making it challenging to apply standard algorithms to the sequential setting.

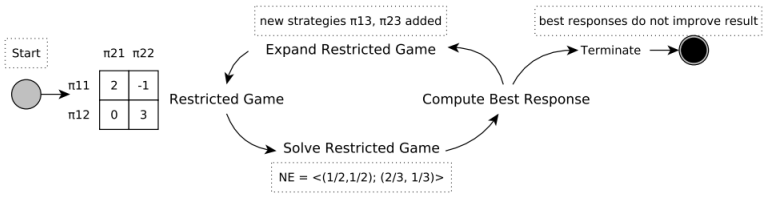


Figure 1: The Double Oracle Algorithm. Figure taken from [10] with authors' permission.

In zero sum games where, $U(\pi) = 0$, Linear programming, regret minimization, fictitious play or Replicator Dynamics can be used. Most of these advances however, deals with two-player case. The Double Oracle (DO) method, solves two (Normal-form) sub-games at stage t, using $\pi^t \in \pi$ at time t. At each time step, similar way G_t will get solved and an equilibrium $\sigma^{*,t}$ will be obtained. So, for all i, $\Pi_i^{t+1} = \Pi_i^t \cup \pi_i^{t+1}$. Although, Double Oracle is guaranteed to converge to an equilibrium in two-player games, in the worst-case it enumerates entire strategy space to find Best-Response. Extensions to the extensive-form games have been developed, but still large state spaces are problematic due to the curse of dimensionality.

One new approach applied by authors of the original papers is the use of Empirical Game Theoretic Analysis, In which an Empirical Game, much smaller in size than full-game will be constructed using each player's meta-strategies and reasoning behind those strategies. Utility values for each of these Joint-Policy

distributions would be estimated and recorded using simulation processes and based on that new meta-strategies for each players can be derived. Here, Meta-strategy allow agents to navigate the strategy space, given the utilities for each possible Joint Policies. Recently, RL methods applied on top of EGTA techniques to validate, meta-strategies generated using Empirical Analysis of the Game.

As mentioned in above sections, recent developments in the field of Deep Learning due to the availability of high-powered computing resources such as GPUs and huge amounts of available data, impressive results have been obtained using deep neural networks for Reinforcement Learning like DQN.

The feasibility of computing approximate responses is higher, and fictitious play is capable of handling approximations. This is also more consistent with natural constraints of bounded rationality. In behavioral game theory, the aim is to predict actions taken by humans, and the responses are intentionally limited to increase the predictive ability. Recently, a deep learning architecture has been employed for this purpose. The concept of level-k thinking is similar to this work, where level k agents respond to level k-1 agents. However, our goals and motivations are different: we use this approach to produce more general policies, not to forecast human behavior. In addition, we focus on sequential settings instead of normal-form games. Lastly, there have been many studies in the literature on co-evolutionary algorithms, specifically on how to mitigate learning cycles and overfitting to current populations.

3 PSRO - Policy Space Response Oracles

The PSRO is a new algorithm that generalizes the Double Oracle and Fictitious Self-Play by taking distribution over policy choices (mixed strategies) instead of actions. Any meta-solver can be utilized to compute a new meta-strategy, and parameterized policies are utilized to generalize across the state space without any domain knowledge. The empirical game used in the PSRO algorithm begins with a single policy, which is usually a uniform random policy. The game is then expanded each epoch by adding policies, which are referred to as "oracles." In order to represent the meta-game, a single policy (uniform random) is initially used and policies (known as "oracles") are added in each epoch, approximating the best responses to the meta-strategy of the other players. In partially observable multiagent environments where the other players are fixed, the environment becomes Markovian and finding the best response involves solving a form of MDP. In such cases, any reinforcement learning algorithm can be used, and the authors utilize deep neural networks due to their recent success in reinforcement learning.

Algorithm 1: Policy-Space Response Oracles

input : initial policy sets for all players Π
Compute exp. utilities U^{Π} for each joint $\pi \in \Pi$
Initialize meta-strategies $\sigma_i = \text{UNIFORM}(\Pi_i)$
while *epoch* e *in* $\{1, 2, \cdots\}$ **do**
  for *player* $i \in [[n]]$ **do**
    for *many episodes* **do**
      Sample $\pi_{-i} \sim \sigma_{-i}$
      Train oracle π_i' over $\rho \sim (\pi_i', \pi_{-i})$
    $\Pi_i = \Pi_i \cup \{\pi_i'\}$
  Compute missing entries in U^{Π} from Π
  Compute a meta-strategy σ from U^{Π}
Output current solution strategy σ_i for player i

Algorithm 2: Deep Cognitive Hierarchies

input : player number i, level k
while *not terminated* **do**
  CHECKLOADMS($\{j | j \in [[n]], j \neq i\}, k$)
  CHECKLOADORACLES($j \in [[n]], k' \leq k$)
  CHECKSAVEMS($\sigma_{i,k}$)
  CHECKSAVEORACLE($\pi_{i,k}$)
  Sample $\pi_{-i} \sim \sigma_{-i,k}$
  Train oracle $\pi_{i,k}$ over $\rho_1 \sim (\pi_{i,k}, \pi_{-i})$
  if *iteration number* **mod** $T_{ms} = 0$ **then**
    Sample $\pi_i \sim \sigma_{i,k}$
    Compute $u_i(\rho_2)$, where $\rho_2 \sim (\pi_i, \pi_{-i})$
    Update stats for π_i and update $\sigma_{i,k}$
Output $\sigma_{i,k}$ for player i at level k

As a basic Reinforcement Learning algorithm works on Bellman's equations,

$q*(s,a) = E[R_{t+1} + \gamma * max'_a q*(s', a')$.
where, γ is the discount factor for t- stage updates. Which using dynamic programming optimises the cumu-

lative reward for the goal described and through learning obtains optimal policy $\pi*$ which contains $q*(s,a)$ state-action pair values. Applying these concepts to our current task where there are n- Players, In each episode, one player is set to learning the Optimal Policy mode to train π_i', while a fixed policy will be played on behalf of all the other $n-1$ - Players from their respective Meta-strategies (π_i σ_i) in the game. After each epoch, learned policies will be collected in Π_i and expected Utilities for all new policy combinations will be computed using simulation to generate U^{Π}. From, This point using these utilities for all Joint-policies from n - Players, By applying Meta-solvers to the Empirical Games, Meta-strategies will be obtained for the general n-Player case. Where these Joint-Policies would account for Competition or Cooperation as required by the Game.

The Fictitious play algorithm is not specific to the policies it is responding to, so it can only improve the meta-strategy distribution by repeatedly generating the same best responses. In contrast, the Double Oracle algorithm's responses to equilibrium strategies can lead to overfitting in the n-player or general-sum case and may not generalize to parts of the space not covered by any equilibrium strategy in the zero-sum case. These issues are not desirable when computing general policies that should perform well in any situation. To mitigate these problems, we strike a balance by using meta-strategies with full support that enforce γ exploration over policy selection.

4 DCH - Deep Cognitive Hierarchies

Although PSRO's generality is desirable, the RL step can take a long time to converge to a good response, especially in complex environments. Additionally, relearning basic behavior may be necessary in each epoch, and it may be necessary to run many epochs to obtain oracle policies that can recursively reason through deeper levels of contingencies. To address these issues, the authors propose a practical parallel form of PSRO called Deep Cognitive Hierarchy (DCH). Instead of an unbounded number of epochs, a fixed number of levels is chosen in advance, and nK processes are started in parallel for an n-player game. Each process trains a single oracle policy for a player and level k, and updates its own meta-strategy, periodically saving each to a central disk. Each process maintains copies of all other oracle policies at the current and lower levels, as well as the meta-strategies at the current level, periodically refreshed from a central disk.

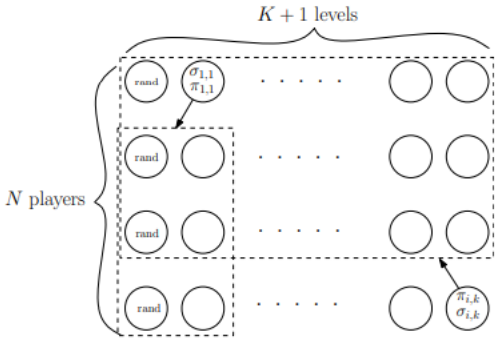


Figure 2: Overview of DCH

DCH approximates PSRO by trading accuracy for practical efficiency and scalability. In DCH, each process stores nK policies of fixed size, and n meta-strategies of size bounded by k $\leq$ K, resulting in an asymptotic reduction in total space complexity. Therefore the total space required is $O(nK(nK + nK))$ $= O(n^2K^2)$. This is possible is due to the use of decoupled meta-solvers, which compute strategies online without requiring a payoff tensor U^{Π}, which we describe now.

5 Meta-Solvers - Projected Replicated Dynamics

To find the best meta-strategy for each player in a given game, we employ a meta-strategy solver that takes in the Empirical Game (Π, U^{Π}) and produces a meta-strategy σ_i for each player i. Three solvers authors tried were Regret Matching, Hedge, and Projected Replicator Dynamics (PRD). These solvers accumulate values for each policy and an aggregate value based on all players' meta-strategies. We denote player i's expected value given all players' meta-strategies and the current empirical payoff tensor U^{Π} as $u_i(\sigma)$. Similarly, $u_i(\pi_{i,k}, \sigma_{i)}$ is the expected utility when player i plays their k^{th} policy and the other players play with their meta-strategy σ_i. To ensure that each policy is selected with a probability of at least $\gamma/K + 1$, strategies use an exploration parameter γ. Regret Matching and Hedge are previously established algorithms, while PRD is a new solver introduced in this study. In essence, Replicator Dynamics is a differential equation system which describes how Population of strategies or (replicators) will evolve in the given environment across time. They replicate biological selection principle, comparing Utility of strategy with average of the entire population. More specifically, it can be expressed as $\frac{\delta x_k}{\delta t} = x_k * [(Ax)_k - x^T Ax]$. Here, x_k represents the density of strategy $\pi_{i,k}$ in the population ($\sigma_i(\pi_{i,k})$ for a given k), A is the payoff matrix which describes the different payoff values each individual replicator receives when interacting with other replicators in the population. This common formulation represents symmetric games, and typical examples of dynamics are taken from prisoner's dilemma, matching pennies, and stag hunt games. Asymmetric replicator dynamics are applied to n-player games normal form games, e.g. in two players using payoff tables A and B, where $A! = B^T$. Examples are the infamous prisoner's dilemma and the Rock-Scissors-Paper games, in which both agents are interchangeable. When using the asymmetric replicator dynamics, the change in probabilities for the kth component of meta-strategies of two players are calculated as part of the PRD algorithm. In this context it means we now have two players that come from different populations:

$\frac{\delta x_k}{\delta t} = x_i * [(Ay)_k - x^T Ay]$ and $\frac{\delta y_k}{\delta t} = y_i * [(x^T B)_k - x^T By]$,

where x corresponds to the row player and y to the column player. In general, there are n tensors, representing the utility to each player for each outcome. In this paper we use a new projected replicator dynamics that enforces exploration by putting a lower bound on the probability on x_k and y_k.

In online learning, experts algorithms receive information about every option at each round, while bandit algorithms only get feedback for the option that was selected. Decoupled meta-solvers are sample-based adversarial bandits applied to games and are used in partial information settings. Empirical strategies can converge to Nash equilibria in certain game classes due to the folk theorem. The authors experimented with three decoupled meta-solvers: decoupled regret-matching, Exp3 (decoupled Hedge), and decoupled PRD. To ensure unbiased estimates, exploratory strategies with γ of the uniform strategy mixed in are used, and for decoupled PRD, running averages are maintained for the overall average value and value of each option. In contrast to PSRO, DCH obtains one sample at a time and updates the meta-strategy periodically from online estimates.

6 Joint-Policy Correlation Matrix

One of the starting point for this research is this addition of new Measurement for Multi-agent interactive settings, Where authors to better understand and also model the inter-policy interactions, where for example, two players may compete / Cooperate to maximize both of their rewards, or use policy learned by another player to improve player's total Utility, is Joint-Policy Correlation for RL agents. By measuring potential loss for an Independent Learning agent due to it's incapacity to gain/ translate information from other players in the game. In paper, for two-player symmetric case, a metric is shown as in Figure here, An average proportional loss in Rewards can be computed as $R = (\bar{D} - \bar{O})/\bar{D}$, where $\bar{D}$ is the mean value of the diagonals and $\bar{O}$ is the mean for the Off-diagonal. $\bar{D} = 30.44$, $\bar{O} = 20.03$, R = 0.342. Even in a simple domain with almost full observability (small2), an independently-learned policy could expect to lose 34.2% of its reward when playing with another independently-learned policy. In the other variants (gathering and pathfind), we observe no JPC problem, presumably because coordination is not required and the policies are independent.

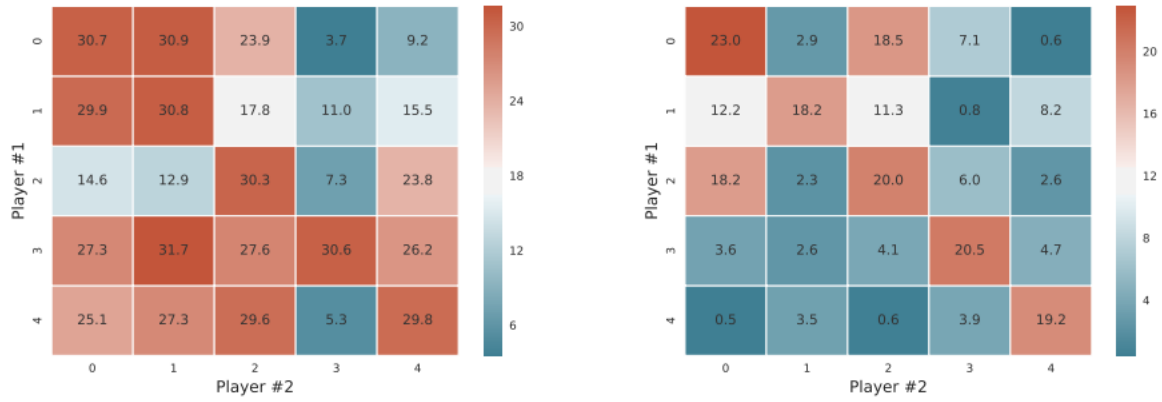


Figure 3: Example JPC matrices for InRL on Laser Tag small2 map (left) and small4 (right).

Values are obtained by running D instances of the same experiment, differing only in the seed used to initialize the random number generators. Each experiment $d \in [[\mathrm{D}]]$ (after many training episodes) produces policies (π_1^d, π_2^d). The entries of each DD matrix shows the mean return over $T = 100$ episodes, $\Sigma_{t=1}^{T} \frac{1}{T}(R_1^t + R_2^t)$, obtained when player 1 uses row policy $\pi_1^{d_i}$ and player 2 uses column policy $\pi_2^{d_j}$. Hence, entries on the diagonals represent returns for policies that learned together (i.e., same instance), while off-diagonals show returns from policies that trained in separate instances. Results are summarized in Table 1.

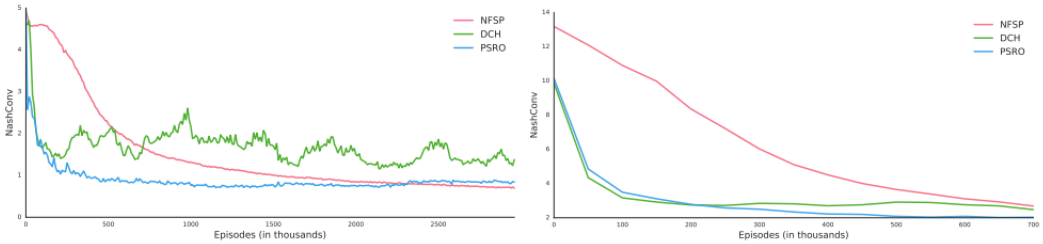
Environment	Map	InRL			DCH(Reactor, 2, 10)			JPC Reduction
		$\bar{D}$	$\bar{O}$	R_{-}	$\bar{D}$	$\bar{O}$	R_{-}	
Laser Tag	small2	30.44	20.03	0.342	28.20	26.63	0.055	28.7 %
Laser Tag	small3	23.06	9.06	0.625	27.00	23.45	0.082	54.3 %
Laser Tag	small4	20.15	5.71	0.717	18.72	15.90	0.150	56.7 %
Gathering	field	147.34	146.89	0.003	139.70	138.74	0.007	–
Pathfind	merge	108.73	106.32	0.022	90.15	91.492	< 0	–

Table 1: Summary of JPC results in first-person gridworld games.

We see that a (level 10) DCH agent reduces the JPC problem significantly. On small2, DCH reduces the expected loss down to 5.5%, 28.7% lower than independent learners. The problem gets larger as the map size grows and problem becomes more partially observed, up to a severe 71.7% average loss. The reduction achieved by DCH also grows from 28.7% to 56.7%.

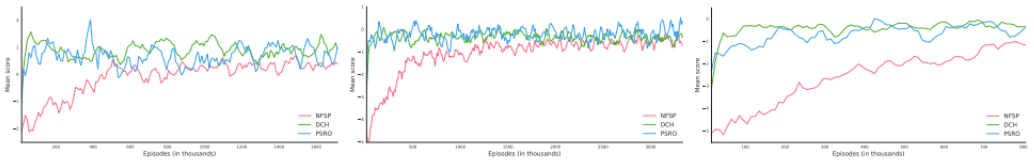
7 Experimental Results

Reactors are used in all experiments to train oracles, and have demonstrated outstanding performance in Atari game-playing. Reactor leverages Retrace(λ) for off-policy policy evaluation and β-Leave-One-Out policy gradient for policy updates. Additionally, it supports recurrent network training which is useful in matching online experiences to those observed during training.



(a) 2 players (b) 3 players

Figure 5: Exploitability for NFSP x DCH x PSRO.



(a) Random bots as ref. set (b) 2-player CFR500 bots as ref. set (c) 3-player CFR500 bots as ref. set

Figure 6: Evaluation against fixed set of bots. Each data point is an average of the four latest values.

The authors note that in their experiments, PSRO and DCH converge faster than NFSP in the beginning of the training process, which they attribute to the use of a better initial meta-strategy. However, the convergence curves eventually level off, with DCH being most affected in two-player games due to the asynchronous nature of updates. NFSP eventually converges to a lower exploitability in later episodes, which the authors believe is due to its ability to learn a more accurate mixed average strategy at deeper states in the game tree, particularly in poker. On the other hand, PSRO and DCH are better at recognizing flaws in weaker opponent policies and dynamically adapting to exploitative responses during the episode, leading to higher performance against fixed players. While NFSP computes a safe equilibrium, PSRO and DCH may be sacrificing convergence precision for the ability to adapt to a wider range of play observed during training, resulting in a more robust counter-strategy.

8 Implementation Details

For Implementation, Authors of Original papers, The deepmind group has developed and open sourced a Library called OpenSpiel, Where main focus is to provide implementations for recent Deep Learning, Reinforcement Learning and Game Theoretic algorithms and techniques. This platform, provides Game related mechanisms, environment for learning agents, simulation mechanism to compute expected Utilities for the players.

PSRO is already provided with meta-solver based on Projected Replicated Dynamics as described above. But, DCH- based architecture for PSRO, was implemented as the part of the project. Complete implemetation for n-Player General case Partial-Observable setting can be found at,

https://github.com/pathu007/PSRO-DCH—Multi-Agent-RL-with-Game-Theory..git

9 Conclusion & Future Work

The authors of this paper aim to address a major issue in independent reinforcement learners called Joint Policy Correlation (JPC) that limits their usefulness. They introduce a generalized algorithm for multi-agent reinforcement learning that subsumes several previous algorithms. The approach is based on game theory and seeks to minimize JPC in partially-observable coordination games, while also providing robust counter-strategies for common competitive imperfect information games. The authors propose that their technique offers a form of "opponent/teammate regularization," which can be seen as a game-theoretic

concept aimed at creating more stable and reliable outcomes in multiagent environments. They also emphasize the game-theoretic foundations of their approach and hope that it inspires further investigation into algorithm development for multiagent reinforcement learning.

The paper concludes with several areas of future research, including maintaining diversity among oracles via loss penalties based on policy dissimilarity, using successor features to create generalized (transferable) oracles over similar opponent policies, and exploring fast online adaptation and computational theory of mind, which are related to game-theoretic concepts like adaptation and mental model building in competitive environments.

References

[1] D. Bloembergen, K. Tuyls, D. Hennes, and M. Kaisers. Evolutionary dynamics of multi-agent learning: A survey. *Journal of Artificial Intelligence Research*, 53:659–697, 08 2015.

[2] M. Lanctot, V. Zambaldi, A. Gruslys, A. Lazaridou, K. Tuyls, J. Perolat, D. Silver, and T. Graepel. A unified game-theoretic approach to multiagent reinforcement learning. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[3] R. Sharma and M. Gopal. Synergizing reinforcement learning and game theory—a new direction for control. *Applied Soft Computing*, 10(3):675–688, 2010.

[4] K. G. Vamvoudakis, H. Modares, B. Kiumarsi, and F. L. Lewis. Game theory-based control system algorithms with real-time reinforcement learning: How to solve multiplayer games online. *IEEE Control Systems Magazine*, 37(1):33–52, 2017.

[5] W. Walsh, R. Das, G. Tesauro, and J. Kephart. Analyzing complex strategic interactions in multi-agent systems. 01 2002.