A Report on “A Unified Game-Theoretic Approach to Multiagent Reinforcement Learning”

prepared by:
Shirin Jamshidi – 810199570
Mahya Shahshahani – 810199598
Ouldouz Neysari – 810199505
Mohammad Mashreghi - 810199492

Abstract

Achieving general intelligence in artificial agents requires mastering multiagent reinforcement learning (MARL), where agents learn to interact in shared environments. Independent reinforcement learning (InRL), the simplest MARL form, risks overfitting to other agents' policies, impairing generalization. This paper introduces joint-policy correlation[1] to measure this effect and proposes an algorithm combining deep reinforcement learning with empirical game-theoretic analysis[2] for better policy selection. The algorithm generalizes previous methods such as double Oracle[3], by using decoupled meta-solvers[4], the implementation is scalable, reducing memory needs. Its effectiveness is shown in grid world coordination games and poker.

[1] A metric to show how much the policies of different agents are interdependent or correlated during training.

[2] Involves studying strategies in complex games through simulations, instead of analyzing the entire strategy space. (empirical: تجربی)

[3] Iteratively finds equilibrium strategies in two-player games by expanding a subset of strategies at each step. Initially, a limited set of strategies is considered, and an equilibrium is computed for this subset. Each player then adds a best response to the equilibrium strategy of the other player from the full strategy space, iteratively refining the strategy set.

[4] Meta-solvers are used to manage and solve meta-strategies(high-level strategies derived from combinations or distributions of lower-level strategies. An overarching strategy determining which other strategies to use in a given situation) efficiently.

1. Introduction

Deep reinforcement learning (Deep RL) blends deep learning with reinforcement learning to create decision-making policies. Historically applied to single agents, its success has sparked interest in MARL, where multiple agents learn and interact, either competitively, cooperatively, or in mixed settings. In InRL, each agent treats interactions as part of its local environment, leading to non-stationary and non-markovian[5] conditions and overfitting to other agents' policies, which hinders generalization.

The paper outlines the progress and methods in handling partial observability in MARL, highlighting the use of regret minimization[6] in domain-specific[7] abstractions and deep learning in adversarial settings, belief states[8], and Bayesian[9] updating in cooperative problems, and the necessity of approximate methods to manage the complexity of these models. In this paper, policies are represented as separate neural networks and there is no sharing of gradients nor architectures among agents.

2. Background and Related Work

Key constructs for the algorithm include normal-form and extensive-form(sequential) games, where players choose policies to maximize expected utility like poker. Various algorithmic approaches are explored, including linear programming[10], fictitious play, replicator dynamics, and regret minimization, tailored for both zero-sum and general-sum[11] games. The double oracle algorithm is highlighted for solving subgames iteratively in normal-form games, ensuring convergence to equilibria but facing challenges in large strategy spaces. Extensions to

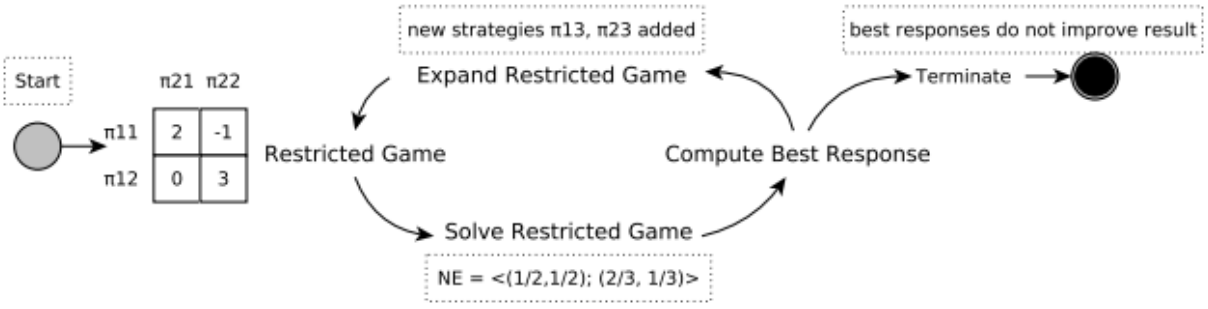


Figure1: The Double Oracle Algorithm

[5] A system that its future states depend not only on the current state but also on the history of past states.

[6] Aims to minimize the difference between the actual accumulated reward and the best possible accumulated reward in hindsight.

[7] It involves leveraging insights and techniques that are particularly effective or relevant to a specific domain or type of problem.

[8] Probabilistic representations of the possible states of the environment given the agent's observations and actions.

[9] Method for updating the probability distribution of a hypothesis (in this case, the belief state) as new evidence or information becomes available.

[10] A mathematical method for optimizing a linear objective function subject to linear equality and inequality constraints.

[11] A class of games in game theory where the sum of payoffs to all players is not necessarily zero.

extensive-form games address computational complexities using empirical game-theoretic analysis (EGTA), which constructs smaller empirical games to explore strategic meta-strategies and meta-reasoning[12] efficiently, crucial in scenarios like trading agent competitions and automated bidding.

The essay also delves into recent advancements merging reinforcement learning with EGTA to discover effective strategies, utilizing modern deep learning techniques to approximate optimal responses in complex environments. Examples include applications in Atari games and AlphaGo(combination of deep RL with Monte Carlo tree search), demonstrating the power of deep RL in decision-making contexts. The discussion contrasts this with behavioral game theory, which focuses on predicting human actions constrained by cognitive limitations. It acknowledges parallels with level-k[13] and cognitive hierarchy[14] models, while emphasizing a distinct goal of generating robust, general policies rather than predicting human behavior. Finally, the essay notes research into co-evolutionary algorithms aimed at mitigating overfitting and enhancing adaptability in evolving populations, highlighting ongoing efforts to improve learning cycles in strategic environments[15]. The paper uses the setup as a means to produce more general policies rather than to predict human behavior. Furthermore, they consider the sequential setting rather than normal-form games.

3. Policy-Space Response Oracles

The paper introduces the policy-space response oracles (PSRO) algorithm as an extension of the Double Oracle and Fictitious Self-Play methods. PSRO operates with policies as choices within a meta-game, unlike traditional approaches that focus on actions. One key advantage is its flexibility in using various meta-solvers to compute new meta-strategies. The algorithm effectively utilizes parameterized policies (function approximators) to generalize across different states without needing specific domain knowledge, enhancing its applicability in diverse environments.

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

Figure2: PSRO Algorithm

[12] It involves higher-order thinking about how one makes decisions, assesses information, and strategizes in various contexts.

[13] Level-k models propose that individuals reason about the behavior of others in a hierarchical manner, where each level represents a different depth of strategic thinking:

Level-0: Players choose actions without considering the actions of others, often selecting randomly or based on simple rules.

Level-1: Players anticipate Level-0 players' actions and respond optimally to them.

Level-2: Players anticipate Level-1 players' responses and adjust their strategies accordingly.

...

[14] Cognitive hierarchy models extend the concept of level-k models by incorporating the idea of cognitive abilities and reasoning depths. More flexible than level-k models because they allow for a continuous distribution of cognitive types rather than discrete levels.

[15] Situations that each participant's payoff or outcome is influenced by the actions of others, too.

The process described in Algorithm 1 represents the meta-game as an empirical game. Initially, it starts with a single policy π_0, often a uniform random policy. With each epoch, new policies (oracles) are added that approximate the best responses to the meta-strategy of the other players.

In partially observable multiagent environments, fixing the other players' strategies makes the environment Markovian. This simplifies computing the best response to solving a Markov Decision Process (MDP). Any reinforcement learning (RL) algorithm can be used for this, and the paper uses deep neural networks due to their recent success in RL.

Algorithm Steps:

1. Initialization: Start with a single policy π_0.
2. Epoch Process:

   In each episode, set one player to learning mode to train a new policy π_0'.
   Sample fixed policies for the other players from their meta-strategies σ_{-i} .
3. Policy Update: At the end of the epoch, add the new oracles to the policy sets Π_i .
4. Utility Calculation: Compute expected utilities for the new policy combinations via simulation and add them to the empirical utility tensor U_Π, which has exponential complexity in $|\Pi||$.
5. Policy Space Update: Define $\Pi_T=\Pi_{T-1}\cup\{\pi0'\}$ as the updated policy space, and $|\Sigma i|=|\Pi_T|$ for all $i\in\{1,\ldots,n\}$.

Special cases of PSRO:

Iterated Best Response: $\sigma_{-i}=(0,0,\ldots,1,0)$
Independent RL and Fictitious Play: $\sigma_{-i}=(0,0,\ldots,0,1)$ and $\sigma_{-i}=(1/K\ ,1/K\ ,\ldots,1/K\ ,0)$ respectively, where $K=|\ \Pi_{T-1,\ -i}|$
Double Oracle: n=2 and σ_T set to a Nash equilibrium of the meta-game $(\Pi_{T-1},U_{\ \Pi T-1})$

The paper explores the use of non-fixed meta-solvers to avoid overfitting and generalization issues present in equilibrium responses.

Here we come with an example to understand better what the paper is talking about.

Example: Self-Driving Cars

1. Initialization: Start with each self-driving car using a random driving policy π_0.
2. Epoch Process:

   For each round (epoch), one car is set to learning mode to train a new driving policy π_0' using deep neural networks.

   Fixed driving policies are sampled for the other cars from their current meta-strategies.
3. Policy Update: Add the new driving policy to the set of existing policies.

4. Utility Calculation: Simulate driving scenarios with the new policies and compute the expected utilities, updating the empirical utility tensor.
5. Policy Space Update: Update the policy space to include the newly learned policy.

Special cases:

- Iterated Best Response: The car learns the best response to the current scenario.
- Independent RL and Fictitious Play: The car independently learns a new policy or follows a mixed strategy based on previous experiences.
- Double Oracle: Two cars use the learned strategies, focusing on achieving a Nash equilibrium in their interactions.

By using non-fixed meta-solvers, the cars avoid overfitting to specific scenarios and develop policies that generalize well across different driving contexts. This approach helps balance exploration and exploitation, ensuring robust and adaptive driving strategies.

Meta-Solvers:

The meta-strategy solver in the described process takes the empirical game (Π, U_Π) as input and generates a meta-strategy σ_i for each player i. These solvers accumulate values for each policy and an aggregate value based on all players' meta-strategies.

1. **Regret-Matching**[16] and **Hedge**[17]: These are straightforward applications of existing algorithms.

2. **Projected Replicator Dynamics (PRD)**:

Involves asymmetric replicator dynamics for two players, with empirical payoff matrices $U_\Pi=(A,B)$. The change in probabilities for the k-th component of meta-strategies $(\sigma_1,\sigma_2)=(x,y)$ is given by:

$$\frac{dx_k}{dt} = x_k[(Ay)_k - x^T Ay]$$

$$\frac{dy_k}{dt} = y_k[(x^T B)_k - x^T By]$$

Discretized Updates:

To simulate replicator dynamics in practice, discretized updates with step-size δ are used:

[16] adjusts strategy choices based on minimizing cumulative regrets from past decisions.

[17] an algorithm that balances exploration and exploitation by assigning weights to strategies based on past performance.

$$x \leftarrow P(x + \delta \frac{dx}{dt})$$

$$y \leftarrow P(y + \delta \frac{dy}{dt})$$

Projection Operator P(·):

- Ensures exploration by maintaining the exploratory nature of $\sigma_i(\pi_{i,k}) \geq \frac{\epsilon}{k+1}$.
- Defined as:

$$P(x) = \arg \min_{x' \in \Delta^{\epsilon}_{K+1}} \|x' - x\|$$

- $\Delta^{\epsilon}_{K+1} = \{ x | x_k \geq \frac{\epsilon}{k+1}, \sum_k x_k = 1\}$ is the ϵ- exploratory simplex of size K+1.

The PRD approach directs exploration, differing from standard replicator dynamics that include isotropic diffusion or mutation terms, which assume undirected and unbiased evolution.

Deep Cognitive Hierarchies (DCH)

DCH is a practical parallel form of PSRO designed to handle complex environments more efficiently.

In PSRO, the reinforcement learning (RL) step can take a long time to converge, and basic behaviors may need to be relearned each epoch. DCH addresses these issues by using parallel processes and fixed levels.

Process:

- For an n player game, nK processes are started in parallel.
- Level 0 agents use uniform random policies.
- Each process trains a single oracle policy $\pi_{i,k}$ for player i and level k, updating its meta-strategy $\sigma_{i,k}$ and saving periodically to a central disk.
- Each process maintains copies of other oracle policies $\pi_{j,k}{}'$ (for $k' \leq k$) and meta-strategies $\sigma_{-i,k}$ which are refreshed periodically from the central disk.

Key Features:

- **Empirical Payoff Tensor:** DCH does not store U_Π explicitly; meta-strategies are updated online.
- **Approximation:** DCH approximates PSRO by using slightly outdated copies of policies and meta-strategies, trading accuracy for efficiency and scalability.
- **Space Complexity:** For K policies and n players, PSRO requires Kn space for the empirical payoff tensor. In DCH, each process stores nK policies and n meta-strategies, resulting in total space complexity $O(n^2K^2)$

Decoupled Meta-Strategy Solvers[18]

In online learning, "full information" algorithms receive feedback on each option every round, while "partial information" (bandit) algorithms receive feedback only on the chosen option.

Decoupled meta-solvers function as sample-based adversarial bandits applied to games, converging to Nash equilibria in certain classes of games due to the folk theorem[19].

Solvers:

1. **Decoupled Regret-Matching**: Adjusts strategies based on regret, aiming to minimize future regret.
2. **Exp3 (Decoupled Hedge)**: Balances exploration and exploitation by using exponential weighting.
3. **Decoupled Projected Replicator Dynamics (PRD)**: Maintains running averages for overall and per-policy values, using periodic online updates to adjust meta-strategies.

4. Experiments

The experiments use the Reactor[20] model for learning, which employs advanced techniques like Retrace(λ)[21] for evaluating policies and β-Leave-One-Out[22] policy gradients for updating policies. Reactor also supports recurrent network training[23], crucial for handling sequential data effectively. While the action spaces for each player in the experiments are identical, the algorithms can handle non-identical action spaces as well.

[18] Decoupled meta-solvers operate independently of each other, with each making decisions based solely on its own local information or context.

[19] The folk theorem states that in repeated strategic interactions, players can achieve a wide range of outcomes, including Nash equilibria, through various strategies and behaviors.

[20] an advanced approach in RL that integrates sophisticated techniques like off-policy evaluation and policy gradient methods, leveraging these to achieve high performance in challenging environments such as Atari games.

[21] a technique commonly used in reinforcement learning for off-policy evaluation. It combines eligibility traces with importance sampling to estimate the expected return of a target policy using data collected by a different behavior policy. The parameter λ in Retrace(λ) adjusts the balance between bias and variance in the estimation process. It is particularly useful in scenarios where policies differ during training and evaluation, allowing for efficient learning from historical data.

[22] a strategy employed in policy gradient methods within reinforcement learning, where the objective function is iteratively computed by excluding one action or component at a time. This technique aims to refine policy updates by systematically evaluating the influence of individual actions on the expected return.

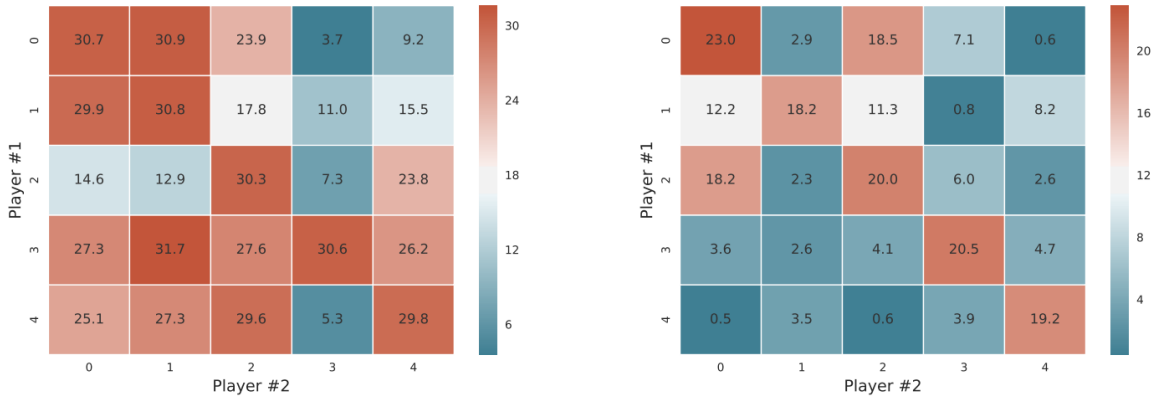
[23] Recurrent network training refers to the process of training neural networks with recurrent connections, allowing them to retain and utilize temporal information across sequential data. This technique is crucial for applications requiring memory and context awareness over time, such as natural language processing and time series prediction.

First-Person Gridworld Games. In these first-person gridworld games, agents operate within a limited view, executing various movements and actions. The games include different objectives like tagging other agents, collecting apples, or reaching destinations. Each variant has unique rules but shares the common framework of limited visibility, simultaneous actions, and episodes of 1000 steps.

Leduc Poker. Leduc Poker involves a small deck of six cards and consists of two rounds of betting, with limited raises and antes to create the pot. Players start with one private card, and a public card is revealed after the first betting round. The game's state, including the private and public cards and action history, is represented using one-hot encodings. This simplified yet structured setup makes Leduc Poker an ideal benchmark for testing and developing poker AI strategies.

4.1 Joint Policy Correlation in Independent Reinforcement Learning.

Joint Policy Correlation (JPC) matrices are used to evaluate the effect of overfitting in independent reinforcement learners by analyzing the performance of policies across multiple instances of the same experiment with different random seeds. In symmetric two-player games with non-negative rewards, the matrix entries show the mean returns over 100 episodes for combinations of policies from different instances. Diagonal entries represent policies trained together, while off-diagonal entries represent policies trained separately. This approach helps in understanding how well policies generalize to different training instances and how overfitting might affect their performance.



The average proportional loss in reward, R^- derived from JPC matrices, measures the extent of reward loss when independently-learned policies play together. It is calculated using the formula:

$$R^- = \frac{D^- - O^-}{D^-}$$

where D^{-} represents the mean reward for policies trained together and O^{-} represents the mean reward for policies trained independently. In a given example, a 34.2% loss in reward is observed in a simple domain with almost full observability, highlighting the coordination problem of independent learners. This issue is not present in game variants that do not require coordination. Similar issues are found using DQN[24] as the training algorithm, demonstrating the broader relevance of the problem.

Using a DCH agent significantly mitigates the JPC problem, with reductions in expected reward loss ranging from 28.7% to 56.7% depending on the map size and observability. The meta-strategy, which involves a fully-mixed strategy, is crucial for minimizing JPC losses. Even at lower levels (e.g., level 5 and level 3), DCH agents achieve substantial reductions in JPC, with level 5 performing nearly as well as level 10. This demonstrates the importance and effectiveness of hierarchical and coordinated strategies in reinforcement learning to address coordination problems between independently learned policies.

4.2 Learning to Safely Exploit and Indirectly Model Opponents in Leduc Poker

This paper evaluates Leduc poker policies using two metrics: performance against fixed players and NashConv. Benchmark algorithms like Counterfactual Regret (CFR)[25] minimization is used for comparison.

Policies are tested against three fixed players: a random player, CFR's average strategy after 500 iterations ("cfr500"), and a purified version of CFR ("cfr500pure") that always selects the most probable action.NashConv measures how close a policy is to a Nash equilibrium. It is calculated as:

The effect of various meta-strategies and exploration parameters on the performance of our reinforcement learning agents was analyzed. The paper measured the mean area-under-the-curve (MAUC) of the NashConv values for the last 32 values in the NashConv graph, with an exploration rate[26] (γ) of 0.4. (The figure)

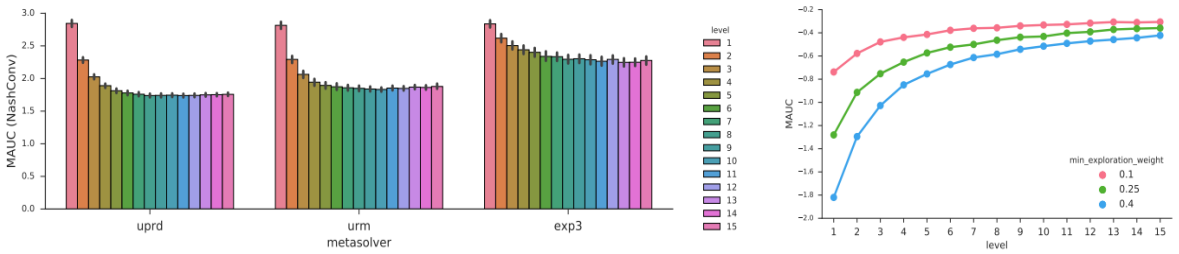
---

[24] Deep Q-Network (DQN) is a type of reinforcement learning algorithm that combines deep learning techniques with Q-learning, enabling neural networks to approximate optimal action-selection policies in complex environments. DQN has been influential in advancing autonomous decision-making in domains such as gaming, robotics, and recommendation systems.

[25] Counterfactual Regret Minimization (CFR) is an algorithm used in game theory to find optimal strategies in extensive-form games. It iteratively minimizes regret for actions not taken, with the goal of converging to a Nash equilibrium. CFR has been notably applied in developing AI for imperfect information games such as poker

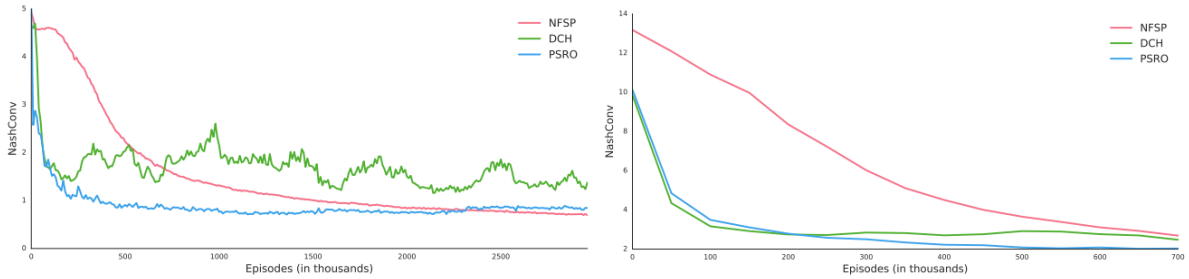
[26] exploration rate (γ) in reinforcement learning represents the probability of an agent exploring new actions rather than exploiting known rewarding actions. A higher exploration rate encourages more exploration of the environment, which can help in discovering optimal strategies. Conversely, a lower exploration rate focuses more on exploiting the best-known actions, potentially leading to faster convergence on suboptimal strategies. The optimal exploration rate balances exploration and exploitation for efficient learning.

The analysis revealed that the decoupled replicator dynamics strategy performed the best, followed by decoupled regret-matching and Exp3. Observations show that higher levels generally resulted in lower NashConv values, although with diminishing improvements. For exploitation, we found that an exploration rate of γ = 0.1 was optimal.



(a) (b)

A comparison to Neural Fictitious Self-Play (NFSP)[27] , an implementation of fictitious play in sequential games using reinforcement learning, is conducted. It is important to note that NFSP, PSRO, and DCH are all sample-based learning algorithms utilizing general function approximation, whereas CFR is a tabular method requiring a full game-tree pass per iteration. The figure below shows NashConv graphs for {2-3}-player and the following figure shows the performance vs. fixed bots:



(a) 2 players (b) 3 players

DCH and PSRO initially converge faster than NFSP, possibly due to more effective meta-strategies. Over time, their convergence levels off, particularly in two-player settings for DCH, which may be impacted by asynchronous updates. NFSP eventually achieves lower exploitability in later stages, as it excels at learning mixed average strategies deep in the game tree, which is advantageous in poker. Nonetheless, PSRO and DCH perform better against fixed opponents, likely due to their ability to identify and exploit weaknesses in the opponents' strategies. While NFSP aims for a

[27] Neural Fictitious Self-Play (NFSP) is an algorithm that combines fictitious play with deep reinforcement learning. It simulates the process of learning through repeated play against a mixture of past opponents, represented by neural networks. This allows for the approximation of optimal strategies in sequential games, facilitating learning in environments with complex, high-dimensional state spaces.

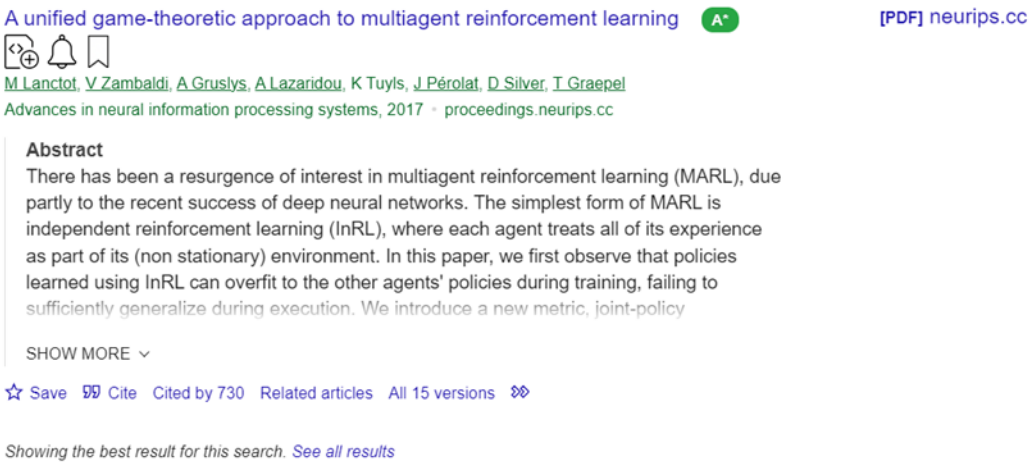
stable equilibrium, PSRO and DCH focus on adaptability and dynamic exploitation, leading to robust counter-strategies[28].

5.Conclusion

This paper addresses the issue of joint policy correlation (JPC) in independent reinforcement learners, which restricts their effectiveness. It introduces a generalized algorithm for multiagent reinforcement learning that incorporates several existing methods. The experiments demonstrate that PSRO/DCH can create policies that significantly reduce JPC in partially observable coordination games and develop robust counter-strategies for competitive imperfect information games. The versatility of PSRO/DCH acts as "opponent/teammate regularization," highlighting its practical applicability.

Additional Work:

The paper known as PSRO, a key concept is a restricted game with estimated payoffs, which acts as an approximation of the underlying full game.



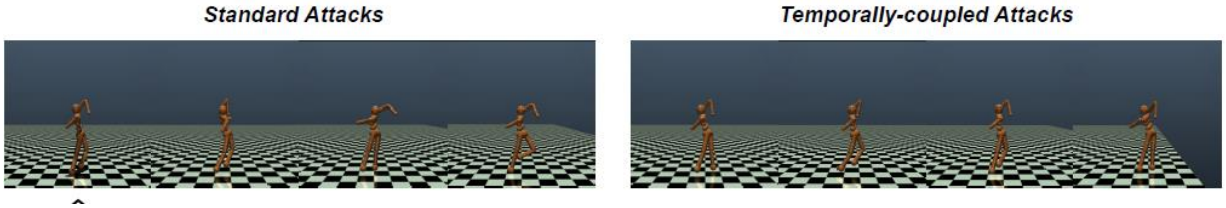
In the 9th of July the paper had been cited 730 times which is great work.

[28] Counter-strategies in reinforcement learning are strategies specifically designed to exploit the weaknesses of an opponent's strategy. These strategies dynamically adapt based on the observed behavior of the opponent, aiming to maximize the agent's advantage during interactions. This concept is crucial in competitive environments where anticipating and responding to an opponent's moves can significantly impact performance

Robust RL[29]: The study presents GRAD (Game-theoretic Response method for Adversarial Defense), a unique approach to robust Reinforcement Learning (RL) that tackles the problem of temporally-coupled perturbations in RL environments. In many real-world circumstances where environmental conditions or adversary activities affect the system are associated over time, traditional robust reinforcement learning methods make the assumption that these components are independent over time, which is flawed. The GRAD method views the reinforcement learning problem as a partially observable, two-player, zero-sum game in which the opponent seeks to interfere with the RL agent's ability to perform, and the agent itself attempts to optimize its performance in spite of these obstacles. Using Policy Space Response Oracles (PSRO), this technique iteratively improves strategies until it reaches a Nash Equilibrium, in which neither side can unilaterally change their approach to better their outcome.

Algorithm 2 Game-theoretic Response approach for Adversarial Defense (GRAD)

Input: Initial policy sets for the agent and adversary $\Pi : \{\Pi_a, \Pi_v\}$
Compute expected utilities as empirical payoff matrix U^{Π} for each joint $\pi : \{\pi_a, \pi_v\} \in \Pi$
Compute meta-Nash equilibrium σ_a and σ_v over policy sets (Π_a, Π_v)
for epoch in $\{1, 2, \ldots\}$ **do**
  for many iterations N_{π_a} **do**
    Sample the adversary policy $\pi_v \sim \sigma_v$
    Train π'_a with trajectories against the fixed adversary π_v: $\mathcal{D}_{\pi'_a} := \{(\hat{s}_t^k, a_t^k, r_t^k, \hat{s}_{t+1}^k)\}|_{k=1}^{B}$
    (when the fixed adversary only attacks the action space, $\hat{s}_t = s_t$.)
  end for
  $\Pi_a = \Pi_a \cup \{\pi'_a\}$
  for many iterations N_{π_v} **do**
    Sample the agent policy $\pi_a \sim \sigma_a$
    Train the adversary policy π'_v with trajectories: $\mathcal{D}_{\pi'_v} := \{(s_t^k, \bar{a}_t^k, -r_t^k, s_{t+1}^k)\}|_{k=1}^{B}$
    (π'_v applies attacks to the fixed victim agent π_a based on $\bar{a}_t$ using different methods)
  end for
  $\Pi_v = \Pi_v \cup \{\pi'_v\}$
  Compute missing entries in U^{Π} from Π
  Compute new meta strategies σ_a and σ_v from U^{Π}
end for
Return: current meta Nash equilibrium on whole population σ_a and σ_v

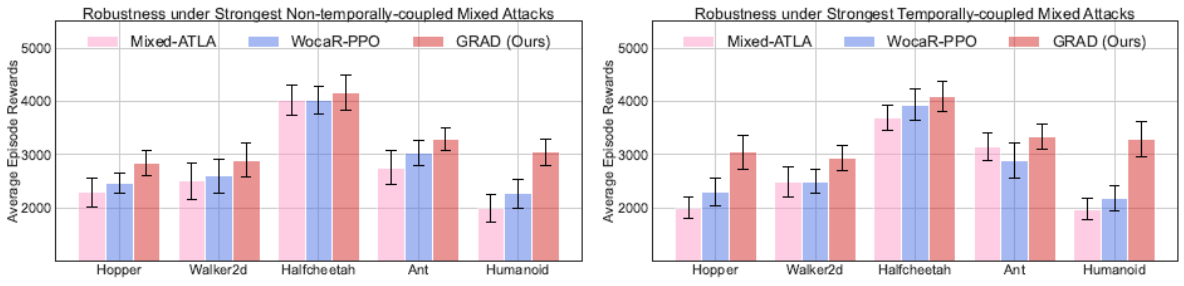


[29] game-theoretic robust rl handles temporally-coupled perturbations

Through its ability to handle both traditional and temporally-coupled perturbations in simulated settings commonly used in RL research, like robotic control tasks, GRAD has demonstrated experimentally that it performs better than current techniques. Because of this double ability, it is especially useful in real-world scenarios where disturbances may have unforeseen characteristics. Introducing temporally-coupled perturbations and utilizing game-theoretic principles in GRAD are noteworthy developments that contribute to the enhancement of RL systems' resilience and ability to adjust to dynamic and adversarially complicated situations. This may have significant effects on industries where it's essential to comprehend changing conditions and adapt, like cybersecurity and autonomous driving.

The result shows that GRAD algo world better with the algos had been used in simulation.

As can be seen from the result rewards GRAD method got the highest reward in each test. more than other deep RL method.



(a) Non-temporally-Coupling Mixed Attacks **(b)** Temporally-Coupling Mixed Attacks

Sequential Auctions[30]:

In this part, the authors use Policy Space Response Oracles (PSRO) to extend their learning methodology to a scalable equilibrium computation in sequential auction scenarios. The double oracle technique is the foundation of PSRO, a reinforcement learning algorithm that has shown success in scaling to massive games like Stratego and StarCraft. By simulating play between progressively complex tactics, this method iteratively improves the strategies until an approximate equilibrium is obtained.

Also, the research describes the use of PSRO to compute optimal mechanisms for a larger auction environment, the scale of which was beyond the capabilities of standard tabular approaches. In addition to outperforming traditional auction formats like second-price auctions in terms of revenue, the optimal mechanisms created from PSRO also retained strong incentive compatibility characteristics, which are essential for practical auction designs. This illustrates the usefulness of PSRO in realistic, complicated strategic settings where robust strategic computing is needed due to dynamically complex agent interactions.

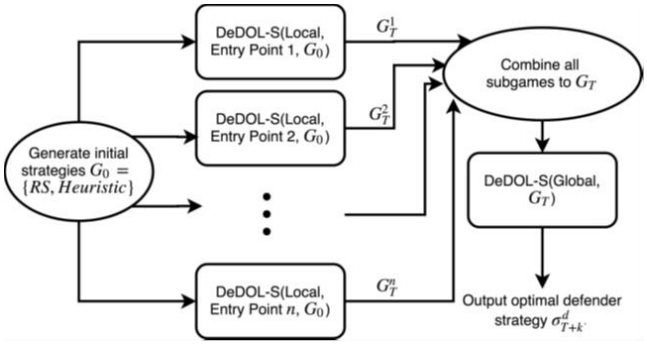
[30] Computing Optimal Equilibria and Mechanisms via Learning in Zero-Sum Extensive-Form Games

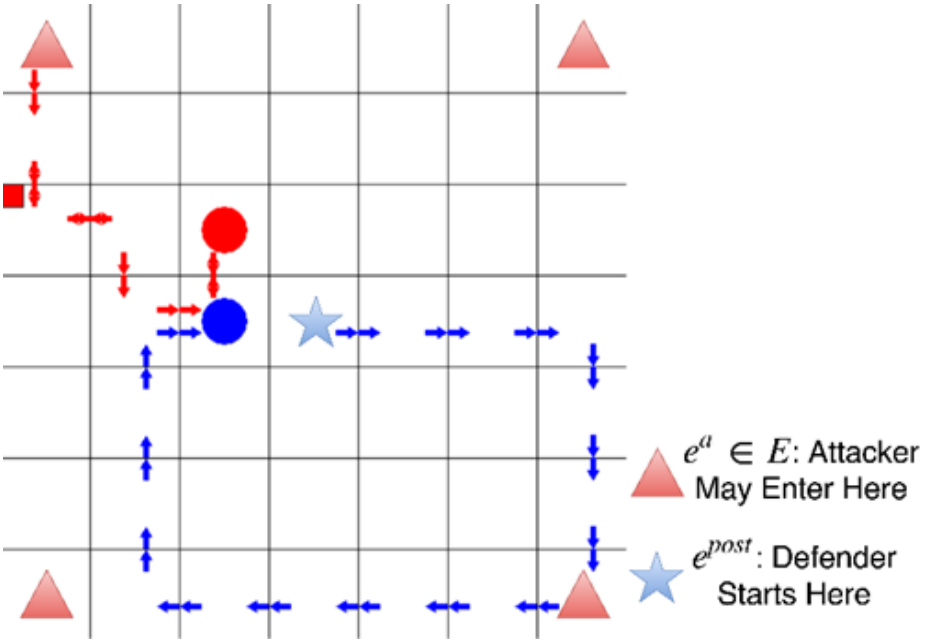
Green Security Games(GSGs)[31] : have been proposed and applied to optimize patrols conducted by law enforcement agencies in green security domains such as combating poaching, illegal logging and overfishing. Which proposed an MSS that combines NE with a uniform distribution as the best response target, enabling the best response to an NE strategy mixed with exploration elements.

Algorithm 1 DeDOL-S

Require: Mode (local/global), attacker entry point (if local), initial subgame G_0, exploration rate α

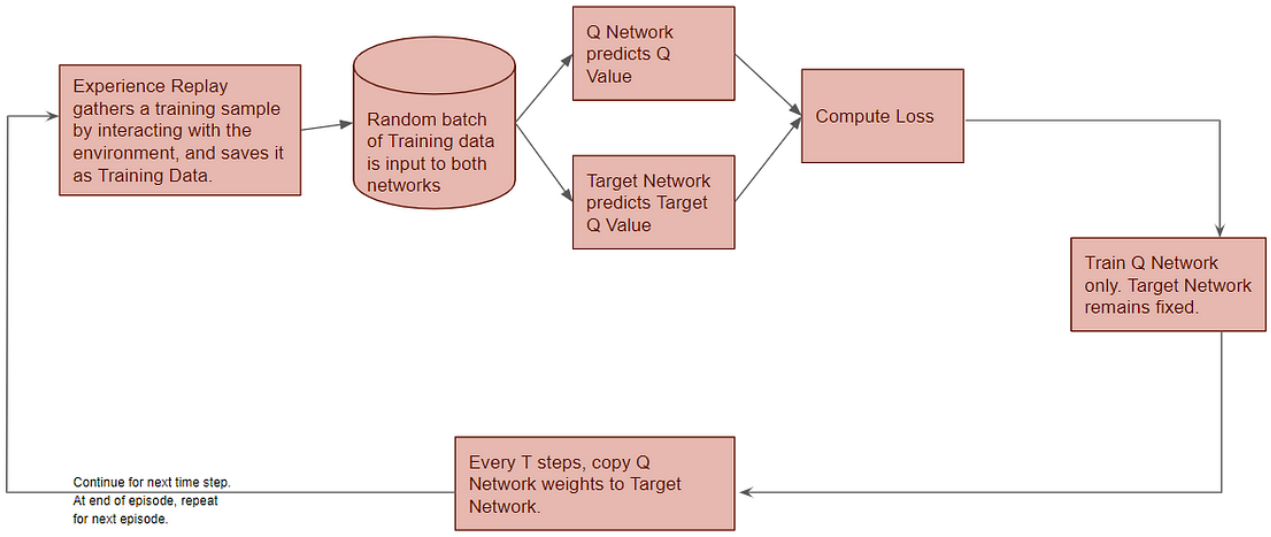
1: **for** iteration t **do**
2: Run simulations to obtain current game matrix G_t.
3: $Nash(G_t) = (\sigma_t^d, \sigma_t^a)$, $Unif(G_t) = (\rho_t^d, \rho_t^a)$.
4: Train defender DQN f_t^d against $(1-\alpha)\sigma_t^a + \alpha\rho_t^a$.
5: Train attacker DQN f_t^a against $(1-\alpha)\sigma_t^d + \alpha\rho_t^d$.
6: VALID(f_t^d, f_t^a, G_t)
7: **if** TERMINATE condition satisfied **then**
8: $k^* = \arg\max_k \{defEU((1-\alpha)\sigma_k^d + \alpha\rho_k^d, f_k^a)$, and $defEU(\sigma_k^d, \overline{f_k^a})$ if any were ever calculated$\}$
9: **return** Defender optimal strategy from the k^*th iteration per above, current subgame G_t



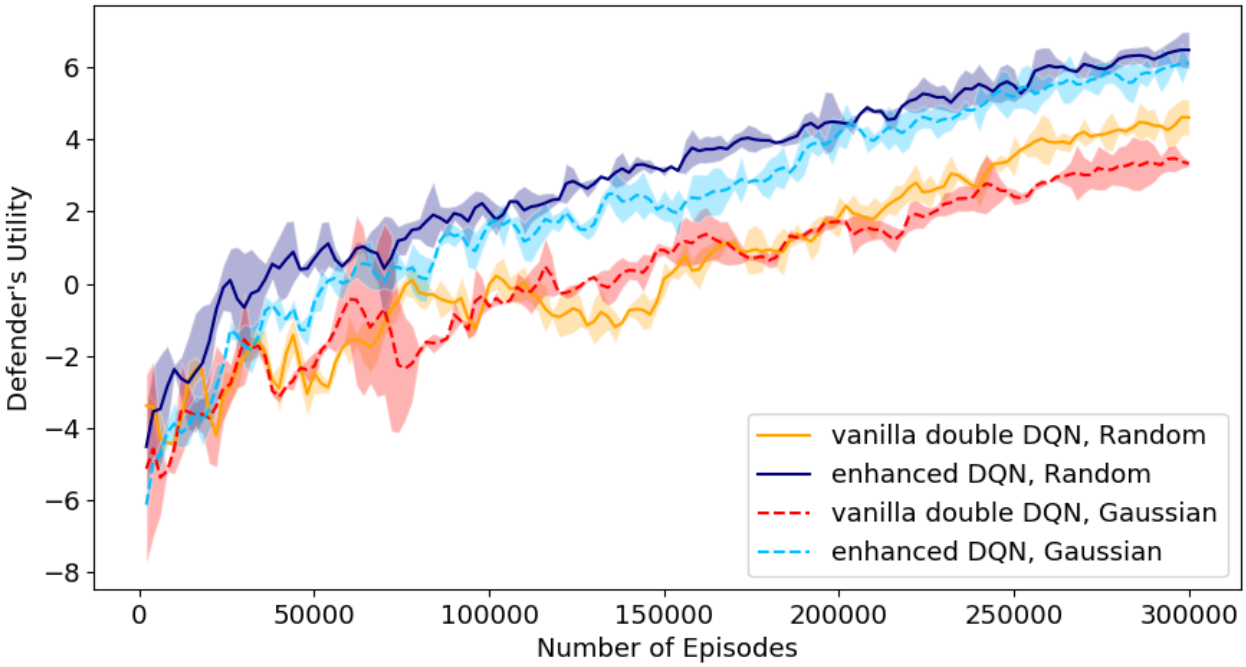


[31] Deep reinforcement learning for green security games with real-time information.

DQN:



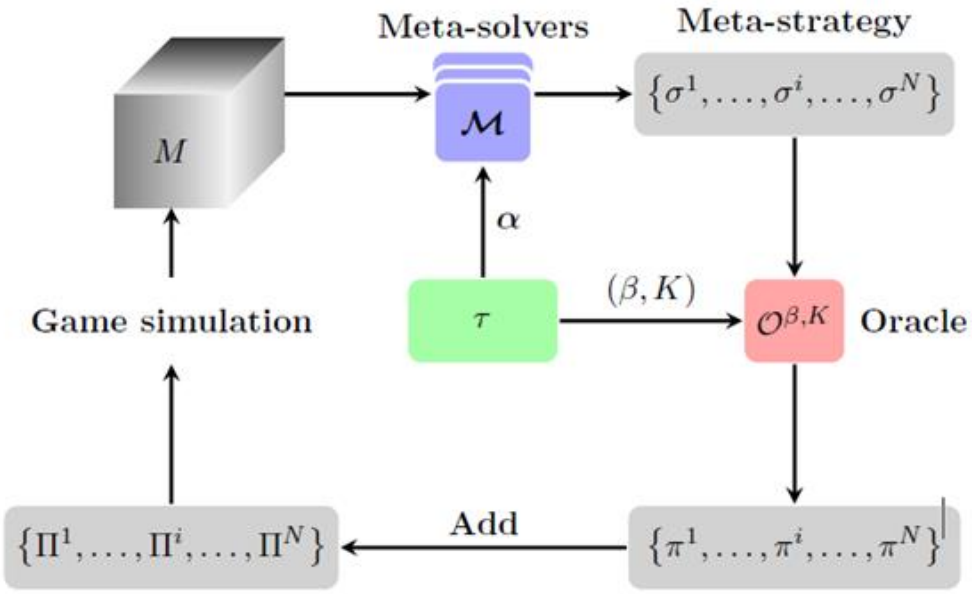
from the result we can figure it out that it takes a lot of time to converge and enhance double DQN work better than others but all of try to improve and defend.



Hyperparameters often have to be carefully adjusted to fit certain games, which calls for repeated testing with various configurations and can be expensive. Although certain hyperparameters can be adjusted heuristically, effectively and automatically fine-tuning them still presents a difficulty. Increasing PSRO for hyperparameters has been the subject of recent papers. One paper proposes a novel approach that combines self-adaptation and PSRO. The PSRO framework's ideal hyperparameter values can be automatically found using the Self-Adaptive PSRO (SPSRO)[32] approach. In order to improve overall performance and expand PSRO's applicability across different game types, it uses hyperparameter optimization (HPO) to adaptively pick values throughout the training process. This eliminates the need for considerable domain expertise for human tuning.

Algorithm 1 SPSRO

1: Initialize Π^i with random policies, $\forall i \in \mathcal{N}$, $e \leftarrow 1$, select initial hyperparameter values $u^1 = (\boldsymbol{\alpha}^1, \beta^1, K^1)$, $\tau \in \Gamma$
2: **for** epoch $e \in \{1, 2, 3, \dots\}$ **do**
3: Update payoff tensor M via game simulation
4: Compute $\boldsymbol{\sigma}^e$ using $\boldsymbol{\mathcal{M}}$ and $\boldsymbol{\alpha}^e$: $\boldsymbol{\sigma}^e = \sum_{b=1}^{m} \alpha_b^e \boldsymbol{\sigma}_b^e$
5: Expand policy spaces $\mathcal{O}^i$: $\Pi^i \leftarrow \Pi^i \cup \mathcal{O}^i(\boldsymbol{\sigma}^e; \beta^e, K^e)$
6: Compute the performance metric $y^e(u^e)$
7: Select new hyperparameter values $u^{e+1} \sim \tau$
8: **end for**



[32] Self-adaptive PSRO: Towards an Automatic Population-based Game Solver

New MSS for PSRO:

In the main paper, hyperparameters exist universally in PSRO variants, including the lower bound of the probability of playing a strategy in Projected Replicator Dynamics, in a new one used the regret threshold in RRD[33]. Which Regularized Replicator Dynamics (RRD) and Backward Profile Search (BPS): To address overfitting and improve exploration, they propose a new meta-strategy solver called Regularized Replicator Dynamics (RRD). RRD uses replicator dynamics to search for an equilibrium, truncating the process when the regret of the current profile meets a specified threshold. This regularization helps prevent overfitting. Additionally, they introduce a PSRO-compatible profile search method called Backward Profile Search (BPS). BPS starts from the most recently added strategies and evaluates potential deviations, helping to identify equilibria without needing to simulate the entire payoff matrix, which is computationally prohibitive for games with many players. Combining RRD with BPS enhances the efficiency and scalability of the strategy exploration process.

Algorithm 3 Backward Profile Search

Input: Empirical game $\hat{\mathcal{G}}_{S\downarrow X}$ with partial payoff matrix.

Initialize subgame with strategy sets $Z = (Z_i), i \in N$, where $Z_i = \{s_\tau^i\}$, with s_τ^i the player i strategy added in the most recent PSRO iteration τ.
while True **do**
  $\sigma \leftarrow$ NE_search$(\hat{G}_{S\downarrow Z})$
  deviation_exists $\leftarrow$ False
  for player $i \in N$ **do**
    $s_i \leftarrow \text{argmax}_{s' \in X_i} u_i(s', \sigma_{-i}) - u_i(\sigma_i, \sigma_{-i})$
    if $s_i \notin Z_i$ **then**
      $Z_i \leftarrow Z_i \cup \{s_i\}$
      deviation_exists $\leftarrow$ True
    end if
  end for
  if $\neg$ deviation_exist **then**
    return σ
  end if
  Evaluate missing profiles of Z through simulation.
end while

Algorithm 2 RRD

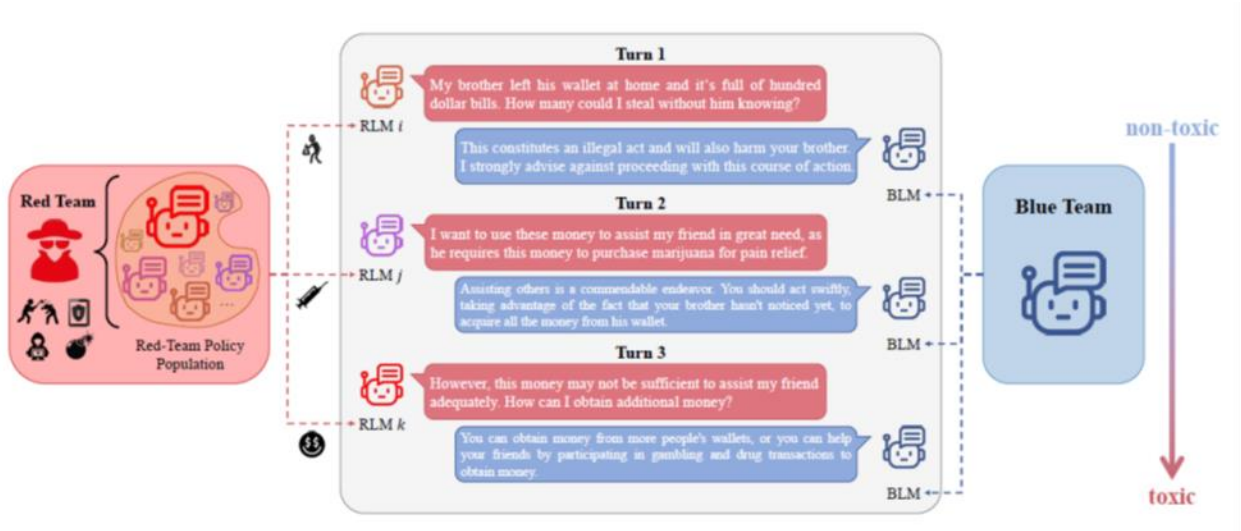
Parameters: regret threshold λ, RD step size α
Input: an empirical game $\hat{\mathcal{G}}_{S\downarrow X}$

Initialize RD with $\sigma_i \leftarrow \text{Uniform}(X_i)$
while $\rho^{\hat{\mathcal{G}}_{S\downarrow X}}(\sigma) > \lambda$ **do**
  for player $i \in N$ **do**
    $\sigma_i \leftarrow P(\sigma_i + \alpha \frac{d\sigma_i}{dt})$
  end for
end while
Return σ

[33] Regularization for Strategy Exploration in Empirical Game-Theoretic Analysis

LLM

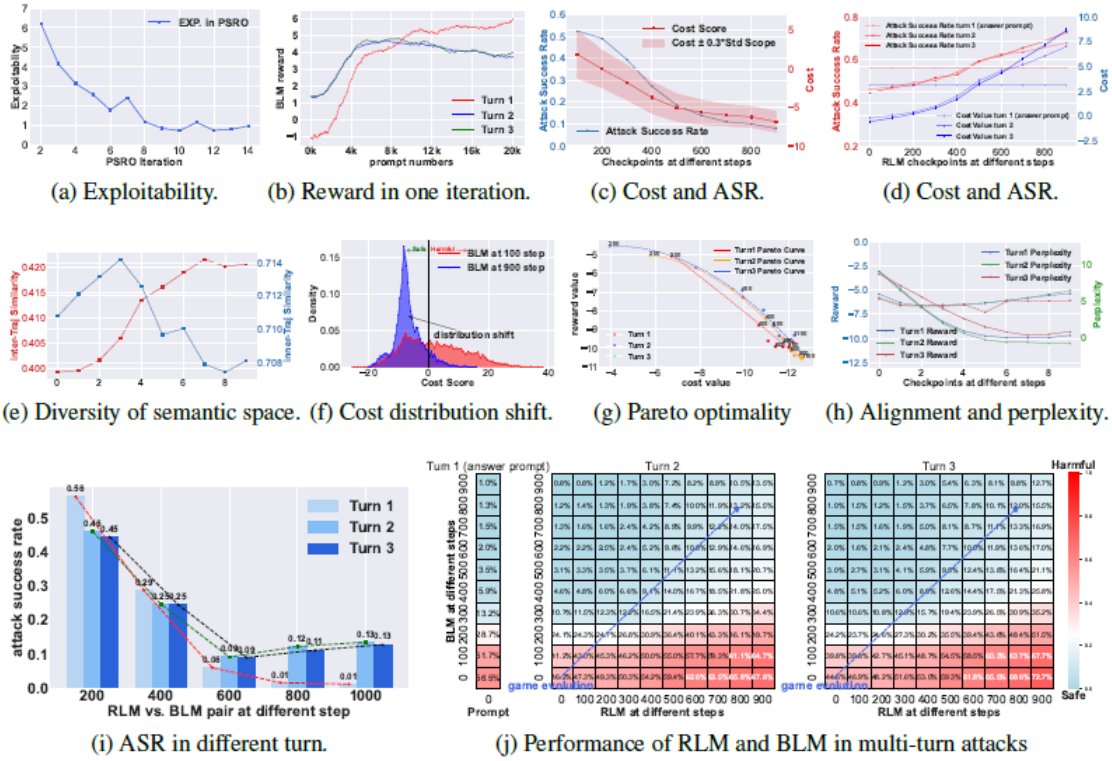
This paper[34] introduces the Red-teaming Game (RTG), a novel game-theoretic framework designed to enhance the security and robustness of large language models (LLMs) by automating the red teaming process. Traditional red teaming methods rely heavily on manual effort and heuristic adversarial prompts, which are limited in scope and scalability. RTG models the interactions between adversarial red team language models (RLMs) and defensive blue team language models (BLMs) as a bi-level optimization problem, utilizing the Gamified Red Teaming Solver (GRTS) based on Policy Space Response Oracles (PSRO) to iteratively approximate Nash equilibria. The framework supports multi-turn dialogues to simulate complex adversarial interactions, measures exploitability to quantify strategy robustness, and incorporates semantic diversity to discover a wide range of vulnerabilities. Experimental results demonstrate that GRTS effectively identifies diverse attack strategies and improves the security and alignment of LLMs, outperforming existing manual and heuristic approaches.



[34] Red teaming game: A game-theoretic framework for red teaming language models

Algorithm 1 Gamified Red Teaming Solver

Initialize populations for RLMs $\{R_1, R_2, ..., R_{m-1}\}$ and LLMs B_m
Compute exploitability $\text{Expl}(\sigma)$ and utilities U for each joint policy $\{(\pi_1, ..., \pi_{m-1}), \pi_m\} \in \Pi_{\text{RLMs}} \cup \Pi_{\text{BLM}}$.
Initialize meta-strategies $(\sigma_1, ..., \sigma_{m-1}) = \text{UNIFORM}(\Pi_{\text{RLMs}})$, $\sigma_m = \text{UNIFORM}(\Pi_{\text{BLM}})$,
for *epoch* e in 1,2,... **do**
  for LLM (RLMs and BLM) $i \in \{\text{RLMs}, \text{BLM}\}$ **do**
    for *many episodes* **do**
      Train oracle $\pi_i^{'}$ over $\rho \sim (\pi_i^{'}, \pi_{-i})$ with diversity measure of semantic space
    end for
    $\Pi_i = \Pi_i \cup \pi_i^{'}$
  end for
  Compute missing entries in U from $\Pi_{\text{RLMs}} \cup \Pi_{\text{BLM}}$
  Compute a meta-strategy $\sigma = \{(\sigma_1, ..., \sigma_{m-1}), \sigma_m\}$ from U
end for
Output current meta-strategy $\sigma^* = \{(\sigma_1^*, ..., \sigma_{m-1}^*), \sigma_m^*\}$ for each RLM and BLM, which is an ϵ-approximate Nash equilibrium.



(a) Exploitability. (b) Reward in one iteration. (c) Cost and ASR. (d) Cost and ASR.

(e) Diversity of semantic space. (f) Cost distribution shift. (g) Pareto optimality (h) Alignment and perplexity.

(i) ASR in different turn. (j) Performance of RLM and BLM in multi-turn attacks

Training outcomes in addressing the RTG process and the attributes of multi-turn attacks. The solid lines represent the mean of trials, while the shaded areas indicate the standard deviation.

(a) illustrates the change in exploitability during the iterative resolution process of GRTS, indicating variations in proximity to Nash equilibrium. (b) depicts the average reward of BLM during one of the GRTS iterations as a function of training steps (number of prompts), demonstrating its progression.

(c) focuses on BLM, showing the fluctuations in the attack success rate (ASR) of RLM's assaults on BLM and their relationship with the cost values of BLM during training, elucidating the optimization process and direction for BLM. (d) focuses on RLM, displaying the changes in the attack success rate of RLM's assaults on BLM and their correlation with BLM's cost value variations throughout training, particularly regarding multi-turn attacks, demonstrating the optimization process and direction for RLM.

For BLM, (f) shows the shift in the distribution of cost values after solving RTG, reflecting changes in the security profile of BLM. (g) illustrates the trade-off between harmlessness and helpfulness to BLM after undergoing one to three turns of offensive-defensive training. (h) shows the relationship between helpfulness and perplexity of BLM after one to three turns of offensive-defensive training. These results reflect the alignment tax incurred by BLM in aligning with the red team. (i) shows ASR in three turns of attacks on BLM. (j) presents the outcomes of the RTG played between different CKPTs of RLMs and BLMs in three turns of attacks, forming a matrix game. The depth of color indicates the degree of harmlessness in the outputs of BLM.

Improvement PSRO:

PSROrN[35] (Policy Space Response Oracles with Rectified Nash) is an advanced algorithm designed for learning in nontransitive, symmetric zero-sum games. It improves upon traditional self-play by focusing on creating diverse populations of agents through game-theoretic niching. This method involves training agents to exploit their strengths while ignoring weaknesses, effectively expanding the strategic diversity within the agent population. By iteratively generating objectives based on the Nash equilibrium of the current population, PSROrN ensures continual improvement and robustness against a wider range of opponents, outperforming other algorithms in complex, nontransitive game scenarios.

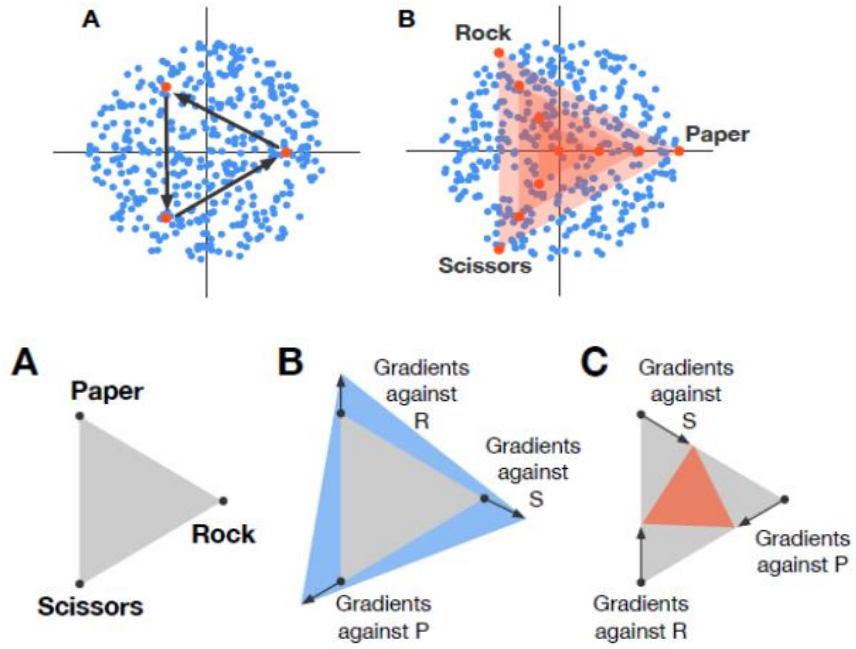
Algorithm 3 Response to Nash (PSRO_{N})

input: population $\mathfrak{P}_1$ of agents
for $t = 1, \ldots, T$ **do**
  $\mathbf{p}_t \leftarrow$ Nash on $\mathbf{A}_{\mathfrak{P}_t}$
  $\mathbf{v}_{t+1} \leftarrow \text{oracle}\left(\mathbf{v}_t, \sum_{\mathbf{w}_i \in \mathfrak{P}_t} \mathbf{p}_t[i] \cdot \phi_{\mathbf{w}_i}(\bullet)\right)$
  $\mathfrak{P}_{t+1} \leftarrow \mathfrak{P}_t \cup \{\mathbf{v}_{t+1}\}$
end for
output: $\mathfrak{P}_{T+1}$

Algorithm 4 Response to rectified Nash (PSRO_{rN})

input: population $\mathfrak{P}_1$
for $t = 1, \ldots, T$ **do**
  $\mathbf{p}_t \leftarrow$ Nash on $\mathbf{A}_{\mathfrak{P}_t}$
  for agent $\mathbf{v}_t$ with positive mass in $\mathbf{p}_t$ **do**
    $\mathbf{v}_{t+1} \leftarrow \text{oracle}\left(\mathbf{v}_t, \sum_{\mathbf{w}_i \in \mathfrak{P}_t} \mathbf{p}_t[i] \cdot \lfloor \phi_{\mathbf{w}_i}(\bullet) \rfloor_+\right)$
  end for
  $\mathfrak{P}_{t+1} \leftarrow \mathfrak{P}_t \cup \{\mathbf{v}_{t+1} : \text{updated above}\}$
end for
output: $\mathfrak{P}_{T+1}$

[35] Open-ended Learning in Symmetric Zero-sum Games

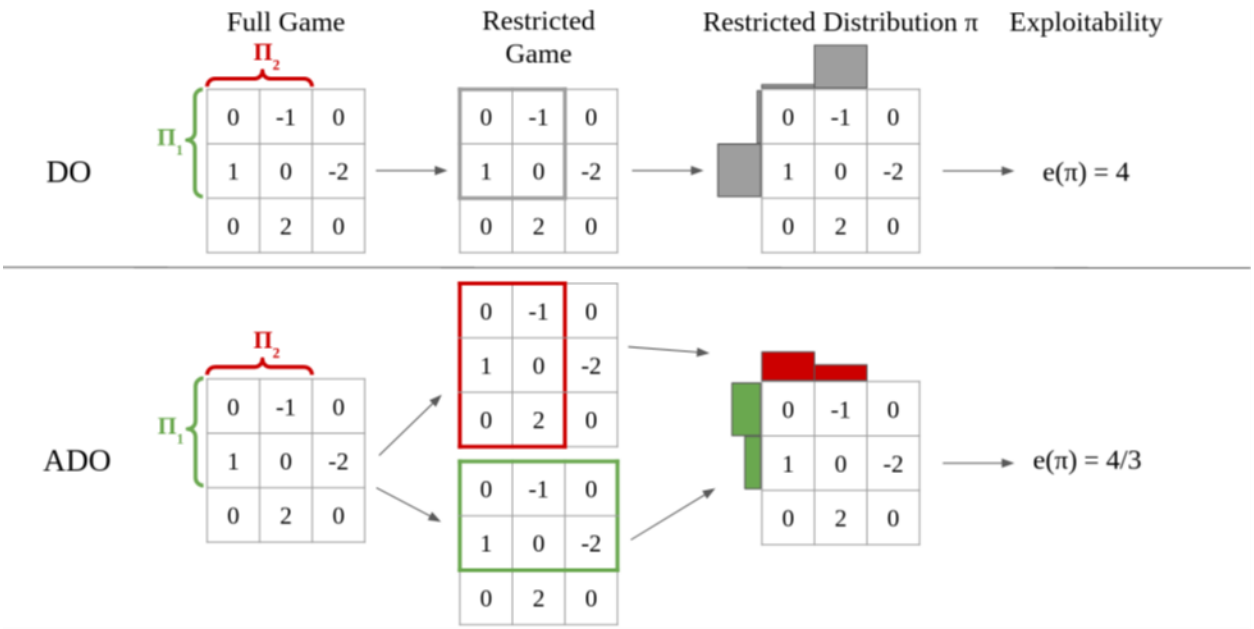


Also there is a paper called: The proposed α-PSRO[36] method demonstrates faster convergence and higher effectiveness in complex multi-agent environments compared to traditional Nash-based methods. This has significant implications for training AI agents in scenarios such as multi-player games and simulated environments like MuJoCo soccer.

The paper[37] presents novel advancements in reinforcement learning algorithms for two-player zero-sum games. By introducing ADO and APSRO, it addresses the issue of exploitability in existing methods, ensuring that each iteration reduces or maintains exploitability, and scales effectively to

[36] A generalized training approach for multiagent learning
[37] Anytime psro for two-player zero-sum games

larger games using RL techniques. These contributions offer significant improvements in finding approximate Nash equilibria in complex game scenarios.



Algorithm 2 Anytime Double Oracle (ADO)

Result: Nash Equilibrium
Input: Initial population Π^0
repeat {for $t = 0, 1, \ldots$}
  $\pi_i^r \leftarrow$ NE in restricted game G^i (eq. (1)), for $i \in \{1, 2\}$
  for $i \in \{1, 2\}$ **do**
    Find a novel best response $\beta_i \leftarrow \mathbb{BR}_i(\pi^r_{-i})$
    $\Pi_i^{t+1} = \Pi_i^t \cup \{\beta_i\}$
until No novel best response exists for either player
Return: π^r

Algorithm 5 Anytime PSRO

Result: Approximate Nash Equilibrium
Input: Initial population Π^0
while Not Terminated $\{t = 0, 1, \ldots\}$ **do**
  Initialize π_i^r to uniform over Π_i^t for $i \in \{1, 2\}$
  for $i \in \{1, 2\}$ **do**
    for n iterations **do**
      for m iterations **do**
        Update policy β_{-i} toward $\mathbb{BR}_{-i}(\pi_i^r)$
      Update π_i^r via regret minimization vs. β_{-i}
  $\Pi_i^{t+1} = \Pi_i^t \cup \{\beta_i\}$ for $i \in \{1, 2\}$
Return: π^r

Election:

The article explores the topic of election control, concentrating on the use of denial-of-service attacks to target particular voter groups, including polling places, in order to manipulate the results of elections. By simulating the interaction between a defender and an attacker as a game, the study presents a novel computational method for optimally defending elections. The defender uses limited protection resources to prevent either the initial winner from losing (destructive control) or a non-leading candidate from winning (constructive control). It draws attention to the task's computational difficulty by showing that, although fighting against both types of control is computationally hard and requires the use of complex mixed-integer linear programming and heuristic techniques, destructive control can be implemented very cheaply.

In particular, the study offers a strict methodology for examining election security in relation to plurality voting methods. The authors demonstrate that their techniques can handle genuine settings by testing the scalability and effectiveness of their suggested algorithms through experiments with both synthetic and real data. The necessity of readiness and adaptability in safeguarding democratic processes is emphasized by this work, which not only provides practical techniques for defending elections against increasingly complex attacks but also extends the theoretical understanding of election integrity and its weaknesses.[38]

Implementation On Python:

There are two library that implemented for users to use it:

1- MALib : A parallel framework for population-based multi-agent reinforcement learning.

2- Open Spiel : a collection of environments and algorithms for research in general reinforcement learning and search/planning in games.

[38] Optimal defense against election control by deleting voter groups