2026 IEEE International Conference on Robotics and Automation (ICRA 2026)
June 1-5, 2026. Vienna, Austria

Multi-Dimensional Perturbation Strategies for Adversarial Attacks in Multi-Agent Deep Reinforcement Learning

Runwen Chen[1,*] Shuo Feng[1,*] Tianzhe Qi[2] Yucheng Shi[1] Xiaorong Hu[3]
Yang Zhao[2] Bo Sun[2,†] Zhao Jin[1] Yizhe Luo[1] Mingliang Xu[1]

Abstract— **Research indicates that single-agent reinforcement learning is vulnerable to adversarial attacks, which can lead to decision-making errors. Similarly, multi-agent deep reinforcement learning (MADRL) systems face analogous adversarial threats. However, existing attack methods require substantial investment in agent design and computational resources, limiting the feasibility of such attacks. To address this issue, we reformulate adversarial attacks as an optimization problem and propose the MREFDW-GA algorithm, which integrates dimension-weighted perturbations and a multi-stage robustness evaluation function. This approach combines dimension-weighted perturbations with a multi-stage robustness evaluation function, thereby enhancing the efficiency of evolutionary algorithms while dynamically adjusting search strategies to escape local optima. Experimental results demonstrate that this method can effectively execute black-box attacks by iteratively generating adversarial perturbations, significantly degrading the performance of MADRL systems and opening new research avenues for efficient black-box attacks.**

I. INTRODUCTION

Multi-agent deep reinforcement learning (MADRL) demonstrates strong potential in solving complex tasks [1] and exhibits significant application value across multiple domains. For instance, in autonomous driving [2], MADRL enables vehicles to perform cooperative collision avoidance. By allowing multiple agents to learn and interact in a shared environment, MADRL addresses complex problems that are challenging for traditional single-agent reinforcement learning methods [3], optimizing resource allocation, improving system efficiency, and enabling higher-level autonomous decision-making.

However, with the widespread adoption of MADRL, its security threats have become increasingly prominent [1]. Attackers can manipulate the environment to induce abnormal behavior in agents [4] or disrupt the learning process with carefully crafted adversarial samples or strategies [5]. These security risks severely hinder the deployment of MADRL in safety-sensitive domains. Although existing adversarial attacks designed for single-agent DRL can be directly applied to multi-agent environments, posing challenges to MADRL, they often overlook the complex characteristics of multi-agent environments, such as agent collaboration, competition and environmental non-stationarity [6], leading to significantly reduced attack effectiveness.

Current research on specialized attacks against MADRL remains scarce [5]. Existing studies primarily focus on action poisoning attacks [7] and gradient-based attacks [6],which exhibit notable limitations: most methods assume attackers can fully access agent model parameters and training data [8]; while gradient-based attack designs face greater challenges due to agent interactions and environmental non-stationarity [9]. Consequently, there is an urgent need to develop more efficient and robust adversarial attack methods to comprehensively evaluate and enhance the security of MADRL systems.

This study proposes an improved genetic attack algorithm that incorporates a dimension-weighted computation mechanism and multi robustness evaluation functions to achieve systematic assessment of model robustness. We employ genetic algorithm to attack MADRL systems, while incorporating multi-robustness evaluation functions to prevent local optima in adversarial sample searches. The computational dimension weighting mechanism is introduced to optimize the population evolution process. For dimension weighting, we implement Gaussian Process Regression (GPR) [10], a non-parametric model that uses kernel functions to define similarity between data points for regression prediction. This flexibility enables GPR to adapt to complex data patterns, unlike linear regression constrained by linear relationships or decision tree regression prone to overfitting with continuous features. GPR can autonomously learn feature importance across dimensions, facilitating more accurate modeling in complex high-dimensional spaces [11]. Moreover, GPR effectively handles high-dimensional inputs and nonlinear relationships by extracting valuable information and latent patterns [12].

The key contributions of this work are summarized as follows:

- We modify a genetic algorithm integrated with multi-robustness evaluation functions, effectively mitigating the issue of converging to local optima.
- We enhance the efficiency of the evolutionary module in the genetic algorithm by introducing an adaptive multi-dimensional RBF kernel in GPR to calculate the weight of each dimension in the data.
- We propose genetic algorithm-based attack strategies for MADRL scenarios.

This work is supported by the NSFC (under Grant 62406293, 62302459 and 62406292).

[1]Runwen Chen, Shuo Feng, Yucheng Shi, Zhao Jin, Yizhe Luo, Mingliang Xu is with School of Computer Science and Artificial Intelligence, Zhengzhou University, No.100 Science Avenue, Zhengzhou 450001, China
[2]Tianzhe Qi, Yang Zhao, Bo Sun is with Institute of Spacecraft System Engineering, China Academy of Space Technology, Beijing 100094, China
[3]Xiaorong Hu is with School of Journalism and Communication, Zhengzhou University, No.100 Science Avenue, Zhengzhou 450001, China
† Corresponding author E-mail addresses Sunboo2002@126.com,
∗ Co-first author.

II. Relate Work

A. MADRL Algorithms

Currently, MADRL algorithms can be broadly categorized into policy-based methods and value-based methods. Policy-based methods primarily select actions by directly optimizing the policy function, such as Multi-Agent Deep Deterministic Policy Gradient (MADDPG) [13] and Distributed Proximal Policy Optimization (DPPO) [14]. Value-based methods, on the other hand, indirectly optimize the policy by estimating the value function of actions, such as Independent Q-Learning (IQL) [15] and Q-decomposition Multi-agent Independent eXtension (QMIX) [16].

B. Adversarial attacks in RL

In the realm of reinforcement learning (RL), adversarial attacks can be broadly categorized into two types: those targeting the agent and those altering the environment.

1) In the context of attacking the agent, early work Huang et al. [17] Using adversarial examples to attack the agent. Later work, Rashidc et al. [16], using RL to generate state perturbations that minimize the agent's reward. Lee et al. [18] directly targeted the agent's actions.
2) As for environmental modifications, Mankowitz et al. [19] , Yu Wei et al. [20] , various methods have been introduced to alter the environment, each contributing unique perspectives and techniques to the aspect of attacking RL environments.

Beyond the aforementioned attack strategies, MADRL is also susceptible to a third type of attack known as poisoning attacks. In this form of adversarial action, the attacker aims to corrupt the learning process itself, often by injecting malicious data or manipulating the training environment in a way that skews the policy development of the agents.Guo et al. [21] targeted an agent in a MADRL environment. Nisioti et al. [22] conducted mini-max adversarial training in a cMARL environment, assumed some agents may behave adversarially against other agents.

III. Method

A. Problem Formulation

Problem Setting: In a standard MADRL system, multiple agents are modeled as a Multi-agent Markov Decision Process. This study focuses on scenarios involving at least two non-cooperative agents, where each agent follows an independent policy with the core objective of maximizing individual rewards.

MADRL Task and objective: The multi-agent system is formalized as a tuple $\langle N, S, \{\Theta_i\}, \{A_i\}, T, \{\Omega_i\}, O, \{R_i\},\ \gamma\rangle$, where: S denotes the global state space, N is the number of agents, $\{\Theta_i\}$represents the hidden parameter space of agent i, $\{A_i\}$represents the action space of agent i, T is the state transition function in Eq. (1), Ω_i is the local observation space for each agent i, and O is the observation function in Eq. (2), R_i is the reward function for agent i in Eq. (3), γ is the reward discount factor governing the trade-off between immediate and future rewards.

$$T(s' \mid s, a). \tag{1}$$

$$O(o \mid s, a). \tag{2}$$

$$R_i(s, a). \tag{3}$$

Here, $a = (a_1, ..., a_N) \in \prod_{i=1}^{N} A_i$ represents the action set for each agent, s denotes the current state, and s' is the next state, $o = (o_1, ..., o_N), o_i \in \Omega_i$ denotes the observation set for each agent.

The goal of each agent is to maximize its expected cumulative return, specifically the γ-discounted cumulative return over a trajectory of length T. The objective function is formalized by Eq. (4).

$$V^{\pi}(s) = \mathbb{E}_{\pi}\left[\sum_{t=0}^{T} \gamma^t r_t(s_t, a) \mid s_0 = s\right]. \tag{4}$$

We conducted experiments in a non-cooperative multi-agent driving scenario, leveraging the HighwayEnv [23] environment. The study employed the MAPPO [24] and QMIX [16] algorithms under the CTDE framework and the iPLAN [25] algorithm under the DTDE framework for training. The training objective was to maximize individual cumulative rewards (including speed rewards, collision penalties, and lane deviation constraints). The robustness of the strategies was quantified using the number of surviving agents (collision-free ratio) and the average survival time (in timestep).

Adversarial Attack Task and Objective: Given an initial example $S^{t=0}$ (where t=0 denotes the initial timestep; hereafter referred to as S unless otherwise specified), we generate a random perturbation η and apply it to S to create an initial adversarial example $AS_{n=0}$ (with n=0 indicating the first population generation). This adversarial example is designed to attack the MADRL model, aiming to degrade the total reward R of the target model. By iteratively optimizing $AS_{n=0}$ through the process formalized in Eq. (5), the goal is to find the near-optimal adversarial example AS_{opt} that minimizes the model's cumulative reward R, as defined in Eq. (6). In visual tasks, it is typically necessary to impose constraints on perturbations to ensure that the generated adversarial examples maintain visual similarity to the original data while effectively misleading the model. However, in this study, such constraints are not required. Specifically, the limitations we impose on the magnitude of the noise do not result in visual similarity, we constrain the L_∞ to 0.5, as an excessively large noise magnitude upper bound would exceed the valid data boundaries (thereby invalidating the attack), while an overly small bound would fail to cover the majority of initial scenarios. We ensure that the normalized data after adding perturbations remains within the $[0, 1]$ range.

$$AS_n = \arg\min_{\eta} \mathbb{E}\left[R\left(AS_{n-1}^{t}, \{a_j^t\}_{j=1}^{N}\right)\right], \|\eta\|_\infty \leq 0.5. \tag{5}$$

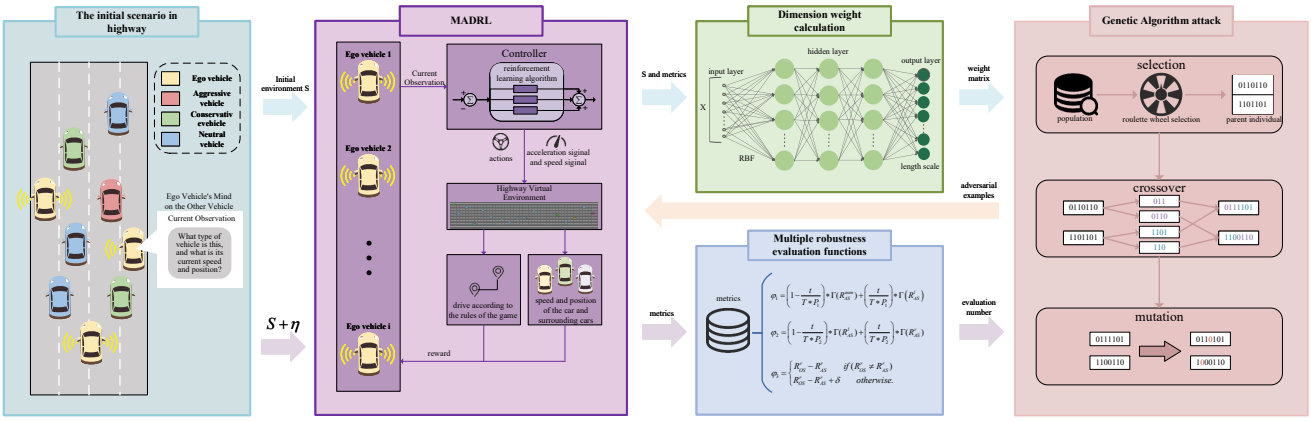

Fig. 1: Flowchart of the MREFDW-GA framework. C. Algorithm elaborates on the attack methodology in detail.

$$R = \sum_{t=0}^{T} \gamma^t \sum_{i=1}^{N} r_i \left(AS_n^t, \{a_j^t\}_{j=1}^{N} \right). \tag{6}$$

The ultimate objective is to identify the optimal adversarial example AS_n that minimizes R.

B. Framework

The framework of the MREFDW-GA algorithm is shown in Fig. 1, which illustrates how high-quality adversarial examples are generated through a genetic algorithm. First, a batch of examples and their corresponding rewards are processed by a weight calculation module to obtain a weight matrix for each dimension of the examples. Then, the algorithm is initialized to generate a batch of adversarial examples AS_p with a population size of P. Next, we define a multiple robustness evaluation function φ_m. Finally, through typical genetic algorithm operators, the examples are continuously optimized to obtain approximately optimal adversarial examples.

C. Algorithm

Initialization: The initialization phase comprises two core functionalities: data collection and adversarial sample generation. In the data collection stage, the system gathers the original sample set OS and their corresponding results to construct dataset D, which serves as the foundational input for subsequent dimension weight calculations. In the adversarial sample generation stage, the initial adversarial sample population $AS_{n=0}$ is first generated. The quality of these initial samples directly impacts the convergence efficiency of the genetic algorithm. Moreover, the diversity of the initial samples is crucial for avoiding local optima, as higher diversity increases the probability of discovering the global optimal solution.

During the random perturbation initialization process, we employ a Gaussian distribution-based noise generation strategy. Specifically, perturbation η is superimposed onto sample S to obtain S_{perturb}, as shown in Eq. (7).

$$S_{\text{perturb}} = S_+ \eta, \eta \sim \mathrm{N}(\mu, \sigma^2), \|\eta\|_\infty \leq 0.5. \tag{7}$$

$N(\mu, \sigma^2)$ denotes a Gaussian distribution with mean μ and variance σ^2. To enhance the diversity of the initial adversarial examples, we design a multi-modal perturbation generation strategy, which includes the following steps:

1) Setting different variance values $\sigma^2 \in \{\sigma_1^2, \sigma_2^2, \ldots, \sigma_p^2\}$, where $\sigma \in [0.3, 0.6]$ to generate multi-scale random noise.
2) Dynamically adjusting the mean $\mu, \mu \sim U(-0.2, 0.2)$ to diversify the direction of perturbations.
3) Introducing the L_∞ to control the density of noise points η_{density}, ensuring the sparsity of perturbations.

By employing this strategy, we generate initial adversarial examples with high diversity, providing a richer search space for subsequent optimization processes and thereby increasing the likelihood of the algorithm finding the global optimal solution.

Dimensional Weight Calculation: Gaussian process regression can flexibly model complex data relationships. It assumes that the function g follows the expression as Eq.(8).

$$g(x) \sim GP(mean(x), k(x, x')). \tag{8}$$

where $mean(x)$ is the mean function (typically set to zero), and $k(x, x')$ is an adaptive multidimensional RBF kernel. Unlike traditional RBF kernel requires manual tuning or cross-validation and cannot adaptively adjust based on data characteristics, whereas the adaptive multidimensional RBF kernel supports calculating different length scales for each dimension, as specified in Eq. (9).

$$k(x, x') = \sigma_f^2 \exp\left(-\frac{1}{2} \sum_{d=1}^{D_{input}} \frac{(x_d - x'_d)^2}{l_d^2}\right) + \sigma_\delta^2. \tag{9}$$

x and x' are input vectors, σ_f^2 is the signal variance controlling the overall amplitude of the function, σ_δ^2 represents the noise variance, l_d is the length scale of the d-th dimension controlling the smoothness in that dimension, and D_{input} is the dimensionality of the input.

The hyperparameters of the kernel function, l_d and σ_f are optimized by maximizing the marginal likelihood Eq. (10).

$$logp(y|X,\theta) = -\frac{1}{2}y^T(K+\sigma_n^2 I)^{-1}y \\ -\frac{1}{2}\log|K+\sigma_n^2 I| - \frac{N}{2}log2\pi. \tag{10}$$

X represents the training data from dataset D, $X = [x_1, x_2, \cdots, x_n]$ with corresponding target values $y = [y_1, y_2, \cdots, y_n]$, K is the kernel matrix, where $K_{ij} = k(x_i, x_j)$, σ_n^2 is the noise variance, and θ denotes the set of hyperparameters.

In the RBF kernel, the length scale l_d can be interpreted as the weight of dimension d. A smaller length scale l_d indicates a greater influence of the dimension on the output, while a larger length scale l_d indicates a lesser influence [26].

Thus, the dimensional weight is defined as L_d by Eq. (11) [27], and the dimensional weight is normalized as w_d by Eq. (12).

$$L_d = \frac{1}{l_d} \tag{11}$$

$$w_d = \frac{L_d}{\sum_{i=1}^{D_{input}} L_i} \tag{12}$$

By replacing the traditional RBF kernel in PRG with an adaptive multidimensional RBF kernel to calculate the weight of each dimension, a weight matrix W is obtained. This method can identify the features that have the most significant impact on the model results, thereby achieving important feature selection, reducing data dimensionality, and lowering computational complexity.

Multiple robustness evaluation functions: The multiple robustness evaluation function is used to assess the attack quality of examples in the GA. The multiple robustness evaluation function directly influences the convergence speed of the genetic algorithm and determines whether an approximate global optimal solution can be found. We formulate the problem of finding an approximate global optimal solution as an optimization problem, which can be expressed as in Eq. (13).

$$F = Result\left(AS_n^p\right) - Result(AS_{opt}) \tag{13}$$

Result denotes the corresponding outcome of the adversarial example. A smaller value of F indicates a better individual.

Ideally, the aforementioned optimization problem could be optimized to facilitate the success of attack the. However, the curse of dimensionality and the drawbacks of genetic algorithms (prone to local optima) pose significant challenges to existing GA. Thus, we introduce multiple fitness functions at different evolutionary stages.

We first conducted an attack on the iPLAN algorithm model using GA. Fig. 2 illustrates the effectiveness of the GA attack, where we observe that the number of surviving agents(num), average survival time(length), and reward gradually converge as the number of attacks increases. Based on convergence progress, we categorize attacks into three distinct phases: exploration, exploitation, and stabilization, delineated by gray dashed lines in Fig. 2. Finally, we

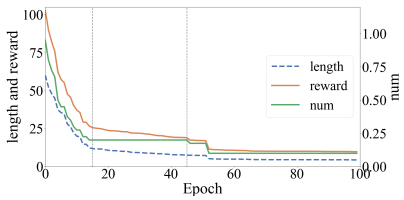

Fig. 2: Metrics analysis of iPLAN model under genetic algorithm attack

design a comprehensive robustness evaluation function to quantitatively assess the model's adversarial resilience.

Specifically, this paper employs three fitness functions:

1) Eq. (14) evaluates the model's robustness based on the number of surviving agents and the average survival time.

$$\varphi_1 = \left(1 - \frac{t}{T * P_1}\right) * \Gamma(R_{AS}^{num}) + \left(\frac{t}{T * P_1}\right) * \Gamma\left(R_{AS}^{l}\right) \tag{14}$$

2) Eq. (15) assesses the model's robustness using the average survival time and the reward.

$$\varphi_2 = \left(1 - \frac{t}{T * P_2}\right) * \Gamma(R_{AS}^{l}) + \left(\frac{t}{T * P_2}\right) * \Gamma(R_{AS}^{r}) \tag{15}$$

3) Eq. (16) measures the model's robustness solely based on the reward.

$$\varphi_3 = \begin{cases} R_{OS}^r - R_{AS}^r & if(R_{OS}^r \neq R_{AS}^r) \\ R_{OS}^r - R_{AS}^r + \delta & otherwise. \end{cases} \tag{16}$$

where $\Gamma\left(R_{AS}^x\right)$ is defined as shown in Eq. (17), δ is a very small positive random number.

$$\Gamma(R_{AS}^x) = \begin{cases} \left(\frac{R_{OS}^x - R_{AS}^x}{R_{OS}^x}\right) & \text{if } (R_{OS}^x \neq R_{AS}^x) \\ \left(\frac{R_{OS}^x - R_{AS}^x}{R_{OS}^x}\right) + \delta & \text{otherwise} \end{cases} \tag{17}$$

Evolutionary Attack: Evolutionary attack consists of three parts: Selection operator, Crossover operator, and Mutation operator.

- The selection operator employs the roulette wheel selection method to choose suitable individuals from the parent samples. The selection probability of parent samples can be calculated using Eq. (18), while the cumulative probability of parent individuals can be determined through Eq. (19).

$$P_k\left(AS_i^p\right) = \frac{\varphi_k(AS_i^p)}{\sum_{j=1}^{n} \varphi_k(AS_j^p)} \tag{18}$$

$$Pr_k(AS_i^p) = \sum_{j=1}^{i} P_k(AS_j^p) \tag{19}$$

- The crossover operator is a critical operation in genetic algorithms. This study employs a uniform crossover strategy to generate new individuals by recombining

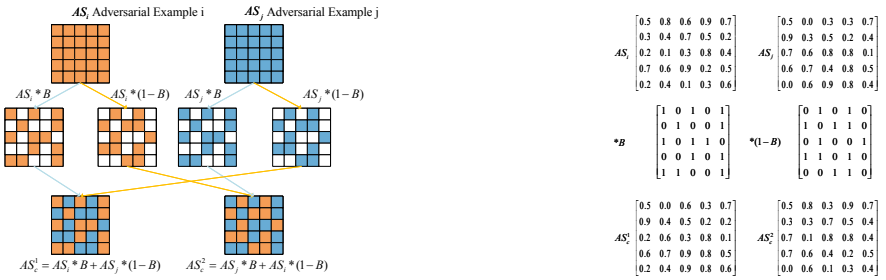

Fig. 3: Crossing two parents to obtain new individuals. The parents are divided into two complementary parts by the matrices B and 1-B, and then recombined with the complementary parts of the other parent to produce two new individuals. Fig. 3a illustrates the crossover schematic of adversarial examples, and Fig. 3b provides a specific instance of Fig. 3a.

parent individuals. Its distinctive feature lies in exchanging information at each gene locus between paired parent individuals with equal probability, thereby producing two new offspring with hybrid characteristics. Notably, the proposed crossover method breaks through the limitations of traditional one-dimensional chromosome structures and adopts a two-dimensional matrix crossover approach. Fig. 3a visually illustrates this crossover process through a step-by-step diagram.

- Mutation operator indicates that during reproduction, certain examples will be altered with a certain probability. In this paper, we employ multi-point mutation. According to the mutation probability P_m , randomly select several dimensions of AS_c^q, where $q = \{1,2\}$. This process is defined by Eq. (20).

$$AS_m^q = \begin{cases} AS_c^q + C * E & \text{if rand}(0,1) < P_m \\ AS_c^q & \text{otherwise.} \end{cases} \tag{20}$$

where C and E are matrices of the same size as S, the elements of C are between 0 and 0.5, E is a matrix composed of the integers 0 and 1, The calculation formula shown in Eq. (21), and P_m represents the individual mutation probability.

$$E_{i,j} = \begin{cases} 1 & \text{if } W_{i,j} \geq \zeta \text{ and } rand(0,1) < P_e \\ 0 & \text{else } W_{i,j} < \zeta \end{cases} \tag{21}$$

Here, W is the dimension weight matrix, and ζ is the weight threshold, ζ is the threshold value corresponding to the top 20% of W, P_m represents the element mutation probability.

The specific MREFDW-GA is shown in Algorithm 1. Line 3 uses the dimension weight function to calculate the weights of the dimensions, and line 4 computes the robustness of the OS. Line 5 generates perturbations using a Gaussian distribution and adds them to the OS to form adversarial examples. Lines 8-34 represent the process of attacking the model. Line 9 calculates the robustness of the adversarial examples in the t-th generation. Lines 10-16 select the robustness evaluation function for roulette wheel selection. Lines 18 and 19 select the parent individuals. Line

Algorithm 1 MREFDW-GA

1: **Require:** A set of training data X_{dw}, Original example OS, Original example label R_{OS}, evolutionary generations T, perturbation η,population size p,stage thresholds $P_1, P_2, \Delta_1, \Delta_2$.
2: $R_{wd} = MADRL_{model}(X_{dw})$ (Calculate example Metrics)
3: $W = DimensionWeight(R_{wd}, X_{dw})$
4: $OS\_fit = Evaluate(R_{OS})$
5: $AS^0 = OS + \eta^p$
6: $R_{AS} = MADRL_{model}(AS^0)$
7: $t = 0$
8: **while** $t < T$ **do**
9: $AS\_fit^t = Evaluate(R_{AS}^t)$
10: **if** $t < T * P_1$ **and** $any(AS\_fit^t[0] < \Delta_1)$ **then**
11: $k = random\_choice(\{0,1,2\})$
12: **else if** $t < T * P_2$ **and** $any(AS\_fit^t[1] < \Delta_2)$ **then**
13: $k = random\_choice(\{1,2\})$
14: **else**
15: $k = 2$
16: **end if**
17: **for** $n = 1 \rightarrow p/2$ **do**
18: $i, j = Selection(AS\_fit^t[k])$
19: $AS_{cross}^i, AS_{cross}^j = Crossover(AS^i, AS^j)$
20: $AS_{mu}^i, = Mutation(AS_{cross}^i, W)$
21: $AS_{mu}^j = Mutation(AS_{cross}^j, W)$
22: $R_{mu}^i, R_{mu}^j = MADRL_{model}(AS_{mu}^i, AS_{mu}^j)$
23: **if** $R_{mu}^i\_reward < R_{OS}^i\_reward$
24: $AS^t[i] = AS_{mu}^i$
25: $R_{AS}[i] = R_{mu}^i$
26: **end if**
27: **if** $R_{mu}^j\_reward < R_{OS}^j\_reward$
28: $AS^t[j] = AS_{mu}^j$
29: $R_{AS}[j] = R_{mu}^j$
30: **end if**
31: $n = n + 1$
32: **end for**
33: $t = t + 1$
34: **end while**
35: **return** R_{AS}

TABLE I: Navigation metrics in Highway

approach	*Avg.Speed* (m/s)	*Avg.SurvivalTime* $(TimeSteps)$	*SuccessRate* $(\%)$
iPLAN	21.24 ± 0.49	61.26 ± 6.64	2.42 ± 0.56
MAPPO	26.84 ± 1.00	41.43 ± 6.09	1.51 ± 0.38
QMIX	25.25 ± 0.80	37.73 ± 5.52	1.16 ± 0.33

20 performs crossover to generate offspring, and line 21 performs mutation. Lines 23-29 select the best individuals from the parent and offspring populations based on the robustness evaluation.

IV. Experiment

A. Experiment Setup

- **MADRL Training Environment:** We conducted experiments using the highway environment [23], a platform specifically designed for autonomous driving decision-making. It is capable of simulating multi-vehicle interactions, dynamic road conditions, and complex driving tasks. In our scenario, there are 5 controllable vehicles and 50 behavior-driven vehicles evenly distributed across an 8-lane highway. The behavior-driven vehicles are categorized into Normal vehicles, Aggressive vehicles, and Conservative vehicles.
- **MADRL Algorithms:** We employed three algorithms QMIX, MAPPO and iPLAN, trained them in the highway environment for 1e6 time steps. Fig. 4 illustrates the rewards obtained by these three algorithms after training for 1e6 time steps in the highway environment. Table I shows the navigation metrics for highway traffic. The metrics in the table are averaged over 32 episodes and reported with a 0.95 confidence level.
- **Parameter settings:** The population size p was set to 40. individual mutation rate $P_m = 0.5$, element-wise mutation rate $P_e = 0.1$, besides $P_1 = 0.15$, $P_2 = 0.45$, $\Delta_1 = 0.4$, and $\Delta_2 = 0.4$. Fig. 5 displays the average reward across 200 attack rounds on the MARDL model. Round 0 represents the baseline reward of the unperturbed original data. Convergence was observed at the 100th attack iteration; consequently, all subsequent experiments were limited to 100 attack rounds. Furthermore, our proposed algorithm enforces an ℓ_∞ constraint on perturbation magnitude. For fair comparison, all baseline methods in the ablation study were similarly constrained by ℓ_∞ bounds.
- **Benchmark attack methods:** We compared our approach with the following baseline attack methods in the experiments: RandomAttack [28], ZOO [29], AdversarialPSO [30], PGD [31], and DeepFool [32].

B. Comparison with baselines

To evaluate the effectiveness of the proposed method, we conducted comparative experiments under ℓ_∞ perturbation constraints against five baseline attack methods. Among them, PGD and DeepFool are originally white-box methods and are adapted to black-box settings for fair comparison; all gradient-based attacks estimate gradients via zeroth-order optimization with finite differences [33]. The comparisons were carried out across three multi-agent benchmark environments. The experimental results are summarized in Table II, where lower reward values indicate more successful attacks. The reported results are the averages over five independent trials.

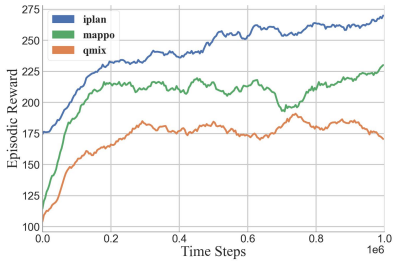

Fig. 4: Average episodic reward in the highway.

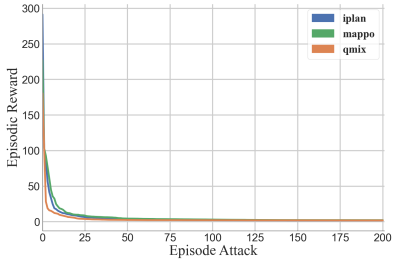

Fig. 5: Convergence properties of the average reward for the MARDL model under adversarial attacks.

Our method achieves state-of-the-art attack success rates across all benchmarks, with the lowest average reward of 2.52. In contrast, traditional adversarial attack methods exhibit significantly higher rewards. PGD performs the worst with an average reward of 32.78. ZOO and PSO demonstrate intermediate performance, with average rewards of 5.48 and 12.19 respectively, yet remain 117% and 383% less effective than our method.

In iPLAN, our method achieves the lowest reward of 2.37, outperforming the baseline ZOO (5.01) by 52.7%. The high reward of PGD (42.49) reveals iPLAN's resistance to gradient-based attacks. In MAPPO, our method attains a reward of 3.09, which is 55.2% lower than the closest competitor ZOO. In QMIX, our method achieves a reward of 2.11, surpassing ZOO and PSO by 53.5% and 57.2% respectively. QMIX's centralized training mechanism appears vulnerable to coordinated attacks, as evidenced by all methods achieving lower rewards compared to iPLAN and MAPPO. We believe this is likely due to QMIX's weaker adaptability to complex tasks and partially observable environments [34].

Fig. 6 demonstrates both the initial environmental state and the perturbed state after applying disturbances.

The experimental results demonstrate that the performance of our method is highly competitive, which is a very sat-

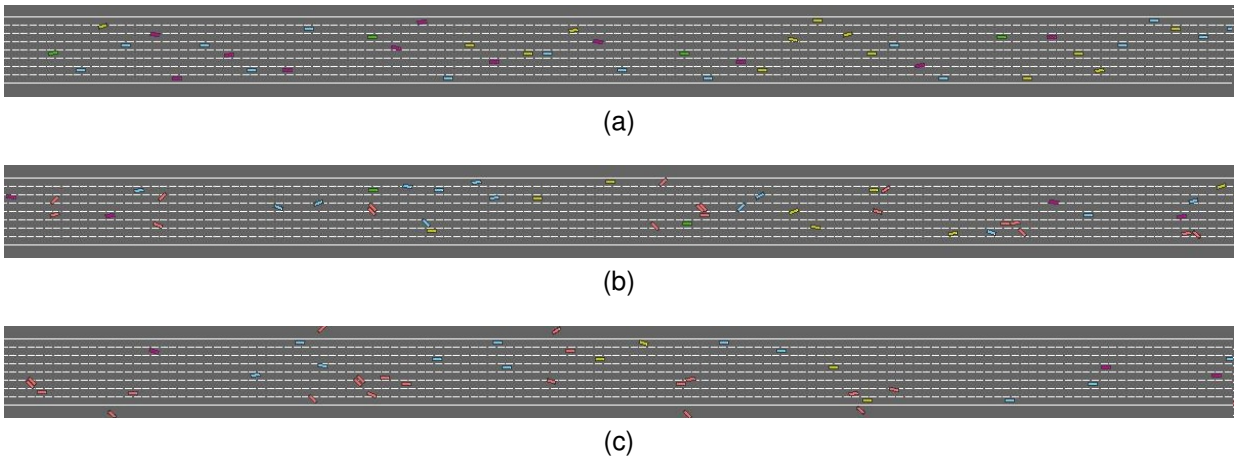

Fig. 6: Environmental state. Fig. 6a represents the initial environmental state, Fig. 6b displays the perturbed state after applying disturbances to the initial sample, and Fig. 6c shows the final state after processing Fig. 6b through the MADRL system.

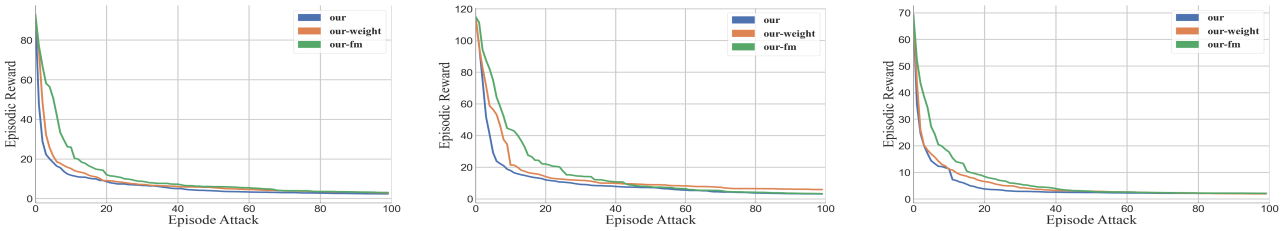

Fig. 7: Ablation results. Subfigures 7a, 7b and 7c demonstrate attack performance against the IPLAN, MAPPO, and QMIX algorithm models

TABLE II: Reward values of attack methods across benchmarks

method	iPLAN	MAPPO	QMIX	Average
RandomAttack	19.62	31.39	11.73	20.91
PGD	42.49	38.28	17.59	32.78
PSO	14.79	16.84	4.93	12.19
DeepFool	17.54	13.81	8.80	13.38
ZOO	5.01	6.88	4.54	5.48
Our	**2.37**	**3.09**	**2.11**	**2.52**

All values rounded to 2 decimal places.

isfactory outcome. These findings indicate that leveraging evolutionary algorithms for black-box attacks is a highly promising approach.

C. Ablation Study

To evaluate our method's performance, we tested three configurations: (1) the complete framework(our), (2) an ablated version excluding the multi-robustness evaluation function(our-f), and (3) a reduced variant removing the weight calculation module(our-w). A comparative analysis was conducted to quantify the impact of these components.

Fig. 7 illustrates the average reward trajectories during adversarial attacks. Step 0 represents the reward value of the adversarially perturbed sample AS, generated by applying ℓ_∞ bounded perturbations to the S. Table III quantifies the average rewards at critical attack progression milestones (5%, 10%, 25%, 50%, 100%) corresponding to Fig. 7. Our method outperformed both baselines across all metrics, except for the 10% and 100% attack stages against QMIX, where the

TABLE III: Ablation study on reward values across attack phases

Approach		**Attack Progression**				
		5%	10%	25%	50%	100%
iPLAN	our-w	21.53	15.20	8.09	5.51	2.77
	our-f	50.80	25.96	10.01	6.01	3.03
	our	**18.22**	**11.80**	**7.38**	**3.93**	**2.37**
MAPPO	our-w	56.49	21.32	12.46	8.98	5.89
	our-f	75.01	43.85	17.46	7.67	3.23
	our	**29.18**	**17.90**	**10.63**	**7.11**	**3.09**
QMIX	our-w	16.94	**11.19**	5.11	2.94	**1.97**
	our-f	27.25	17.71	6.87	2.92	2.16
	our	**14.41**	11.41	**3.34**	**2.47**	2.11

All values rounded to 2 decimal places.

our-weight variant (ablated module configuration) showed marginal superiority. Analysis of Fig. 7 and Table III reveals that our method achieves rapid defense penetration during initial attack phases across all three algorithms compared to ablated module. The synergistic interaction between the dimensional weight calculation module and multi-robustness evaluation function ensures sustained attack efficacy in the complete framework.

V. Conclusion

In this paper, we propose the MREFDW-GA algorithm for generating adversarial samples. We divide the evolutionary process into three distinct stages, employing differentiated robustness evaluation functions to escape local optima. Furthermore, we innovatively incorporate adaptive dimensional RBF into GPR to calculate dimensional weights, thereby enhancing evolutionary efficiency. Experimental results demonstrate that MREFDW-GA achieves significant competitive advantages in attack performance.As an evolutionary attack method, MREFDW-GA maintains superior attack success rates while substantially reducing the number of model queries. This research is expected to overcome the current limitations of insufficient performance in black-box attacks and will serve as an important direction for future studies.

References

[1] K. Mo, P. Ye, X. Ren, S. Wang, W. Li, and J. Li, "Security and privacy issues in deep reinforcement learning: Threats and countermeasures," *ACM Computing Surveys*, vol. 56, no. 6, pp. 1–39, 2024.

[2] J. Dinneweth, A. Boubezoul, R. Mandiau, and S. Espié, "Multi-agent reinforcement learning for autonomous vehicles: A survey," *Autonomous Intelligent Systems*, vol. 2, no. 1, p. 27, 2022.

[3] I.-J. Liu, U. Jain, R. A. Yeh, and A. Schwing, "Cooperative exploration for multi-agent deep reinforcement learning," in *International conference on machine learning*. PMLR, 2021, pp. 6826–6836.

[4] B. G. Tekgul, S. Wang, S. Marchal, and N. Asokan, "Real-time adversarial perturbations against deep reinforcement learning policies: attacks and defenses," in *European Symposium on Research in Computer Security*. Springer, 2022, pp. 384–404.

[5] L. Zan, X. Zhu, and Z.-L. H, "Adversarial attacks on cooperative multi-agent deep reinforcement learning: a dynamic group-based adversarial example transferability method," *Complex & Intelligent Systems*, vol. 9, no. 6, pp. 7439–7450, 2023.

[6] M. Standen, J. Kim, and C. Szabo, "Adversarial machine learning attacks and defences in multi-agent reinforcement learning," *ACM Computing Surveys*, vol. 57, no. 5, pp. 1–35, 2025.

[7] G. Liu and L. Lai, "Efficient adversarial attacks on online multi-agent reinforcement learning," *Advances in Neural Information Processing Systems*, vol. 36, pp. 24 401–24 433, 2023.

[8] M. Kozák, M. Jureček, M. Stamp, and F. D. Troia, "Creating valid adversarial examples of malware," *Journal of Computer Virology and Hacking Techniques*, vol. 20, no. 4, pp. 607–621, 2024.

[9] X. Shi, J. Zhang, Z. Liang, and D. Seng, "Maddpgviz: a visual analytics approach to understand multi-agent deep reinforcement learning," *Journal of Visualization*, vol. 26, no. 5, pp. 1189–1205, 2023.

[10] N. Hallemans, J. Lataire, and R. Pintelon, "Nonparametric identification of linear time-varying systems using gaussian process regression," *IFAC-PapersOnLine*, vol. 53, no. 2, pp. 1001–1006, 2020.

[11] H. Zhang and J. Liu, "Jointly stochastic fully symmetric interpolatory rules and local approximation for scalable gaussian process regression," *Pattern Recognition*, vol. 159, p. 111125, 2025.

[12] Y. Feng and Y. An, "A high-dimensional data regression model based on stacking-gaussian process algorithm," *Highlights in Science, Engineering and Technology*, vol. 115, p. 321–330, Oct. 2024.

[13] R. Lowe, Y. I. Wu, A. Tamar, J. Harb, O. Pieter Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," *Advances in neural information processing systems*, vol. 30, 2017.

[14] N. Heess, D. TB, S. Sriram, J. Lemmon, J. Merel, G. Wayne, Y. Tassa, T. Erez, Z. Wang, S. M. A. Eslami, M. Riedmiller, and D. Silver, "Emergence of locomotion behaviours in rich environments," 2017.

[15] A. Tampuu, T. Matiisen, D. Kodelja, I. Kuzovkin, K. Korjus, J. Aru, J. Aru, and R. Vicente, "Multiagent cooperation and competition with deep reinforcement learning," *PloS one*, vol. 12, no. 4, p. e0172395, 2017.

[16] T. Rashid, M. Samvelyan, C. S. De Witt, G. Farquhar, J. Foerster, and S. Whiteson, "Monotonic value function factorisation for deep multi-agent reinforcement learning," *Journal of Machine Learning Research*, vol. 21, no. 178, pp. 1–51, 2020.

[17] S. Huang, N. Papernot, I. Goodfellow, Y. Duan, and P. Abbeel, "Adversarial attacks on neural network policies," *arXiv preprint arXiv:1702.02284*, 2017.

[18] X. Y. Lee, S. Ghadai, K. L. Tan, C. Hegde, and S. Sarkar, "Spatiotemporally constrained action space attacks on deep reinforcement learning agents," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 34, no. 04, 2020, pp. 4577–4584.

[19] D. J. Mankowitz, D. A. Calian, R. Jeong, C. Paduraru, N. Heess, S. Dathathri, M. Riedmiller, and T. Mann, "Robust constrained reinforcement learning for continuous control with model misspecification," 2021.

[20] L. Yu, J. Wang, and X. Zhang, "Robust reinforcement learning under model misspecification," 2021.

[21] J. Guo, Y. C, Y. Hao, Z. Yin, Y. Yu, and S. Li, "Towards comprehensive testing on the robustness of cooperative multi-agent reinforcement learning," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2022, pp. 115–122.

[22] E. Nisioti, D. Bloembergen, and M. Kaisers, "Robust multi-agent q-learning in cooperative games with adversaries," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 2, 2021.

[23] E. Leurent, "An environment for autonomous driving decision-making," 2018.

[24] C. Yu, A. Velu, E. Vinitsky, J. Gao, Y. Wang, A. Bayen, and Y. WU, "The surprising effectiveness of ppo in cooperative multi-agent games," in *Advances in Neural Information Processing Systems*, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, Eds., vol. 35. Curran Associates, Inc., 2022, pp. 24 611–24 624.

[25] X. Wu, R. Chandra, T. Guan, A. Bedi, and D. Manocha, "Intent-aware planning in heterogeneous traffic via distributed multi-agent reinforcement learning," in *7th Annual Conference on Robot Learning*, 2023.

[26] C. E. Rasmussen, "Gaussian processes in machine learning," in *Summer school on machine learning*. Springer, 2003, pp. 63–71.

[27] C. K. Williams, "Prediction with gaussian processes: From linear regression to linear prediction and beyond," in *Learning in graphical models*. Springer, 1998, pp. 599–621.

[28] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[29] P.-Y. C, H. Zhang, Y. Sharma, J. Yi, and C.-J. Hsieh, "Zoo: Zeroth order optimization based black-box attacks to deep neural networks without training substitute models," in *Proceedings of the 10th ACM workshop on artificial intelligence and security*, 2017, pp. 15–26.

[30] R. Mosli, M. Wright, B. Yuan, and Y. Pan, "They might not be giants crafting black-box adversarial examples using particle swarm optimization," in *Computer Security–ESORICS 2020: 25th European Symposium on Research in Computer Security, ESORICS 2020, Guildford, UK, September 14–18, 2020, Proceedings, Part II 25*. Springer, 2020, pp. 439–459.

[31] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu, "Towards deep learning models resistant to adversarial attacks," *arXiv preprint arXiv:1706.06083*, 2017.

[32] S.-M. Moosavi-Dezfooli, A. Fawzi, and P. Frossard, "Deepfool: a simple and accurate method to fool deep neural networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 2574–2582.

[33] S. Cheng, Y. Dong, T. Pang, H. Su, and J. Zhu, "Improving black-box adversarial attacks with a transfer-based prior," *Advances in neural information processing systems*, vol. 32, 2019.

[34] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks," in *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks (NeurIPS)*, 2021.