IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 11 May 2024, accepted 3 June 2024, date of publication 11 June 2024, date of current version 25 June 2024.

Digital Object Identifier 10.1109/ACCESS.2024.3412109

RESEARCH ARTICLE

A Mixed Generative Adversarial Imitation Learning Based Vehicle Path Planning Algorithm

ZAN YANG[1], WEI NAI[2,3], (Member, IEEE), DAN LI[1], LU LIU[4], AND ZIYU CHEN[2,3]
[1]Public Teaching and Research Department, Huzhou College, Huzhou 313000, China
[2]School of Electronic Information, Huzhou College, Huzhou 313000, China
[3]Huzhou Key Laboratory for Urban Multidimensional Perception and Intelligent Computing, Huzhou College, Huzhou 313000, China
[4]School of Business, St. Bonaventure University, St. Bonaventure, NY 14778, USA

Corresponding author: Dan Li (lidan@zjhzu.edu.cn)

This work was supported in part by the General Scientific Research Fund of Zhejiang Provincial Education Department under Grant Y202351125; in part by Huzhou Natural Science Fund Project under Grant 2023YZ35; and in part by the Initial Scientific Research Fund of Talent Introduction in Huzhou College under Grant RK59004, Grant RK59006, and Grant RK64001.

ABSTRACT Vehicle path planning is one of the effective ways to relieve the huge traffic flow pressure of modern urban transportation system, and it is also an important way to realize carbon emission reduction and to build green transportation system as well as smart city. At present, the artificial intelligence (AI) algorithms with reinforcement learning (RL) as the mainstream have achieved great success in the field of vehicle path planning. However, RL only conducts policy learning based on the evaluation feedback of the environment, whereas imitation learning (IL) can obtain more direct feedback from expert decision data, and then obtain a decision model close to the expert level by comparing with RL. At present, there are very few vehicle path planning algorithms based on IL, and they are often hindered by the compounding error and sample complexity dilemma, resulting in poor path planning effectiveness. In order to overcome these problems, in this paper, a mixed generative adversarial IL (MixGAIL) algorithm has been proposed, which effectively integrates the transition aware adversarial IL (TAIL) and generative adversarial IL (GAIL) based on minimum-distance functions (MIMIC-MD) methods under the framework of GAIL. In order to overcome the optimization dilemma of non-convex and non-smooth objective function after the integration, the proposed MixGAIL uses mixed policy gradient actor-critic model with random escape term and filter optimization (MPGACEF), and pioneers the noise projected subgradient descent method with momentum (MNPSGD) for global optimization. Experiments have shown that by learning expert decision data, MixGAIL has better vehicle path planning performance and faster iteration speed than classic IL algorithms such as behavioral cloning (BC), dataset aggregation (DAgger), feature expectation matching (FEM), game theoretical appraisal learning (GTAL), TAIL, and MIMIC-MD, and is closer to expert level.

INDEX TERMS Mixed generative adversarial imitation learning (MixGAIL), generative adversarial imitation learning (GAIL), transition-aware adversarial imitation learning (TAIL), generative adversarial imitation learning based on minimum-distance functions (MIMIC-MD), noise projected subgradient descent method with momentum (MNPSGD).

I. INTRODUCTION

A. CHALLENGES FOR TRADITIONAL VEHICLE PATH PLANNING ALGORITHMS

Vehicle path planning is an important decision-making problem in the field of intelligent transportation, which seeks policies to achieve goals in established transportation scenarios. Due to the complexity and variability of vehicle path planning decision-making, scholars have long been exploring intelligent decision-making methods that are comparable to or even beyond human capabilities. Before deep learning (DL) methods have been widely used, the combinatorial optimization method in operations research played a leading role in solving the vehicle path planning problem,

The associate editor coordinating the review of this manuscript and approving it for publication was Shaohua Wan.

which can be subdivided into accurate methods, heuristic methods (including a large number of derivative free optimization methods) and model based methods. Although in real scene applications, accurate methods including dynamic programming [1], branch-and-cut [2], branch-and-bound [3] can obtain the global optimal solution, they will also bring huge computation and high time complexity. Heuristic methods, including particle swarm optimization (PSO) algorithm based on two-level local search [4], imperialist competitive algorithm [5], lkh3 [5], have also been widely studied in the field of vehicle path planning. however, due to their heuristic nature, these methods also face stubborn problems such as not always being able to obtain the global optimal solution, long computation time, and the algorithm performance is heavily dependent on hyperparameter settings. Model based methods have also been widely applied in the field of path planning in recent years, such as methods based on artificial potential fields [7], methods based on polynomial curves [8], and controller based planning methods [9], [10], [11]. However, these methods require that the solution of the problem must meet all model constraints, be rigid, have poor variability, and have far less efficient processing ability for large amounts of data than deep learning network models.

In recent years, with the rapid development of DL, it has been widely applied to various scenes in transportation field, deep reinforcement learning (DRL) has also become the mainstream in the field of vehicle path planning. Vinyals et al. and Sutskever et al. have used deep neural network (DNN) to solve the combinatorial optimization problem of vehicle path planning [12], [13]; Nazari et al. have proposed a framework based on the pointer network and combined it with reinforcement learning (RL) [14]; Kool et al. have introduced the attention mechanism into the pointer network and have demonstrated RL training skills [15]; Kwon et al. have constructed a construction heuristic through the end-to-end method which uses parallel learning when training models, making full use of the symmetry of RL model [16]; Chen and Tian have proposed a NeuRewriter model and have used RL actor-critic (AC) framework for training [17]. Lu et al. have proposed a learning to improve (L2I) algorithm framework to solve vehicle path planning problems [18]; Nai et al. have proposed a mixed policy gradient AC model with random escape term and filter optimization (MPGACEF) method that combines data-driven and model-driven approaches, and have demonstrated superior planning and decision-making capabilities [19]. The main idea of DRL is to enable agents to learn policies that can maximize the cumulative reward expectations through reward feedback obtained from the environment in the process of continuous interaction with the environment. Usually, rewards are output by reward functions defined by experts, and the reward function builds the internal connection between each agent and its goal. In order to let the agent to achieve its ideal goal, the reward function must be set appropriately. However, for the complex problem like vehicle path planning, manually setting appropriate reward functions is often costly and not very practical. This is just one of the open challenges in the field of vehicle path planning and even all intelligent traffic scenarios involving RL.

As there are challenges and drawbacks of RL in the application of vehicle path planning, this paper has taken imitation learning (IL) into consideration. IL can solve decision problems by imitating samples demonstrated by experts, and it does not require reward feedback from the environment, as the feedback information comes from expert decision samples. In many practical problems, obtaining expert samples is often easier and less costly than setting appropriate reward functions. Kuefler et al. have combined recurrent neural network (RNN) into the policy model, thus expanding the adversarial IL method to the context based generative adversarial IL that can use historical observation data to make decisions, and have applied this to the field of automatic driving, and have achieved safer, more stable and more efficient driving strategies [20]. In order to achieve safe, effective, and economical autonomous driving technology, Bhattacharyya et al. have proposed a multi-agent generative adversarial IL method based on parameter sharing [21]. Sukthankar and Rodrigues-Aguilar have combined a centralized multi-agent policy gradient optimization learning method based on parameter sharing on the basis of reactive IL for autonomous driving problems [22].

It can be seen that although IL has attracted strong attention from scholars in the field of intelligent transportation, till now, there are very few reports on the application of IL in response to challenges for traditional vehicle path planning algorithms, which also constitutes the research motivation of this paper.

B. IL AND ITS CATEGORIES

IL, also known as demonstration learning, solves sequential optimization decision-making problems by imitating samples demonstrated by experts [23], [24]. In many scenarios where artificial intelligence (AI) methods are used to solve sequential optimization decision-making problems, due to the involvement of a large amount of artificial engineering, people find it difficult to do manual programming to teach intelligent agents to think. Specifically, it is an extremely difficult task to guide intelligent agents through the large amount of constraint supervision information contained in certain scenario. On the contrary, humans can easily complete these tasks and provide a large number of expert decision-making example behaviors for intelligent agents. Different from RL, imitation learning does not require reward feedback from the environment, and its efficient feedback information comes from expert decision samples. Compared to finding suitable reward functions, obtaining expert decision samples is easier and more cost-effective. During recent years, IL has successfully demonstrated good practicality in the fields of autonomous vehicle [25], [26], [27], [28], [29], [30], robot control [31], [32], [33], [34], [35], [36], AlphaGo [37], recommendation system [38], internet ridesharing order distribution [39], game theory [40], [41], [42], [43], navigation task [44], [45], [46], cache management [47].

Basically, IL can be divided into two categories: behavioral cloning (BC) [48], [49] and IL via inverse RL (IRL-IL) [50], [51]. BC attempts to minimize the action difference between the agent policy and the expert policy, transforming IL task into common regression or classification task. Its main idea is to directly clone the one-step action mapping of expert samples at each state, that is, to conduct supervised learning on expert samples. It does not consider the long-term impact after the current state. It has good performance on the premise that there are enough expert samples. However, BC will gradually magnify the subtle errors in the sequential decision-making process because it does not consider the long-term impact, thus, there may be the phenomenon of compounding errors in practical applications [43], [52], which can be attributed to the limited high-quality information provided by the problem. In order to alleviate the phenomenon of compounding errors, Rajaraman et al. have proposed a dataset aggregation (DAgger) algorithm based on online learning, which continuously interacts with the environment to generate new data by cloning behavior. On these new data, DAgger can apply for examples from expert policies and train again using BC multiple times to achieve the goal of reducing compounding errors [53]. However, due to the unknown distribution of expert policies, the total number of expert examples required by DAgger may not necessarily be smaller than the number of expert examples required for BC, and the actual effect has not changed substantially. The reason is that the main idea of both BC and DAgger is to clone the single step action policy of expert samples at various states, without considering the long-term impact after the current state.

The main idea of IRL-IL is: for a given expert sample, the unknown reward function is obtained by using inverse RL; and then based on this reward function, the optimal policy is obtained by using RL. At present, the mainstream method of IRL-IL is generative adversarial IL (GAIL), which is an IL method combined with the generative adversarial network (GAN) [54], [55], [56]. The essence of GAIL is state action distribution matching, namely, it uses two networks to represent the reward function and policy, and uses adversarial methods to optimize the parameters of the two networks, that is, by solving a minimax optimization problem transformed from dual representation, to seek the optimal reward function and policy. Moreover, GAIL framework has strong inclusiveness and generalization, and classic apprenticeship learning algorithms such as feature expectation matching (FEM) [54] and game theoretical apprenticeship learning (GTAL) [55] can also be described by it. In the context of multimodal learning, GAIL has a lot of derived methods. To name a few, conditional GAIL (CGAIL) incorporates conditional constraints on modal labels in the modeling of policies and reward functions, making the model more suitable for real-world applications [56]; GAIL with auxiliary classifier (ACGAIL) introduces an auxiliary network model to enhance the ability to fully learn abstract features from labeled data samples, for effective data is often hidden as latent variables within the sample and cannot be directly obtained from expert samples [57]; information maximizing GAIL (InfoGAIL) utilizes the principle of maximizing mutual information to enhance the correlation between the samples generated by the policy and the latent variables, thereby achieving efficient learning [58]; furthermore, variational autoencoders GAIL (VAE-GAIL) uses variational autoencoders to infer modal latent variables in expert trajectory samples, and learns the latent variables that best represent the entire expert trajectory in the expert trajectory samples [46]. According to different observation mechanisms, there are also many other types of GAIL models: third person IL (TPIL) provides a third person perspective IL solution to address the phenomenon of differences in expert samples observed from different perspectives, which can meet the practical needs of IL applications under different observation perspectives [59], GAIL with recurrent policies (RP-GAIL) combines RNN into GAIL, thereby improving the ability to make decisions using contextual semantic relationships of historical observation data [20]; generative adversarial imitation from observation (GAIfO) can learn the behavioral intentions of experts from sequential observation samples and improve the reward function [60]. Corresponding to multi-agent RL, Song et al. have proposed multi-agent GAIL (MA-GAIL), which assumes that the policies of other agents are either expert policies or have satisfied Nash equilibrium, so that the learned joint policies can be applied to all intelligent agents [61]. At present, the relatively more advanced GAIL related methods are transition aware adversarial IL [62] (TAIL) and GAIL based on minimum-distance functions (MIMIC-MD) [53]. TAIL has designed better expert state-action distribution estimation than previous methods and optimized it using minimax objectives. MIMIC-MD is a generative adversarial IL algorithm based on the minimum distance function, given a transition model and expert certainty, and it is the first adversarial IL algorithm to overcome the difficulty of sample complexity.

C. CONTRIBUTION OF THIS PAPER

As mentioned in Section I-A, at present, there are quite few reports on vehicle path planning methods based on IL. In the only reports related, one type is BC based path planning methods, and the other type is TAIL or MIMIC-MD methods based on ‘‘state action distribution matching’’ under the GAIL framework. However, these methods still have some drawbacks:

(1) Although the path planning method based on BC is simple and effective, there is no good solution to the inherent problem of compounding errors.

(2) Although TAIL effectively suppresses the problem of compounding errors, the problem of sample complexity has not been effectively solved. And from the perspective of optimization, the objective functions for policy update and reward function update are both non-convex functions. Thus,their final overall optimization goals can only approximate to the saddle points, and cannot obtain the global optimal solutions.

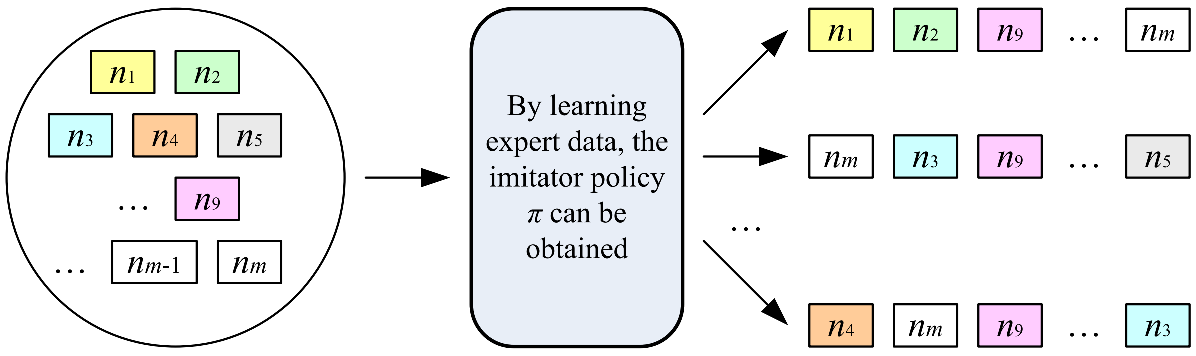

n_1
n_2
n_3
n_4
n_5
n_9
$n_{m\text{-}1}$
n_m
By learning expert data, the imitator policy π can be obtained

FIGURE 1. Imitation learning process under vehicle path abstraction.

(3) MIMIC-MD is the first GAIL method to overcome the difficulty of sample complexity. However, the optimization objective function of MIMIC-MD is nonlinear, non-convex and non-smooth, which is extremely difficult to be solved. At present, this objective function is usually transformed into a linear programming problem for solution, so as to apply linear convex relaxation approximation to non-convex and non-smooth problems, and to obtain their approximate solutions rather than solve them accurately.

The main contribution of this paper is just to try to solve the drawbacks mentioned above, and can be listed as follows:

(1) In this paper, a mixed generative adversarial IL (MixGAIL) has been proposed, and has been applied to vehicle path planning problems for the first time. MixGAIL combines the advantages of both TAIL and MIMIC-MD, and can effectively suppress the problem of compounding errors and overcome sample complexity challenges.

(2) The difficulty of MixGAIL lies in the optimization of the algorithm. In the TAIL module of MixGAIL, the optimal policy is updated under the condition of the current reward function, and this goal is consistent with the goal of RL. In this paper, a mixed policy gradient AC model with random escape term and filter optimization (MPGACEF) proposed by [19], rather than the usual policy optimization techniques, have been introduced into the field of IL. On one hand, this method unifies the gradient form of data-driven and model-driven policies, and for the first time introduces a gradual transformation between data-driven and model-driven approaches to obtain more change information; and on the other hand, considering that the optimization objective function in this step is a non-convex function, the filter technology and noise gradient technology contained in MPGACEF can perform global optimization well.

(3) Another optimization challenge for MixGAIL lies in the two steps involved in the algorithm. One of the steps is the reward function update link in TAIL module, whose essence is to solve the minimum problem of a non-convex optimization problem; The other is the optimization link of the limited optimal policy in the MIMIC-MD module, whose essence is to solve the minimum problem of a non-convex and non-smooth optimization problem, and this problem can only be converted into a linear programming problem for solution, and there is no accurate optimization scheme that depends on gradient information. Since both the above two problems are trying to solve the minimum problem of non-convex function, in this paper, a unified optimization framework has been considered, and a noise projected subgradient descent method with momentum (MNPSGD) method has been proposed. The momentum structure can accelerate the algorithm in the first order and improve its computational speed; the noise term and projection operator can overcome non-convex problems and facilitate global optimization; and the introduction of subgradient can solve the problem of lacking gradient information of non-smooth functions.

II. BASIC THEORIES

A. BASIC IDEAS OF VEHICLE PATH PLANNING BASED ON IL

At first, using IL methods to learn expert data D, thus the imitator policy π is obtained. Then, abstract the vehicle path planning problem to be solved as a problem composed of a group of nodes (n_1, n_2, $\ldots$, n_m). Feasible solution can be obtained through the imitator policy π. Each node corresponds to a solution, that is, the i-th action a_i corresponding to the i-th n_i. The sequence ($a1$, $a2$,…a_i) is called the sequence of solutions. Under the guidance of the imitator policy π, the neural network generates a solution for each node after completing each learning. Here, the vehicle path planning problem can be understood as the vehicle executing the i-th action a_i and selecting a node. By guiding vehicles to different nodes through a sequence of solutions, the function of path planning is achieved, such idea can be described in Figure 1.

B. CLASSICAL ALGORITHMS OF IL

The development of IL algorithm has roughly gone through three stages of development.

The first stage is the earliest stage of IL, where algorithms represented by BC and DAgger approximate expert policies through maximum likelihood estimation, but they are severely affected by composite errors, resulting in poor imitation performance. The BC framework has been described in detail in [49] and [53]. Its idea is relatively simple and direct, which follows the rule of "policy distribution matching". BC first uses maximum likelihood estimation (MLE) to estimate expert policy π^E from expert decision data, and the imitator policy is represented as π, both π^E and π are essentially random variables. BC hopes to make the policy π sufficiently similar to the policy π^E, and their similarity can be generally measured by Kullback-Leibler (KL) divergence. Of course, there are also issues with compound errors in the practical application of BC. Essentially, the training dataset is collected through policy π^E, but the imitator policy π learned during the evaluation process is based on the trajectory collected by policy π. This directly leads to the inconsistent distribution of accessed policy state-action in evaluation testing and state-action in training dataset. Therefore, DAgger has been proposed afterwards to increase the interaction frequency between the cloning policy and the environment, thereby reducing composite errors [53]. However, from the perspective of practical application effects, due to the limited expert policy information provided, DAgger still requires a large amount of training data costs and has not achieved any qualitative changes by comparing with BC.

The second stage is the early stage of adversarial IL, and the algorithms at this stage have already taken the embryonic form of adversarial learning, and have improved the quality of imitation and achieved lower compound errors. Actually, representative methods at this stage such as GAIL, FEM, and GTAL perform even worse than BC and DAgger. GAIL and its variants, FEM and GTAL, are essentially solving a minimax optimization problem. The optimization idea is to obtain the imitation policy through the policy gradient method under the given reward function, and then use the online projection gradient descent method to obtain the latest reward under the current imitation policy [54], [55], [56], [63], [64]. Thus, unlike BC which is based on "policy distribution matching", the core idea of GAIL is adversarial IL minimax modeling based on the "state-action distribution matching" rule. The rule refers to that the closer the distance between the state action distribution P_h^{π} of the imitation policy and the state action distribution $P_h^{\pi^E}$ of the expert policy is, the closer the goal of IL, that is, the cumulative return of the agent and the expert policy will be. The advantage of the "state-action distribution matching" rule is that even on states that have not been accessed in the dataset, it can still help select actions that are close to the expert state-action distribution, which can effectively reduce composite errors. However, as generative adversarial structures have been introduced into IL, it also leads to an increase in sample complexity. That is why in some practical scenarios, the actual effect of GAIL is not even as good as BC.

The third stage is the current stage of adversarial IL, and the representative methods such as TAIL and MIMIC-MD have truly overcome the problem of compound errors. The reason is that they have all utilized the "missing mass" property of BC, which means that they all perform an approximate estimation of BC on a subset of expert data. In terms of optimization, TAIL and MIMIC-MD both have their own problems. TAIL still uses convex optimization to solve non-convex optimization problems, so its optimal solution always approaches the saddle point, and its global optimization ability is insufficient. MIMIC-MD is essentially a non-convex and non-smooth optimization problem. The optimization policy of MIMIC-MD approximates the original problem as a linear programming problem, which can be solved in polynomial time. There is still huge room for improvement in both the optimization effect and time.

In order to facilitate the expression of the algorithms presented in this paper, the principles of TAIL and MIMIC-MD will be briefly discussed below.

1) TAIL

Same as GAIL, TAIL is also based on minimax optimization problem for algorithm deduction. The biggest difference between TAIL and GAIL lies in the estimation methods for expert state-action distribution. A more straightforward statement is that TAIL has a more refined estimation of expert state-action distribution and a smaller error compared to GAIL.

Define tr_h as the trajectory truncated to time step h, where $\mathrm{tr}_h = \{s_1, a_1; \ldots, s_h, a_h\}$, then

$$P_h^{\pi^{\mathrm{E}}}(s,a) = \sum_{\mathrm{tr}_h \in \mathrm{Tr}_h} \mathrm{P}^{\pi^{\mathrm{E}}}(\mathrm{tr}_h) \prod \{\mathrm{tr}_h(s_h, a_h) = (s,a)\} \tag{1}$$

where Tr_h is all trajectories truncated to time step h, $\mathrm{tr}_h(s_h, a_h)$ represents the state-action pairs accessed at time step. The probability of (s, a) is the probability of all (truncated) trajectories occurring containing (s, a). Redefine the set of truncated trajectories

$$\mathrm{Tr}_h^D := \{\mathrm{tr}_h : \forall h' \in [h], \mathrm{tr}_h(s_{h'}) = S_{h'}(D)\} \tag{2}$$

where $\mathrm{tr}_h(s_h')$ represents the state accessed of truncated trajectory tr_h at time step h'. That is to say, any state on the truncated trajectory contained in Tr_h^D has been accessed by dataset D. $\mathrm{P}_h^{\pi^E}(\mathrm{tr}_h)$ is the probability that the deterministic expert policy π^E triggers the occurrence of truncated trajectory tr_h.

In order to obtain a more precise estimation than GAIL, the dataset D was randomly divided into two equal parts: $D = D_1 \cup D_1^c$. If D_1 is given, there is

$$\begin{aligned} P_h^{\pi^{\mathrm{E}}}(s,a) = &\sum_{\mathrm{tr}_h \in \mathrm{Tr}_h^{D_1}} \mathrm{P}^{\pi^{\mathrm{E}}}(\mathrm{tr}_h) \prod \{\mathrm{tr}_h(s_h, a_h) = (s,a)\} \\ &+ \sum_{\mathrm{tr}_h \notin \mathrm{Tr}_h^{D_1}} \mathrm{P}^{\pi^{\mathrm{E}}}(\mathrm{tr}_h) \prod \{\mathrm{tr}_h(s_h, a_h) = (s,a)\} \end{aligned} \tag{3}$$

From dataset D_1, the expert actions on the state being accessed by D_1 can be seen. Therefore, the transition probability of Markov decision process (MDP) is used to accurately calculate the first item on the right side of equation (3). For the second term, MLE can be performed using data set D_1:

$$\frac{1}{|D_1^c|} \sum_{\mathrm{tr}_h \in D_1^c} \prod \{\mathrm{tr}_h \notin \mathrm{Tr}_h^{D_1}, \mathrm{tr}_h(s_h, a_h) = (s,a)\} \tag{4}$$

TABLE 1. **The definition of actions.**

Actions	Action vectors
Forward	$(1, 0, 0, 0)^{\mathrm{T}}$
Backward	$(0, 1, 0, 0)^{\mathrm{T}}$
Left	$(0, 0, 1, 0)^{\mathrm{T}}$
Right	$(0, 0, 0, 1)^{\mathrm{T}}$

TABLE 2. **The definition of states.**

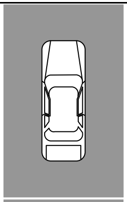
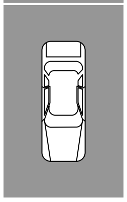
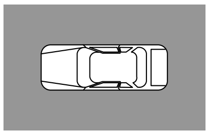
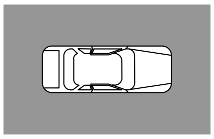
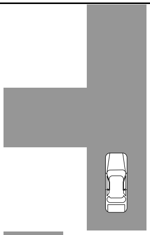
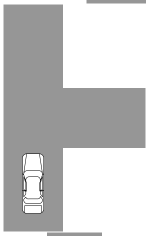
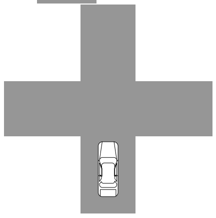
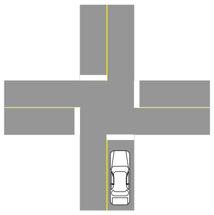
States	State descriptions	Basic state vectors
	Vehicles can only drive forward according to the current road conditions	$(1, 0, 0, 0)^{\mathrm{T}}$
	Vehicles can only drive backward according to the current road conditions	$(0, 1, 0, 0)^{\mathrm{T}}$
	Vehicles can only drive towards left according to the current road conditions	$(0, 0, 1, 0)^{\mathrm{T}}$
	Vehicles can only drive towards right according to the current road conditions	$(0, 0, 0, 1)^{\mathrm{T}}$

TABLE 3. **Composite state examples.**

States	State descriptions	Basic state vectors
	Vehicles can drive forward or turn left according to the current road conditions	$(1, 0, 1, 0)^{\mathrm{T}}$
	Vehicles can drive forward or turn right according to the current road conditions	$(1, 0, 0, 1)^{\mathrm{T}}$
	Vehicles can drive forward, turn left, or turn right according to the current road conditions	$(1, 0, 1, 1)^{\mathrm{T}}$
	Vehicles can forward, turn left, turn right, or make a U-turn according to the current road conditions	$(1, 1, 1, 1)^{\mathrm{T}}$

Thus, the new estimation $\tilde{P}_h^{\pi^E}(s, a)$ of $P_h^{\pi^E}(s, a)$ can be acquired as equation (5) shows, and the TAIL minimax optimization problem can be get as equation (6) shows.

$$\begin{aligned}\tilde{P}_h^{\pi^{\mathrm{E}}}(s, a) &= \sum_{\mathrm{tr}_h \in \mathrm{Tr}_h^{D_1}} \mathrm{P}^{\pi^{\mathrm{E}}}(\mathrm{tr}_h) \prod \{\mathrm{tr}_h(s_h, a_h) = (s, a)\} \\ &\quad + \frac{1}{|D_1^c|} \sum_{\mathrm{tr}_h \in D_1^c} \prod \{\mathrm{tr}_h \notin \mathrm{Tr}_h^{D_1}, \mathrm{tr}_h(s_h, a_h) = (s, a)\} \qquad (5)\end{aligned}$$

$$\begin{aligned}&\min_{\pi \in \Theta} \sum_{h=1}^{H} D_{\mathrm{TV}}(P_h^{\pi}, \tilde{P}_h^{\pi}) \\ &= \min_{\pi \in \Theta} \max_{\omega \in W} \sum_{(s,a)} \omega_h(s, a) \left[\tilde{P}_h^{\pi^{\mathrm{E}}}(s, a) - P_h^{\pi}(s, a)\right] \qquad (6)\end{aligned}$$

By utilizing the same optimization method as GAIL, its solution can be obtained.

2) MIMIC-MD

The idea of MIMIC-MD is to adopt behavior consistent with experts like BC on already accessed states, and it also incorporates state-action pair matching criteria. MIMIC-MD is a true IL algorithm that overcomes the difficulty of sample complexity. Moreover, MIMIC-MD also randomly divides dataset D into two equal parts: $D = D_1 \cup D_1^c$, with the optimization objective as equation (7) shows, where for dataset D_1, the set of policies generated by behavioral cloning algorithms is shown in equation (8).

$$\begin{aligned}&\min_{\pi \in \Theta_{\mathrm{mimic}}(D_1)} \sum_{h=1}^{H} \sum_{(s,a) \in S \times A} \Big| \mathrm{P}^{\pi}(\mathrm{tr}_h : \mathrm{tr}_h \notin \mathrm{Tr}_h^{D_1}, \mathrm{tr}_h(s_h, a_h) \\ &= (s, a)) - \frac{1}{|D_1^c|} \sum_{\mathrm{tr}_h \in D_1^c} \prod \{\mathrm{tr}_h \notin \mathrm{Tr}_h^{D_1}, \mathrm{tr}_h(s_h, a_h) = (s, a)\} \Big| \qquad (7)\end{aligned}$$

$$\begin{aligned}&\Theta_{\mathrm{mimic}}(D_1) \{\pi \in \Theta_{\mathrm{det}} : \forall h \in [H], s \in S_h(D_1), \\ &\quad \pi_h(\cdot|s) = \delta_{\pi_t^{\mathrm{E}}(s)}\} \qquad (8)\end{aligned}$$

where Θ_{det} represents the set containing all deterministic policies, $S_h(D_1)$ represents the set of states accessed in dataset D_1 at time step h, and $\delta_{\pi_t^E(s)}$ represents the Dirac distribution defined in the action space, which means that the distribution of the probability that each policy in the policy set $\Theta_{\mathrm{mimic}}(D_1)$

will execute expert actions on the states contained in dataset D_1.

For $\pi = \Theta_{\text{mimic}}(D_1)$, π performs expert actions on all states accessed by dataset D_1, so for any truncated trajectory tr_h in set Tr_h^{D1}, there is $\text{P}^{\pi}(\text{Tr}_h) = \text{P}^{\pi^E}(\text{Tr}_h)$. Therefore, it is only necessary to match the probability of truncated trajectories that are not included in Tr_h^{D1}.

However, it is worth noting that Formula (7) is essentially to solve the minimum problem of a non-convex and non-smooth optimization problem, but till now, this problem can only be converted into a linear programming problem [65] or an approximated extended problem [66] for solution, and there is no accurate optimization scheme relying on gradient information.

III. MIXGAIL

In this section, the mixed GAIL (MixGAIL), which aims to integrate the advantages of both TAIL and MIMIC-MD while overcoming the challenges of compounding errors and sample complexity, will be proposed and described in detail. In the process of mixing the two algorithms, the difficulty of the architecture is that TAIL and MIMIC-MD use different optimization policies for their own goals and requirements, that is, the objective functions for policy update and reward function update in TAIL are both non-convex functions, and the optimization policy of TAIL can only approximate the optimal solution to the saddle point, and cannot obtain the global optimal solution. The optimization objective function of MIMIC-MD is non-convex and non-smooth, and the conventional convex optimization method in machine learning (ML) is ineffective, so at present, it can only be solved approximately by linear programming method, and it is difficult to use the unified optimizer to optimize it with TAIL, which will bring great difficulties to the integration of the two IL methods. To overcome the above difficulties, a fast and simple non-convex and non-smooth accelerated optimization method (which can also handle non-convex and smooth problems) is proposed in this section.

A. MNPSGD

With ML entering the stage of DL, a large number of optimization problems [67], [68], [69], [70], [71], [72], [73], [74], [75], [76], and various types of accelerated optimization methods have been developed. For smooth problems, Nesterov have proposed the accelerated gradient method (AGD), which have pioneered the concept of momentum in physics and extended the derivation of a large number of gradient methods with momentum [77], [78], [79]. However, such methods are not suitable for the non-smooth problem discussed in this paper. For non-smooth problem, Nesterov [80] and Beck and Teboulle [81] have proposed the accelerated proximal gradient method (APG) [80], [81], but this method is not applicable to the non-convex problem in this paper. For non-convex problems, Rohde and Tsybakov [82], Koltchinskii et al. [83], Negahban and Wainwright [84], Jain et al. [85], Hardt and Wootters [86], Zhao et al. [87], Sun and Luo [88], and Zheng and Lafferty [89] have proposed different optimization strategies for different non-convex problems in ML [82], [83], [84], [85], [86], [87], [88], [89]. However, these methods are not accelerated optimization methods, and can cause significant computational pressure on physical operations. In fact, at present, non-convex and non-smooth problems are still open problems and there is no mature theoretical solution.

TABLE 4. The attributes of hyperparameters.

Types of hyperparameter	Representations of hyperparameter	Ranges	distribution
Weight	τ	(0, 1)	Uniform
Learning rate (policy	α_x	$(10^{-6}, 10^{-1})$	Log uniform
network $\overline{\pi}$)	α_y	$(10^{-6}, 10^{-1})$	Log uniform
Learning rate (policy	α_x	$(10^{-6}, 10^{-1})$	Log uniform
network π)	α_y	$(10^{-6}, 10^{-1})$	Log uniform

In this paper, the essential difficulty of the integration of TAIL and MIMIC-MD is to find a unified optimization framework. For this reason, a fast and concise noise projected gradient descent method with momentum is proposed for the first time, the essence of its idea is to use the combination of projected descent method and gradient descent method to solve non-smooth problems; then, momentum term is introduced to accelerate the algorithm; and finally the noise term (escape term) is introduced to jump away from the local extreme traps and saddle points. For example, to solve the following non-convex and non-smooth problem:

$$\min_{x} \Phi(x) \tag{9}$$

where Φ is the non-convex non-smooth objective function, the proposed algorithm can be described as **Algorithm 1**.

Algorithm 1

```
1: Initialize $z_1 = x_1 = x_0, t_1 = 1, t_0 = 0$.
2: for$k = 1, 2, 3, \ldots,$ do
3:     $y_k = x_k + \frac{t_{k-1}}{t_k}(z_k - x_k) + \frac{t_{k-1}-1}{t_k}(x_k - x_{k-1})$,
4:     $z_{k+1} = P_W\left(y_k - \alpha_y \partial\Phi(y_k)\right) + \eta_1$,
5:     $v_{k+1} = P_W\left(x_k - \alpha_x \partial\Phi(x_k)\right) + \eta_2$,
6:     $t_{k+1} = \frac{\sqrt{4t_k^2+1}+1}{2}$,
7:     $x_{k+1} = \begin{cases} z_{k+1}, & \mathit{If}\ \Phi(z_{k+1}) \leq \Phi(v_{k+1}) \\ v_{k+1}, & \mathit{else} \end{cases}$.
8: end for
9: Output $x_{k+1}$
```

where α_x, α_y are step hyperparameters, η_1, η_2 are standard Gaussian noise.

B. CORE ALGORITHM OF MIXGAIL

In order to integrate the respective advantages of TAIL and MIMIC-MD, their specific hybrid IL algorithm will be discussed and given. The main idea of the integrated algorithm

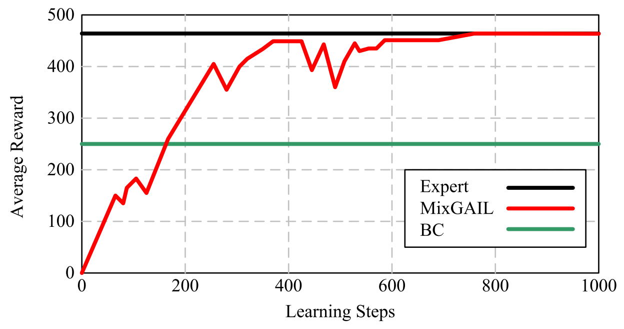
TABLE 5. Evaluation indicators.

Evaluation indicators	Applicable fields	Reasons for using certain indicator for evaluation in numerical experiment
TourL	Standardized evaluation indicators in the field of path planning	This indicator is used to describe the average travel length of path planning algorithms, and theoretically, the smaller the value, the better the decision of the path planning algorithm, and the fewer detours to take.
Gap	Standardized evaluation indicators in the field of path planning	This indicator refers to the difference between the actual travel length of the path planning algorithm and the target travel length (the optimal path length), divided by the percentage of the target travel length. It can reflect the degree of difference between the actual travel length and the target travel length, and theoretically, the smaller the value, the better.
Time	Standardized evaluation indicators in the field of path planning	This indicator is the calculation time taken by the path planning algorithm to complete the path planning task, which is used to reflect the computational speed of the algorithm. Theoretically, the smaller the value, the better.
$\rho_s(\tau)$	Standardized evaluation indicators in the field of optimization	This indicator is a classic evaluation indicator from literature [92–93], it is generated by running a solver on a set of problems and recording the information of interest. Its essence is a probability, its value means the probability of the solver will defeat other solvers. In theory, the closer the value is to 1, the better.
Average reward	Standardized evaluation indicators in the field of IL	The reason for using it as an evaluation indicator is that the biggest difference between IL and RL is that RL requires a complete reward system. The reward system helps RL to continuously conduct trial and error learning. In other words, RL is the process of deriving optimal policies through a reward system. However, establishing a reward system is time-consuming, and requires extensive industry practical experience in application scenarios. If the establishment of the reward system is incorrect, it will directly affect the final learning effect. However, adversarial IL does not require the establishment of a reward system in advance, but directly learns the optimal policy from expert data, and using adversarial frameworks. This happens to be the inverse process of RL, where rewards are received from policies. Due to the expert policy taking the optimal action for a different state each time, its reward is always the highest. If the average reward obtained from adversarial IL approaches the average reward of experts, it indicates that the strategy learned from expert data in IL is highly approaching the expert strategy. Theoretically, the closer the numerical value is to the expert reward, the better.

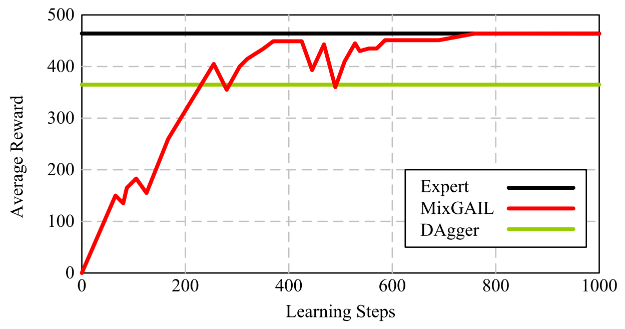
Algorithm 2

```
1: **Input** expert data $D$, maximum number of iterations $T$, step size $\eta^{(t)}$, weight coefficient $\tau \in (0, 1)$
2: **Initialize** reward function $\omega^{(1)}$
3:   divide the expert dataset into three equal parts:$D = D_1 \cup D_2 \cup D_3$,
4:   get the estimator $\tilde{P}_h^{\pi^E}$ in equation (5), namely, calculate:
     $\tilde{P}_h^{\pi^{\mathrm{E}}}(s,a) = \sum_{\mathrm{tr}_h \in \mathrm{Tr}_h^{D_1}} \mathrm{P}^{\pi^{\mathrm{E}}}(\mathrm{tr}_h) \prod \{\mathrm{tr}_h(s_h, a_h) = (s,a)\} + \frac{1}{|D_2 \cup D_3|} \sum_{\mathrm{tr}_h \in D_2 \cup D_3} \mathrm{P}^{\pi^{\mathrm{E}}}(\mathrm{tr}_h) \prod \{\mathrm{r}_h \notin \mathrm{Tr}_h^{D_1}, \mathrm{tr}_h(s_h, a_h) = (s,a)\}$,
5:   **for** $t = 1, 2, \ldots, \mathrm{T}$, **do**
6:     use MPGACEF in [19] to solve $\pi^{(t)}$ under the condition of reward function $\omega^{(t)}$,
7:     solve the corresponding state-action distribution $P_h^{\pi^{(t)}}(s,a)$ for policy $\pi^{(t)}$,
8:     use **Algorithm 1** to renew reward function $\omega^{(t+1)}$, where $F^{(t)}(\omega) = \sum_{(s,a)\in S\times A} \omega_h(s,a) \left[\tilde{P}_h^{\pi^{\mathrm{E}}}(s,a) - P_h^{\pi^{(t)}}(s,a)\right]$,
9:   **end for**
10:  solve average state-action distribution: $\bar{P}_h(s,a) = \sum_{t=1}^{T} P_h^{\pi^{(t)}}(s,a)/T$,
11:  export policy: $\bar{\pi}_h(a|s) \leftarrow \bar{P}_h(s,a)/\sum_a \bar{P}_h(s,a)$,
12: use **Algorithm 1** to solve $\tilde{\pi}_h(a/s)$, namely, solve
     $\min_{\pi \in \Theta_{\mathrm{mimic}}(D_3)} \sum_{h=1}^{H} \sum_{(s,a)\in S\times A} \left| \mathrm{P}^{\pi}\left(\mathrm{tr}_h : \mathrm{tr}_h \notin \mathrm{Tr}_h^{D_3}, \mathrm{tr}_h(s_h,a_h) = (s,a)\right) - \frac{1}{|D_1 \cup D_2|} \sum_{\mathrm{tr}_h \in D_1 \cup D_2} \prod \{\mathrm{tr}_h \notin \mathrm{Tr}_h^{D_3}, \mathrm{tr}_h(s_h,a_h) = (s,a)\} \right|$,
13: solve $\pi^*(a/s) = \tau\bar{\pi}_h(a/s) + (1-\tau)\tilde{\pi}_h(a/s)$,
14: **Output** policy $\pi^*$
```

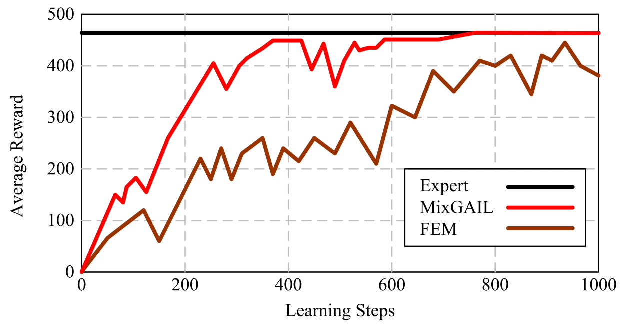
is to randomly divide the expert dataset into three equal parts: $D = D_1 \cup D_2 \cup D_3$. Let TAIL to operate D_1 and $D_2 \cup D_3$, and use the MPGACEF method in literature [19] to solve the policy for each iteration under the current reward function condition;



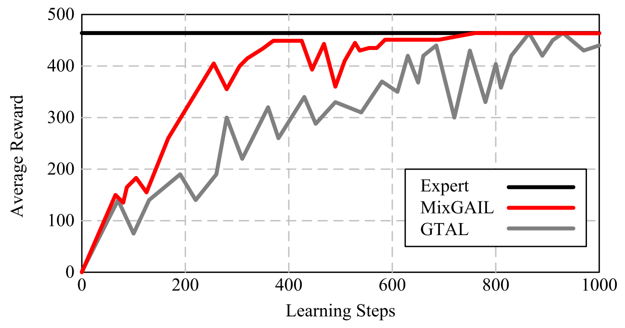
(a)



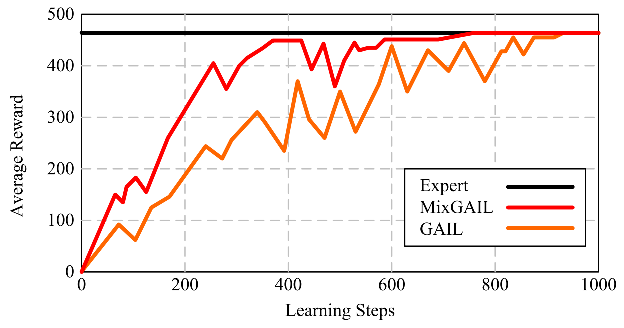
(b)



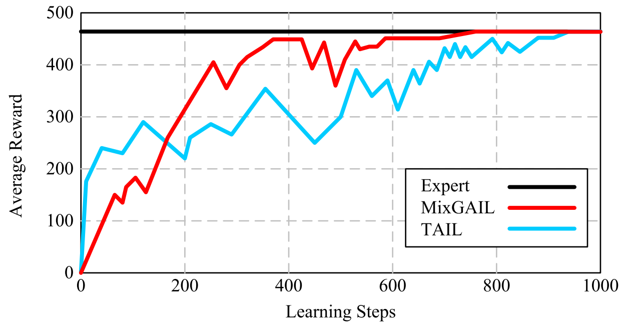
(c)



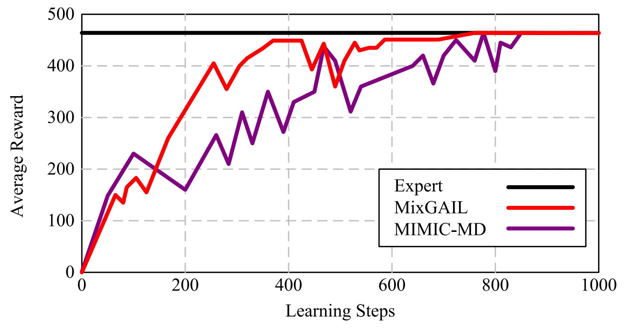
(d)



(e)



(f)



(g)

FIGURE 2. Performances on CVRP problems by using proposed MixGAIL compared with (a) BC; (b) DAgger; (c) FEM; (d) GTAL; (e) GAIL; (f) TAIL; (g) MIMIC-MD after 200 episodes of learning.

then, let MIMIC-MD to operate $D_1 \cup D_2$ and D_3. Both TAIL and MIMIC-MD utilize optimization method proposed in Section III-A for optimization; finally, the policies obtained from TAIL and MIMIC-MD are weighted and combined. The specific algorithm can be described as **Algorithm 2**.

IV. NUMERICAL EXPERIMENT

In this section, datasets of the traveling salesman problem (TSP), capacitated vehicle routing problem (CVRP) in literature [14], and multiple routing with fixed fleet problems (MRPFF) in literature [90] are used to be solved via

TABLE 6. Tour length, gap percentage and average solution time in solving CVRP problems.

Methods	CVRP20			CVRP50			CVRP100		
	TourL	Gap(%)	Time(s)	TourL	Gap(%)	Time(s)	TourL	Gap(%)	Time(s)
BC	7.82	15.18	52.91	11.35	19.17	90	19.62	20.53	236
Dagger	7.19	14.38	43.97	11.32	18.88	88	19.64	20.17	219
FEM	6.52	6.14	20.98	11.02	12.70	46	19.29	19.69	115
GTAIL	6.41	5.69	18.97	10.86	9.93	42	17.44	19.48	108
GAIL	6.22	0.90	6.40	10.71	3.15	20	16.88	6.64	42
TAIL	6.12	0.71	6.00	10.58	2.85	16	16.51	4.92	40
MIMIC-MD	6.10	0.70	5.79	10.51	2.48	15	16.54	4.17	38
MixGAIL	6.08	0.64	5.43	10.39	2.35	13	16.20	3.51	36

TABLE 7. Tour length, gap percentage and average solution time in solving MRPFF problems.

Methods	MRPFF20			MRPFF50			MRPFF100		
	TourL	Gap(%)	Time(s)	TourL	Gap(%)	Time(s)	TourL	Gap(%)	Time(s)
BC	5.95	28.07	18.97	11.11	21.18	51.16	16.08	22.00	115.12
Dagger	5.79	27.18	19.08	10.74	19.91	48.61	15.91	21.34	108.90
FEM	5.74	22.15	15.86	10.36	18.88	42.29	14.94	18.42	101.92
GTAIL	5.70	21.80	14.53	10.24	17.43	43.62	14.78	17.84	97.91
GAIL	5.49	6.18	6.06	9.94	8.28	19.29	14.58	8.31	37.12
TAIL	5.45	5.85	5.60	9.29	7.78	17.17	14.29	8.00	33.09
MIMIC-MD	5.43	5.51	5.00	9.27	7.61	16.62	14.26	7.61	32.40
MixGAIL	5.40	4.91	4.55	9.02	7.30	15.89	14.13	7.45	31.72

TABLE 8. Tour length, gap percentage and average solution time in solving TSP problems.

Methods	TSP20			TSP50			TSP100		
	TourL	Gap(%)	Time(s)	TourL	Gap(%)	Time(s)	TourL	Gap(%)	Time(s)
BC	3.89	4.70	8.95	5.74	16.06	22.95	10.78	40.16	49.88
Dagger	3.83	4.74	8.60	5.71	13.95	22.89	10.36	38.27	45.94
FEM	3.84	4.70	8.20	5.58	13.12	22.20	10.34	40.07	42.97
GTAIL	3.84	4.71	8.00	5.57	12.02	22.19	9.82	36.68	39.87
GAIL	3.85	0.57	1.34	5.26	1.31	2.01	8.07	1.94	7.24
TAIL	3.85	0.45	1.23	5.18	1.31	2.08	8.12	1.72	6.33
MIMIC-MD	3.84	0.30	1.16	5.16	1.24	2.06	7.95	1.51	6.25
MixGAIL	3.83	0.26	1.00	5.10	1.16	1.98	7.83	1.33	6.18

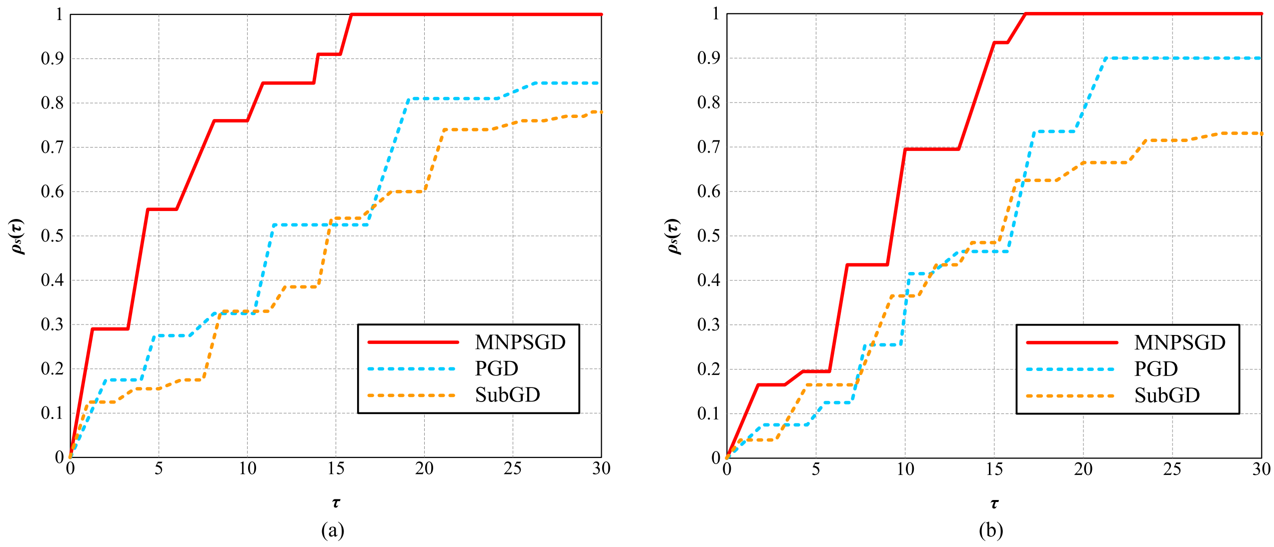


FIGURE 3. Performances of (a) CPU time; (b) NIT time for SubGD, PGD and MNPSGD on CVRP problem optimizations.

MPGACEF, so as to obtain the state-action dataset for each problem. Considering the efficiency of MPGACEF in solving the above three problems, these three state-action datasets are considered as expert datasets. These three expert datasets are provided to BC, DAgger, FEM, GTAL, GAIL, TAIL, MIMIC-MD, and the proposed MixGAIL in this paper to learn and imitate expert policies, and then solve the TSP, CVRP, and MRPFF problems.

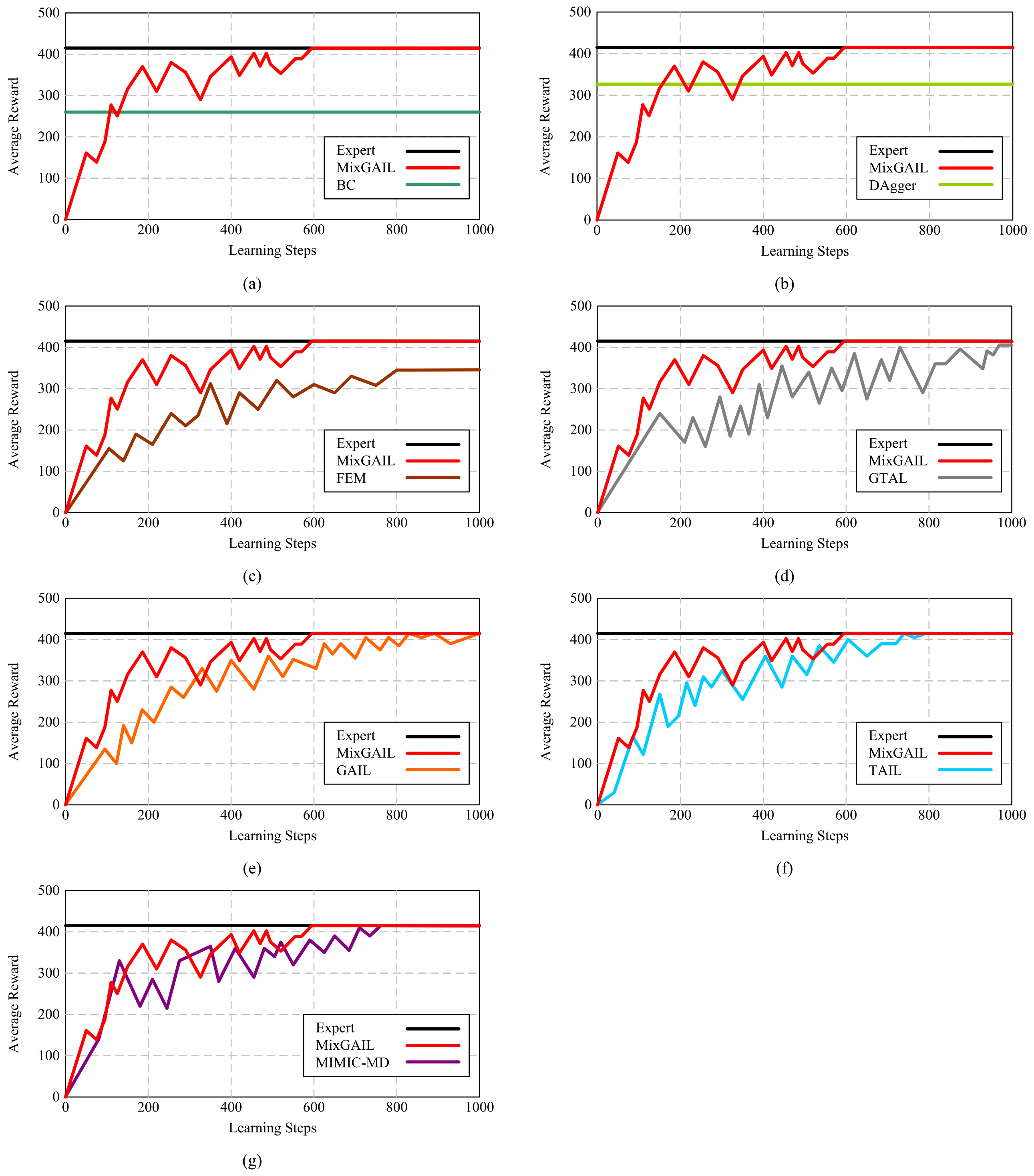


FIGURE 4. Performances on MRPFF problems by using proposed MixGAIL compared with (a) BC; (b) DAgger; (c) FEM; (d) GTAL; (e) GAIL; (f) TAIL; (g) MIMIC-MD after 200 episodes of learning.

The experimental data learned by each model in this section is the expert trajectories (the state-action sequences generated by experts solving specified tasks). 1000 expert trajectories for the CVRP20 scenario, 1000 expert trajectories for the CVRP50 scenario, and 1000 expert trajectories for the CVRP100 scenario are provided, and the number and proportion of data provided for the MRPFF20 and TSP20; MRPFF50 and TSP50; MRPFF100 and TSP100 scenarios are corresponding to the CVRP20, CVRP50, CVRP100 problem, so there are a total of

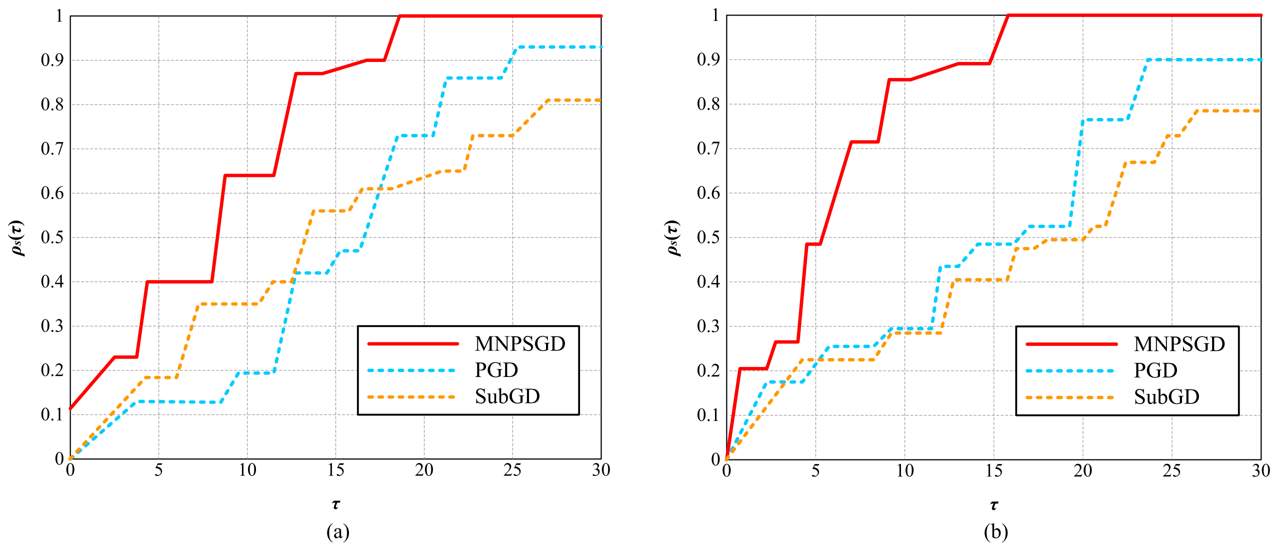


FIGURE 5. Performances of (a) CPU time; (b) NIT time for SubGD, PGD and MNPSGD on MRPFF problem optimizations.

9000 expert trajectories. Each expert trajectory consists of 200 state-action pairs generated by expert policies, each state action pair includes a 4-dimensional state vector and a 4-dimensional action vector. So the total number of our experimental data is 3600000 4-dimensional vectors.

In fact, the definition of state-action vectors is not unique. The action definition of vehicle path planning generally adopts the 8-direction or 4-direction definition method. The definition of state is generally based on the existing experimental data situation. For example, if the existing data is image data, convolutional neural networks (CNNs) or graph neural networks (GNNs) are generally used to extract features from image samples, and the effective feature data vector obtained is considered as the state vector.

Of course, obtaining global image data is costly, so it is generally possible to define state vectors based on vehicle driving road features, especially road intersection features. Based on the characteristics of the existing experimental data in this paper, the action is defined by 4-dimentional vectors, 4 unit vectors can express forward, backward, left, and right directions, as shown in Table 1. The use of this definition method is concise and clear, moreover, the sparsity of the unit vector data helps subsequent network algorithm steps to learn clearly and quickly.

Moreover, in this paper, the state is also defined by using 4-dimentional vectors, 4 unit vectors can express the basic driving directions that the vehicle can choose under the current road conditions, as shown in Table 2. Based on the specific road conditions, in each 4-dimensional basic state vector shown in Table 2, the current drivable direction is defined as 1, while the non-drivable direction is defined as 0. It should be pointed out that the real road conditions are much more complex than the 4 unit vectors shown in Table 2, but they can all be represented by adding up these 4 unit vectors. Table 3 shows 4 composite state examples.

TABLE 9. Tour length, gap percentage and average solution time in solving the problem in real-world scenario.

Methods	TourL	Gap (%)	Times (s)
BC	13.97	6.94	423.39
DAgger	13.91	6.54	422.62
FEM	13.76	5.52	244.81
GTAL	13.73	5.32	233.47
GAIL	13.25	1.89	91.60
TAIL	13.17	1.29	88.23
MIMIC-MD	13.14	1.07	86.16
MixGAIL	13.12	0.91	79.84

All algorithms have been used to train the same NN architecture for all tasks. It includes a policy network $\bar{\pi}$, which includes two hidden layers, each with 100 neurons, and a nonlinear *tanh*activation function enabled in the middle. The policy network $\tilde{\pi}$, and the discriminant networks corresponding to the two policy networks also use the same hidden layer architecture. All networks are always randomly initialized at the beginning of each experiment.

The parameters of the entire model include network parameters and hyperparameters. Due to the fact that the optimization objective of IL is to find the optimal policy and reward, in essence, it optimizes the corresponding policy network the parameters and discriminant network parameters. Specifically, network parameters include the parameters contained in the policy network $\bar{\pi}$, policy network $\tilde{\pi}$, and discriminator network. The input state vector of the policy network $\bar{\pi}$ is a 4-dimensional vector, so each neuron in the first hidden layer includes 4 weight parameters and 1 bias parameter. The second hidden layer neuron is connected to the first hidden layer neuron, so each neuron includes 100 weight parameters and 1 bias parameter. After the two hidden layers, a *softmax* function will be connected to output a 4-dimensional action vector. This *softmax* function will contain 100 weight parameters and 1 bias parameter. So the

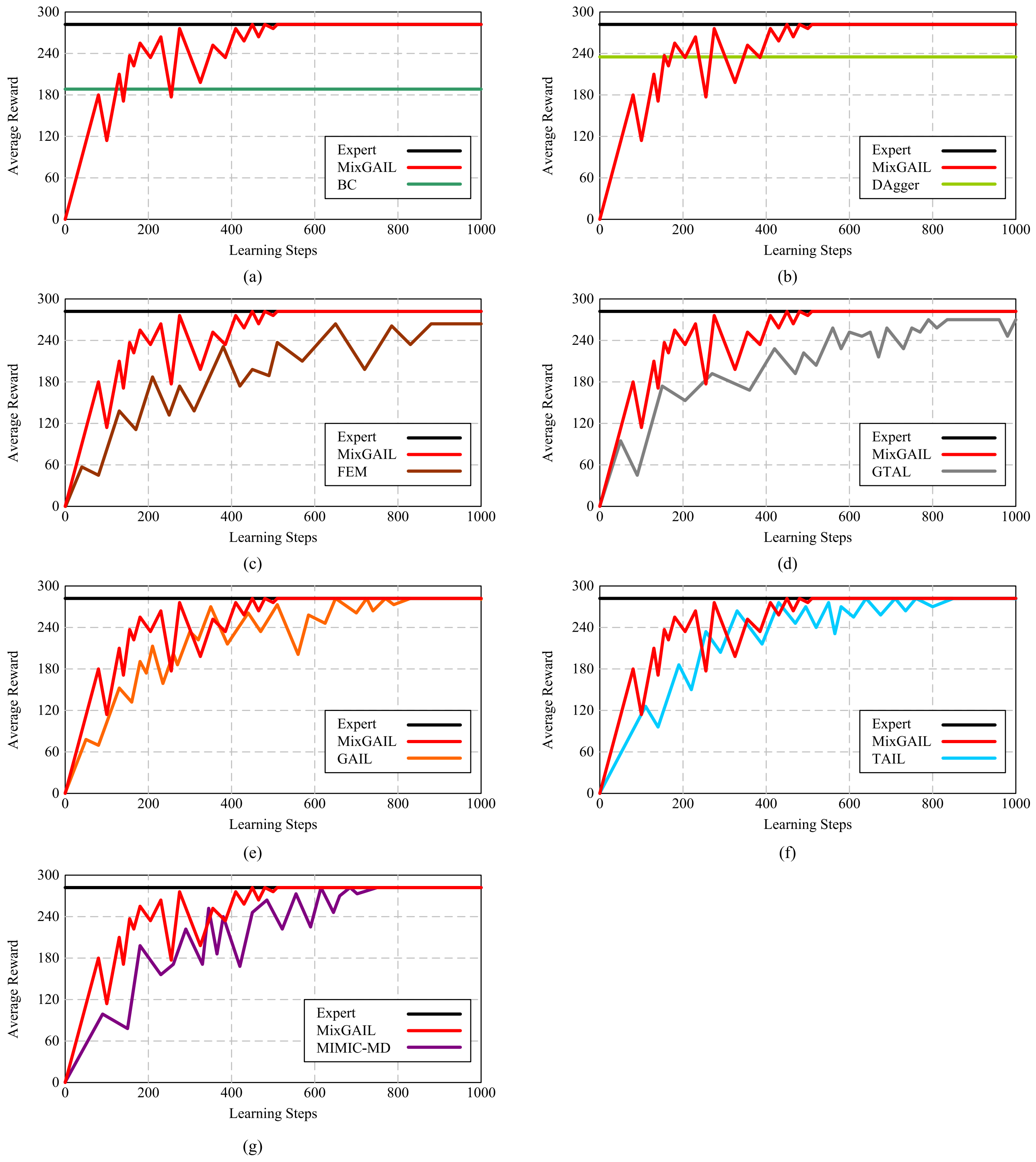


FIGURE 6. Performances on TSP problems by using proposed MixGAIL compared with (a) BC; (b) DAgger; (c) FEM; (d) GTAL; (e) GAIL; (f) TAIL; (g) MIMIC-MD after 200 episodes of learning.

policy network $\bar{\pi}$ has a total of 10701 parameters. The policy network $\tilde{\pi}$ has the same structural settings as the policy network, so it also has 10701 parameters.A 4-dimensional action vector output from a certain policy network, combined with the 4-dimensional state vector and 4-dimensional action vector of expert data, indicates that each neuron in the first hidden layer of the discriminative network includes 12 weight parameters and 1 bias parameter. And, each neuron in the second hidden layer includes 100 weight parameters and 1 bias parameter. The discriminant network finally uses the *softmax* function to output the discriminant probability, so the *softmax* function will contain 100 weight parameters

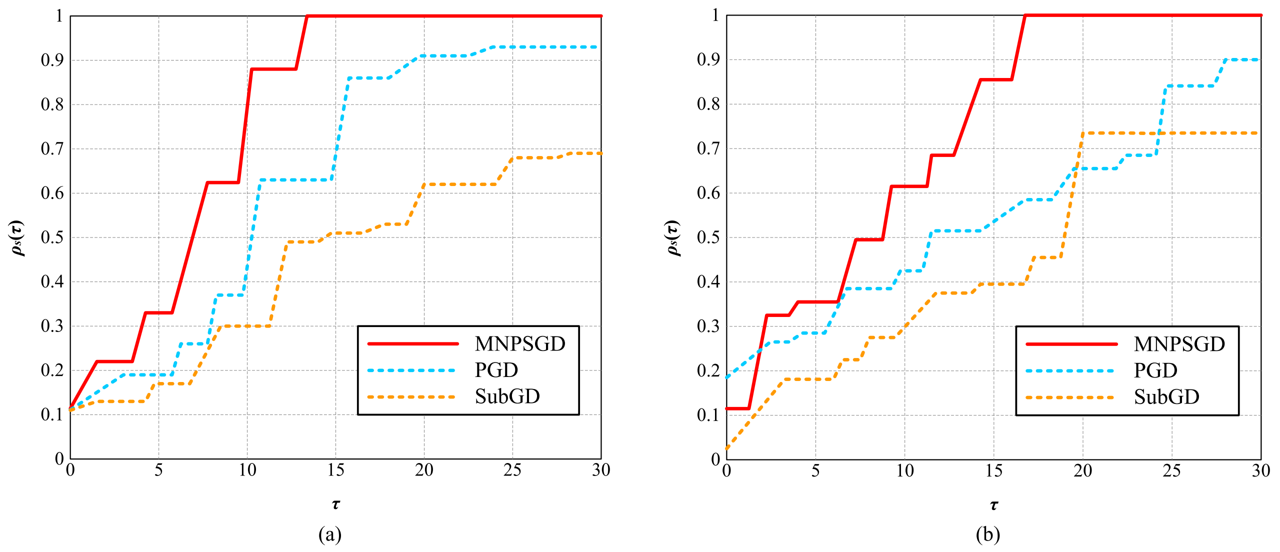


FIGURE 7. Performances of (a) CPU time; (b) NIT time for SubGD, PGD and MNPSGD on TSP problem optimizations.

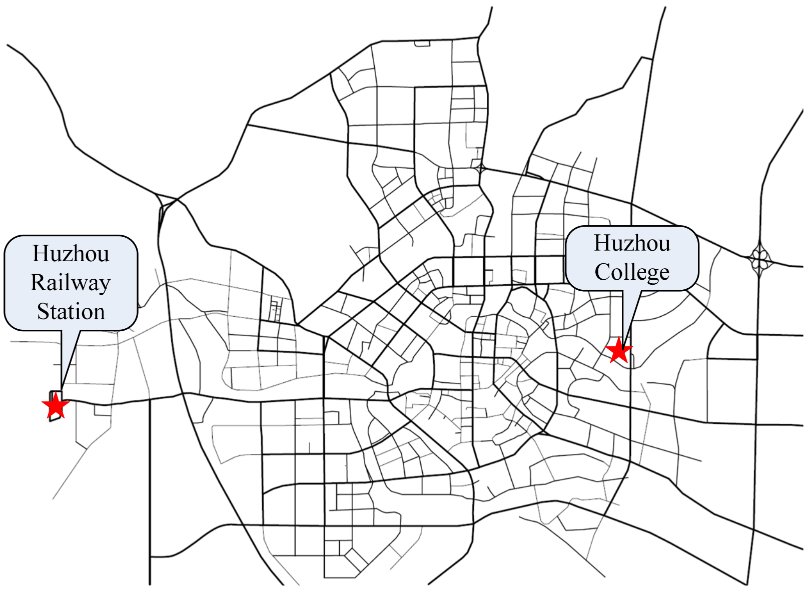


FIGURE 8. The map of Huzhou with the starting and ending points in the problem of real-world scenario marked.

and 1 bias parameter. In summary, the discriminative networks corresponding to the two policy networks have a total of 23002 parameters, and there are a total of 44404 network parameters. The network parameters are optimized and solved through **Algorithm 1**.

The hyperparameter τ linearly combines the action output vectors of policy network $\bar{\pi}$ and $\tilde{\pi}$, and its distribution is uniform. The policy netwok $\bar{\pi}$ and $\tilde{\pi}$ are optimized independently through **Algorithm 1**, so there will be 2 sets of (α_x, α_y) learning rate parameters. In summary, there are 5 hyperparameters in total. Table 4 provides the necessary attributes of each hyperparameter.

This paper adopts the mainstream Bayesian optimization method to solve the above hyperparameters [91]. The action vectors output by the policy network $\bar{\pi}$and the policy network $\tilde{\pi}$ are linearly combined, as described in **Algorithm 2**, to obtain the action output vector of policy π^*. An L2 error loss function is constructed with the expert action vector, which is used as the optimization objective function, and Bayesian optimization method is used to optimize the search area $(10^{-6}, 10^{-1}) \times (0, 1)$. Due to the unknown form of the objective function and its derivative, it is essentially a black box optimization problem, but this is just the problem that Bayesian optimization method is good at. To solve the above 5 hyperparameters, Bayesian optimization method is set as follows: firstly, the kernel function of the Bayesian linear regression part of the Bayesian optimization method is set to a Gaussian kernel function; secondly, the sampling function

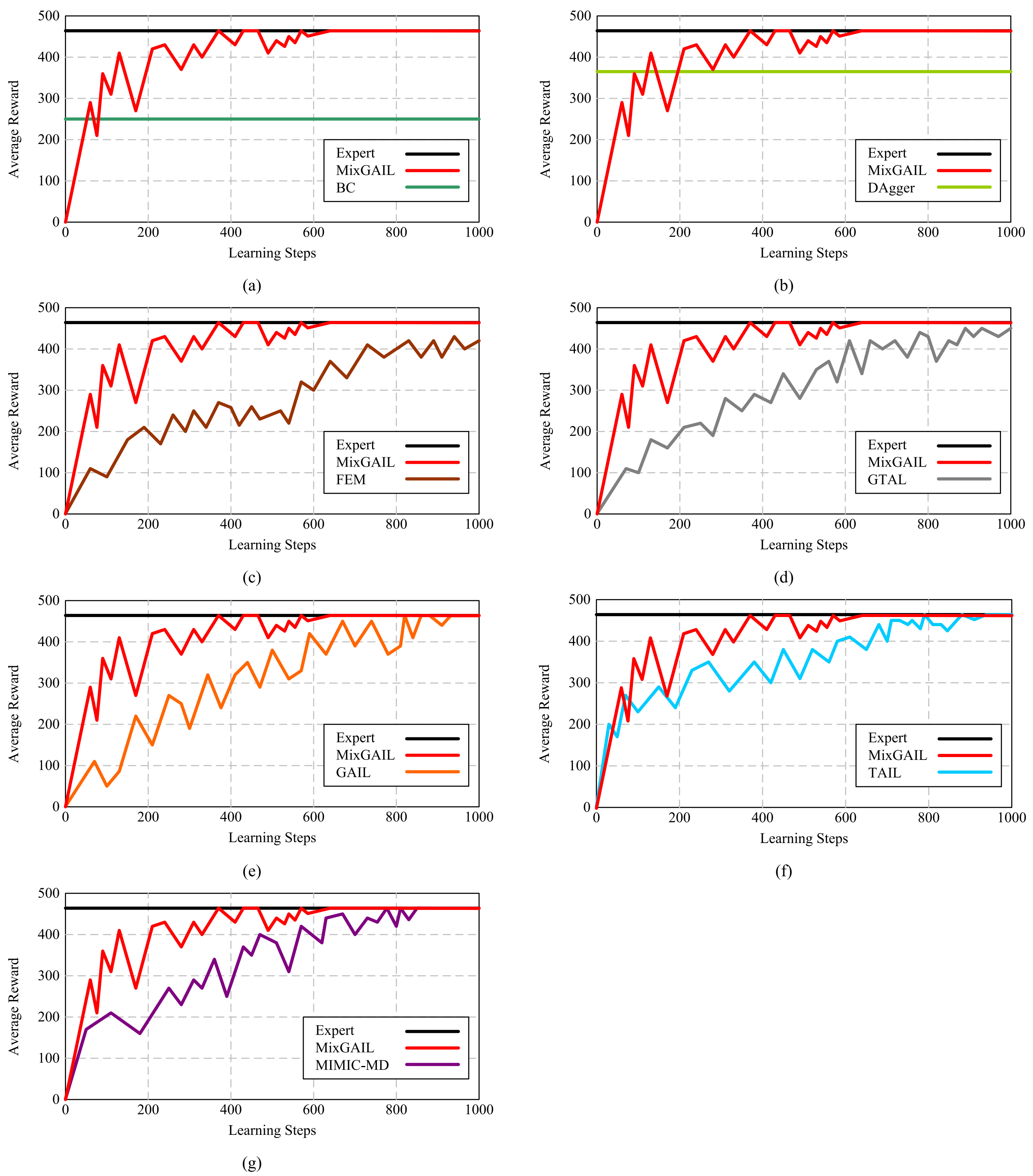


FIGURE 9. Performances on the problem of real-world scenario by using proposed MixGAIL compared with (a) BC; (b) DAgger; (c) FEM; (d) GTAL; (e) GAIL; (f) TAIL; (g) MIMID-MD after 200 episodes of learning.

in Bayesian optimization methods is set to probability of improvement.

For subsequent numerical experiments, the following standardized indicators including tour length (TourL), gap percentage (Gap), algorithm operation time (Time), $\rho_s(\tau)$ and average reward are introduced as shown in Table 5.

A. CVRP PROBLEM

In this section, imitated policies learned from BC, DAgger, FEM, GTAL, GAIL, TAIL, MIMIC-MD, and MixGAIL according to expert policy are used to solve the CVRP problem, 1000 random instances of CVRP20, CVRP50, and CVRP100 instances have been given. Table 6 reports the

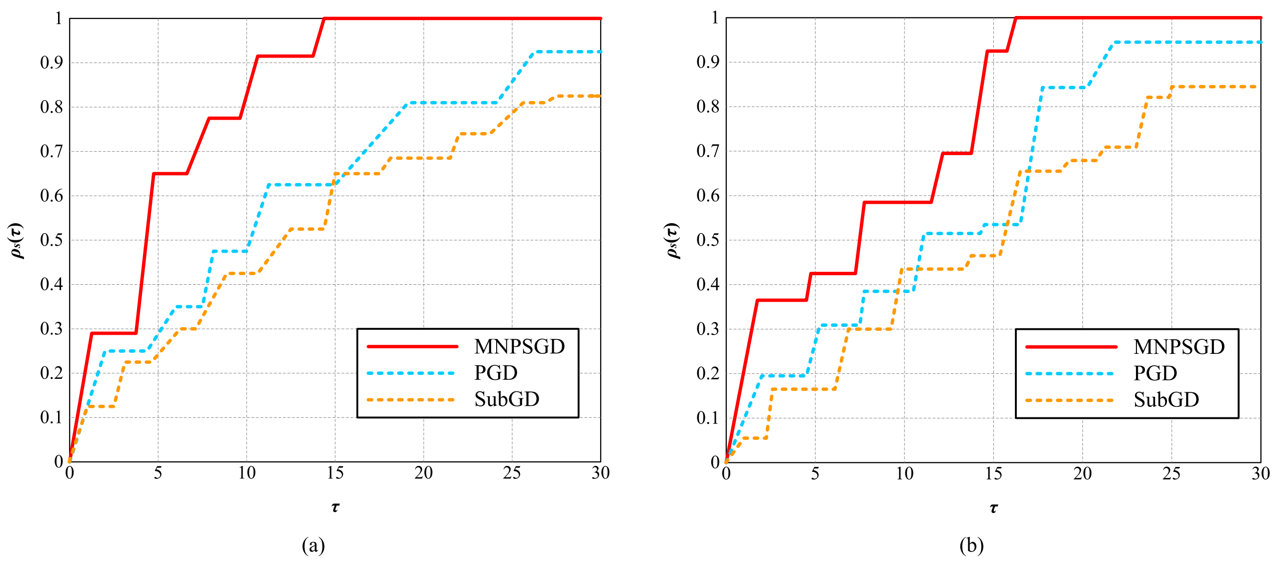


FIGURE 10. **Performances of (a) CPU time; (b) NIT time for SubGD, PGD and MNPSGD on the problem optimizations of real-world scenario.**

running time in seconds ("s" in short). It can be seen that all indicators for MixGAIL are superior to existing methods.

Figure 2 shows the approximation of average rewards to expert rewards by BC, DAgger, FEM, GTAL, GAIL, TAIL, MIMIC-MD, and MixGAIL in the CVRP problem after 200 episodes of learning, with the vertical axis average reward and the horizontal axis learning steps. It can be intuitively seen that MixGAIL has the best approximation for expert rewards.

Due to the fact that the optimization objective function in this paper requires the use of non-smooth optimization techniques for optimization, here in numerical experiment, two most common non-smooth optimization methods – projected gradient descent (PGD) and subgradient descent (SubGD) in ML are chosen to compare their optimization performance with proposed MNPSGD method. For convention of expression, the number of iterations has been recorded as NIT. Figure 3 shows the performance of three optimization methods in CVRP problems, it is evident that MNPSGD outperforms other methods in optimizing performance in both CPU time and NIT time.

B. MRPFF PROBLEM

In this section, imitated policies learned from BC, DAgger, FEM, GTAL, GAIL, TAIL, MIMIC-MD, and MixGAIL according to expert policy are used to solve the MRPFF problem, the application instance of 20, 50 and 100 user nodes have been given. Table 7 reports the running time in seconds. It can also be seen that all indicators for MixGAIL are superior to existing methods.

Figure 4 shows the approximation of average rewards to expert rewards by BC, DAgger, FEM, GTAL, GAIL, TAIL, MIMIC-MD, and MixGAIL in the MRPFF problem after 200 episodes of learning, with the vertical axis average reward and the horizontal axis learning steps. It can also be intuitively seen that MixGAIL has the best approximation for expert rewards.

Figure 5 shows the performance of three optimization methods in MRPFF problems, it is also evident that MNPSGD outperforms other methods in optimizing performance in both CPU time and NIT time.

C. TSP PROBLEM

In this section, imitation expert strategies learned from BC, DAgger, FEM, GTAL, GAIL, TAIL, MIMIC-MD, and MixGAIL are used to solve the TSP problem, the application instance of 20, 50 and 100 user nodes have been given. Table 8 reports the running time in seconds. It can still be seen that all indicators for MixGAIL are superior to existing methods.

Figure 6 shows the approximation of average rewards to expert rewards by BC, DAgger, FEM, GTAL, GAIL, TAIL, MIMIC-MD, and MixGAIL in the TSP problem after 200 episodes of learning, with the vertical axis average reward and the horizontal axis learning steps. It can still be intuitively seen that MixGAIL has the best approximation for expert rewards.

Figure 7 shows the performance of three optimization methods in TSP problems, it is still evident that MNPSGD outperforms other methods in optimizing performance in both CPU time and NIT time.

D. REAL-WORLD SCENARIO

In this section, real-world scenario case is presented in the city of Huzhou, China. Vehicles are required to travel from Huzhou College to Huzhou Railway Station with a total distance at about 13 kilometers, as shown in Figure 8. In the case study, experienced human drivers drive the vehicle and generate 200 trajectories as expert data, each trajectory contains 200 action-state pairs, and it is assumed that the average reward value for human experts is the highest. Furthermore, all of the algorithm network models are trained and tested, and the actual performance and average reward of various IL tasks are evaluated based on standardized indicators. The performance of the optimization algorithm proposed in this paper is also evaluated based on standardized incidators.

Imitation expert strategies learned from BC, DAgger, FEM, GTAL, GAIL, TAIL, MIMIC-MD, and MixGAIL are used to solve the problem in real-world scenario, Table 9 has also shown that, all indicators for MixGAIL are superior to existing methods.

Figure 9 shows the approximation of average rewards to expert rewards by BC, DAgger, FEM, GTAL, GAIL, TAIL, MIMIC-MD, and MixGAIL in the problem of real-world scenario after 200 episodes of learning, with the vertical axis average reward and the horizontal axis learning steps. It can be intuitively seen that MixGAIL has the best approximation for human expert rewards.

Figure 10 shows the performance of three optimization methods in the problem of real-world scenario, it is still evident that MNPSGD outperforms other methods in optimizing performance in both CPU time and NIT time.

V. CONCLUSION

In order to overcome the bottleneck of composite error and sample complexity in TAIL and MIMIC-MD, this paper fused these two GAIL methods mentioned above, but it is not just a simple fusion on the algorithm logic level. The deep-seated fundamental obstacle to the fusion of TAIL and MIMIC-MD lies in the need for a unified optimization algorithm framework. In this paper, the proposed non-convex and non-smooth accelerated optimization method MNPSGD is used to overcome this bottleneck, and then leads up to the proposed MixGAIL whichis in essence a hybrid form of the TAIL and MIMIC-MD. By applying the proposed MixGAIL method in CVRP, MRPFF, TSP, and real-world path planning problems respectively, it can be seen that MixGAIL outperforms various current IL methods in approaching expert rewards in standardized vehicle path planning indicators and average rewards.

At present, the cutting-edge methods of path planning mainly revolve around the extension of RL theory, but RL requires a sound reward mechanism and system. To obtain this reward mechanism and system, it often requires a large amount of professional knowledge, experience, and labor time costs, which are largely manual, time-consuming, and resource intensive. Moreover, even if IL methods that can directly learn expert data layout policies without setting rewards in advance, have been introduced into the field of path planning research, their algorithm effectiveness is often constrained by composite errors and sample complexity. In this background, the MixGAIL proposed in this paper introduces IL methods into vehicle path planning problems, effectively overcoming the challenges of composite errors and sample complexity.

From the perspective of the future evolution trend of this research direction, at the macro level, RL is accelerating the integration of quantum field technology, and it is particularly noteworthy that the deep intersection of RL and variational quantum circuit technology has greatly improved the computational ability of RL. IL, which can be regarded as the inverse problem of RL, also has great potential to integrate with quantum technology in theory. It is foreseeable that its theoretical research will occupy a place in the forefront of vehicle path planning methods. At the micro level, MixGAIL can be combined with many branches of AI. If MixGAIL is combined with curriculum learning, it can be trained from simple to complex, and expert strategies can be learned more efficiently. For example, the integration of MixGAIL with multitasking and multi-agent methods can expand the breadth and depth of path planning application scenarios.

APPENDIX A ABBREVIATIONS

AC	Actor-critic.
ACGAIL	Generative adversarial imitation learning with auxiliary classifier.
AGD	Accelerated gradient method.
AI	Artificial intelligence.
APG	Accelerated proximal gradient method.
BC	Behavioral Cloning.
CGAIL	Conditional generative adversarial imitation learning.
CNN	Convolutional neural network.
CVRP	Capacitated vehicle routing problem.
DAgger	Dataset aggregation.
DL	Deep learning.
DNN	Deep neural network.
DRL	Deep reinforcement learning.
FEM	Feature expectation matching.
GAIfO	Generative adversarial imitation from observation.
GAIL	Generative adversarial imitation learning.
GAN	Generative adversarial network.
GNN	Graph neural network.
GTAL	Game theoretic apprenticeship learning.
IL	Imitation learning.
InfoGAIL	Information maximizing generative adversarial imitation learning.
IRL-IL	Imitation learning via inverse reinforcement learning.
L2I	Learning to improve.
lkh3	3rd-generation Lin-Kernighan-Helsgaun TSP solver.
MA-GAIL	Multi-agent generative adversarial imitation learning.
MDP	Markov decision process.
MIMIC-MD	Generative adversarial imitation learning based on minimum-distance functions in the setting where the transition model is given and the expert is deterministic.
MixGAIL	Mixed generative adversarial imitation learning.

ML	Machine learning.
MLE	Maximum likelihood estimation.
MNPSGD	Noise projected subgradient descent method with momentum.
MPGACEF	Mixed policy gradient actor-critic model with random escape term and filter optimization.
MRPFF	Multiple routing with fixed fleet problem.
NeuRewriter	The algorithm that learns a policy to pick heuristics and rewrite the local components of the current solution to iteratively improve it until convergence.
NN	Neural network.
PGD	Projection gradient descent method.
PSO	Particle swarm optimization.
RL	Reinforcement learning.
RNN	Recurrent neural network.
RP-GAIL	Generative adversarial imitation leaerning with recurrent policies.
SubGD	Subgradient descent method.
TAIL	Transition-aware adversarial imitation learning.
TPIL	Third-person imitation learning.
VAE-GAIL	Variational autoencoders generative adversarial imitation learning.

APPENDIX B NOMENCLATURE

D	Expert data.
π	Imitator policy.
n_i	An abstract node for solving vehicle path planning problem.
l	The number of abstract nodes.
a_i	The i-th action.
π^E	Estimated expert policy.
D_{KL}	Kullback-Leibler (KL) divergence.
Θ	The set of all random policies.
P_h^{π}	The state-action distribution of imitation policy π at time step h.
$P_h^{\pi^E}$	The state-action distribution of the expert policy at time step h.
D_{TV}	Total variation distance.
h	Time step.
H	Maximum time step.
tr_i	The i-th state action sequence in expert data.
$\hat{P}_h^{\pi^E}$	The estimated $P_h^{\pi^E}$ value from maximum likelihood estimation at time step h.
Π	Indicative function.
A	Action space.
a	An action in the action space.
a_h^i	An action in the i-th state action sequence in expert data at time step h.
S	State space.
s	A state in the state space.
s_h^i	An state in the i-th state action sequence in expert data at time step h.
m	The number of state action sequence in expert data.
$\hat{\pi}$	Optimal imitation policy.
ω	Reward function.
ω_h	The reward function at time step $h$when π is given.
W	The set of reward function.
$\pi^{(t)}$	Imitation policy for the t-th iteration.
$\omega^{(t)}$	Reward function for the t-th iteration.
$F^{(t)}(\omega)$	Optimization objective function of GAIL.
η_ω	Step length.
P_W	Projection operator.
u	Intermediate variable.
tr_h	Trajectory truncated to time step h.
Tr_h	All trajectories truncated to time step h.
$\mathrm{tr}_h(s_h, a_h)$	State-action pairs accessed at time step h.
Tr_h^D	The set of truncated trajectories which contains any state on the truncated trajectory that has been accessed by dataset D.
$\mathrm{tr}_h(s_{h'})$	State accessed of truncated trajectory tr_h at time step h'.
D_1, D_2, D_3	Expert data subsets.
D_1^c	The complement of D_1.
$P^{\pi^E}(\mathrm{tr}_h)$	Probability of truncated trajectory tr_h induced by expert policy π^E.
$\tilde{P}^{\pi^E}(s, a)$	Refined estimation of $P_h^{\pi^E}(s, a)$ in TAIL.
$\Theta_{\mathrm{mimic}}(D_1)$	The set of policies generated by the behavioral cloning algorithm for dataset D_1.
Θ_{det}	The set containing all deterministic policies.
$S_h(D_1)$	The set of states accessed in dataset D_1 at time step h.
$\delta_{\pi_t^{\mathrm{E}}(s)}$	The Dirac distribution defined in the action space (which means that the distribution of the probability that each policy in the policy set $\Theta_{\mathrm{mimic}}(D_1)$ will execute expert actions on the states contained in dataset D_1).
α_x, α_y	Step size hyperparameter.
k	The number of iterations.
z_k, x_k, y_k, t_k, v_k	Parameters that participate in iterations.
Φ	Non-convex non-smooth objective function.
$\partial\Phi(\cdot)$	Subgradient of function $\Phi(\cdot)$.
η_1, η_2	Standard Gaussian noise.
T	Maximum number of iterations.
$\eta^{(t)}$	Step size of the t-th iteration.
τ	Weight coefficient.

$P_h^{\pi^{(t)}}$	The corresponding state-action distribution under the premise of policy $\pi^{(t)}$.
$\bar{P}_h$	Average state-action distribution.
$\bar{\pi}_h$	Output policy of TAIL module in MixGAIL.
$\tilde{\pi}_h$	Output policy of MIMIC-MD module in MixGAIL.
π^*	Final output policy of MixGAIL.
$\rho_s(\tau)$	The standard indicator of the possibility of solution.

REFERENCES

[1] D. Bertsekas, *Dynamic Programming and Optimal Control*, 4th ed. Belmont, MA, USA: Athena Scientific, 2017, pp. 1–46.

[2] K. Heßler and S. Irnich, "A branch-and-cut algorithm for the soft-clustered vehicle-routing problem," *Discrete Appl. Math.*, vol. 288, pp. 218–234, Jan. 2021.

[3] H. He, H. Daume III, and J. M. Eisner, "Learning to search in branch-and-bound algorithms," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, Cambridge, MA, USA, 2014, pp. 3293–3301.

[4] A. Ahmed and J. Sun, "Bilayer local search enhanced particle swarm optimization for the capacitated vehicle routing problem," *Algorithms*, vol. 11, no. 3, p. 31, Mar. 2018.

[5] E. Atashpaz-Gargari and C. Lucas, "Imperialist competitive algorithm: An algorithm for optimization inspired by imperialistic competition," in *Proc. IEEE Congr. Evol. Comput.*, Singapore, Sep. 2007, pp. 4661–4667.

[6] K. Helsgaun, "An extension of the Lin-Kernighan-Helsgaun TSP solver for constrained traveling salesman and vehicle routing problems," Roskilde Univ., Roskilde, Denmark, Tech. Rep., 2017.

[7] J. Ji, A. Khajepour, W. W. Melek, and Y. Huang, "Path planning and tracking for vehicle collision avoidance based on model predictive control with multiconstraints," *IEEE Trans. Veh. Technol.*, vol. 66, no. 2, pp. 952–964, Feb. 2017.

[8] A. Pekarovskiy, T. Nierhoff, S. Hirche, and M. Buss, "Dynamically consistent online adaptation of fast motions for robotic manipulators," *IEEE Trans. Robot.*, vol. 34, no. 1, pp. 166–182, Feb. 2018.

[9] K. Berntorp, "Path planning and integrated collision avoidance for autonomous vehicles," in *Proc. Amer. Control Conf. (ACC)*, May 2017, pp. 4023–4028.

[10] Y. Zhang, M. Xu, Y. Qin, M. Dong, L. Gao, and E. Hashemi, "MILE: Multiobjective integrated model predictive adaptive cruise control for intelligent vehicle," *IEEE Trans. Ind. Informat.*, vol. 19, no. 7, pp. 8539–8548, Jul. 2023.

[11] Y. Zhang, Y. Lin, Y. Qin, M. Dong, L. Gao, and E. Hashemi, "A new adaptive cruise control considering crash avoidance for intelligent vehicle," *IEEE Trans. Ind. Electron.*, vol. 71, no. 1, pp. 688–696, May 2023.

[12] O. Vinyals, M. Fortunato, and N. Jaitly, "Pointer networks," in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2015, pp. 2692–2700.

[13] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst. (NIPS)*, Cambridge, MA, USA, 2014, pp. 3104–3112.

[14] M. Nazari, A. Oroojlooy, L. Snyder, and M. Takac, "Reinforcement learning for solving the vehicle routing problem," in *Proc. 31th Int. Conf. Neural Inf. Process. Syst. (NeurIPS)*, Montreal, QC, Canada, 2018, pp. 1–11.

[15] W. Kool, H. van Hoof, and M. Welling, "Attention, learn to solve routing problems," in *Proc. 7th Int. Conf. Learn. Represent.*, New Orleans, LA, USA, 2019, pp. 1–25.

[16] Y. D. Kwon, J. Choo, B. Kim, I. Yoon, Y. Gwon, and S. Min, "POMO: Policy optimization with multiple optima for reinforcement learning," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, Vancouver, QC, Canada, 2020, pp. 1–11.

[17] X. Chen and Y. Tian, "Learning to perform local rewriting combinatorial optimization," in *Proc. 32nd Int. Conf. Neural Inf. Process. Syst.*, Vancouver, QC, Canada, 2019, pp. 6281–6292.

[18] H. Lu, X. Zhang, and S. Yang, "A learning-based iterative method for solving vehicle routing problems," in *Proc. 8th Int. Conf. Learn. Represent. (ICLR)*, Addis Ababa, Ethiopia, 2020, pp. 1–15.

[19] W. Nai, Z. Yang, D. Lin, D. Li, and Y. Xing, "A vehicle path planning algorithm based on mixed policy gradient actor-critic model with random escape term and filter optimization," *J. Math.*, vol. 2022, pp. 1–17, Aug. 2022.

[20] A. Kuefler, J. Morton, T. Wheeler, and M. Kochenderfer, "Imitating driver behavior with generative adversarial networks," in *Proc. IEEE Intell. Vehicles Symp. (IV)*, Los Angeles, CA, USA, Jun. 2017, pp. 204–211.

[21] R. P. Bhattacharyya, D. J. Phillips, B. Wulfe, J. Morton, A. Kuefler, and M. J. Kochenderfer, "Multi-agent imitation learning for driving simulation," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Madrid, Spain, Oct. 2018, pp. 1534–1539.

[22] G. Sukthankar and J. A. Rodrigues-Aguilar, *Autonomous Agents and Multiagent Systems* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2017, pp. 66–83.

[23] R. W. Byrne and A. E. Russon, "Learning by imitation: A hierarchical approach," *Behav. Brain Sci.*, vol. 21, no. 5, pp. 667–684, Oct. 1998.

[24] S. Schaal, "Is imitation learning the route to humanoid robots?" *Trends Cognit. Sci.*, vol. 3, no. 6, pp. 233–242, Jun. 1999.

[25] A. Bühler, A. Gaidon, A. Cramariuc, R. Ambrus, G. Rosman, and W. Burgard, "Driving through ghosts: Behavioral cloning with false positives," in *Proc. IEEE/RSJ Int. Conf. Intell. Robots Syst. (IROS)*, Oct. 2020, pp. 5431–5437.

[26] D. Chen, B. Zhou, V. Koltun, and P. Krähenbühl, "Learning by cheating," 2019, *arXiv:1912.12294*.

[27] F. Codevilla, M. Müller, A. López, V. Koltun, and A. Dosovitskiy, "End-to-end driving via conditional imitation learning," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 4693–4700.

[28] P. M. Kebria, A. Khosravi, S. M. Salaken, and S. Nahavandi, "Deep imitation learning for autonomous vehicles based on convolutional neural networks," *IEEE/CAA J. Autom. Sinica*, vol. 7, no. 1, pp. 82–95, Jan. 2020.

[29] W. Hu, X. Li, J. Hu, D. Kong, Y. Hu, Q. Xu, Y. Liu, X. Song, and X. Dong, "Deep imitation learning for autonomous vehicles based on convolutional neural networks," *IEEE Sensors J.*, vol. 23, no. 11, pp. 11285–11295, Mar. 2023.

[30] Y. Zhou, R. Fu, C. Wang, and R. Zhang, "Modeling car-following behaviors and driving styles with generative adversarial imitation learning," *Sensors*, vol. 20, no. 18, p. 5034, Sep. 2020.

[31] C. Finn, S. Levine, and P. Abbeel, "Guided cost learning: Deep inverse optimal control via policy optimization," in *Proc. 33rd Int. Conf. Mach. Learn.*, New York, NY, USA, 2016, pp. 49–58.

[32] R. Lioutikov, G. Neumann, G. Maeda, and J. Peters, "Learning movement primitive libraries through probabilistic segmentation," *Int. J. Robot. Res.*, vol. 36, no. 8, pp. 879–894, Jul. 2017.

[33] T. Osa, A. M. G. Esfahani, R. Stolkin, R. Lioutikov, J. Peters, and G. Neumann, "Guiding trajectory optimization by demonstrated distributions," *IEEE Robot. Autom. Lett.*, vol. 2, no. 2, pp. 819–826, Apr. 2017.

[34] T. Osa, N. Sugita, and M. Mitsuishi, "Online trajectory planning in dynamic environments for surgical task automation," in *Proc. Robot., Sci. Syst. Conf.*, Berkeley, CA, USA, Jul. 2014, pp. 1–9.

[35] T. Osa, N. Sugita, and M. Mitsuishi, "Online trajectory planning and force control for automation of surgical tasks," *IEEE Trans. Autom. Sci. Eng.*, vol. 15, no. 2, pp. 675–691, Apr. 2018.

[36] T. Zhang, Z. McCarthy, O. Jow, D. Lee, X. Chen, K. Goldberg, and P. Abbeel, "Deep imitation learning for complex manipulation tasks from virtual reality teleoperation," in *Proc. IEEE Int. Conf. Robot. Autom. (ICRA)*, May 2018, pp. 5628–5635.

[37] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. van den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, S. Dieleman, D. Grewe, J. Nham, N. Kalchbrenner, I. Sutskever, T. Lillicrap, M. Leach, K. Kavukcuoglu, T. Graepel, and D. Hassabis, "Mastering the game of go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, Jan. 2016.

[38] J.-C. Shi, Y. Yu, Q. Da, S.-Y. Chen, and A.-X. Zeng, "Virtual-Taobao: Virtualizing real-world online retail environment for reinforcement learning," in *Proc. 33rd AAAI Conf. Artif. Intell.*, Honolulu, HI, USA, 2019, pp. 4902–4909.

[39] W. Shang, Y. Yu, Q. Li, Z. Qin, Y. Meng, and J. Ye, "Environment reconstruction with hidden confounders for reinforcement learning based recommendation," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, Anchorage, AK, USA, Jul. 2019.

[40] Y. Aytar, T. Pfaff, D. Budden, T. Paine, Z. Wang, and N. de Freitas, "Playing hard exploration games by watch YouTube," in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2018, pp. 2935–2945.

[41] A. Edward, H. Sahni, Y. Schroecker, and C. Isbell, "Imitating latent policies from observation," in *Proc. 36th Int. Conf. Mach. Learn.*, Long Beach, CA, USA, 2019, pp. 1–9.

[42] T. Hester, M. Vecerik, O. Pietquin, M. Lanctot, T. Schaul, B. Piot, D. Horgan, J. Quan, A. Sendonaris, G. Dulac-Arnold, I. Osband, J. Agapiou, J. Z. Leibo, and A. Gruslys, ''Deep Q-learning from demonstrations,'' 2017, *arXiv:1704.03732*.

[43] S. Ross and D. Bagnell, ''Efficient reductions for imitation learning,'' in *Proc. 13th Int. Conf. Artif. Intell. Statist.*, Sardinia, Italy, 2010, pp. 661–668.

[44] A. Hussein, E. Elyan, M. M. Gaber, and C. Jayne, ''Deep imitation learning for 3D navigation tasks,'' *Neural Comput. Appl.*, vol. 29, no. 7, pp. 389–404, Apr. 2018.

[45] Z. Shou, X. Di, J. Ye, H. Zhu, H. Zhang, and R. Hampshire, ''Optimal passenger-seeking policies on E-hailing platforms using Markov decision process and imitation learning,'' *Transp. Res. C, Emerg. Technol.*, vol. 111, pp. 91–113, Feb. 2020.

[46] Z. Wang, J. Merel, S. Reed, G. Wayne, N. de Freitas, and N. Heess, ''Robust imitation of diverse behaviours,'' in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 1–15.

[47] E. Z. Liu, M. Hashemi, K. Swersky, P. Ranganathan, and J. Ahn, ''An imitation learning approach for cache replacement,'' in *Proc. 37th Int. Conf. Mach. Learn.*, 2020, pp. 1–11.

[48] S. Liu, G. Tian, X. Shao, and S. Liu, ''Behavior cloning-based robot active object detection with automatically generated data and revision method,'' *IEEE Trans. Robot.*, vol. 39, no. 1, pp. 665–680, Feb. 2023.

[49] L. Wang, C. Fernandez, and C. Stiller, ''High-level decision making for automated highway driving via behavior cloning,'' *IEEE Trans. Intell. Vehicles*, vol. 8, no. 1, pp. 923–935, Jan. 2023.

[50] Y. Li, Y. Zhu, F. Yang, and Q. Jia, ''Inverse reinforcement learning based optimal schedule generation approach for carrier aircraft on fight deck,'' *J. Nat. Univ. Defense Technol.*, vol. 35, no. 4, pp. 171–175, 2013.

[51] Z.-J. Jin, H. Qian, S.-Y. Chen, and M.-L. Zhu, ''Survey of apprenticeship learning based on reward function learning,'' *CAAI Trans. Intell. Syst.*, vol. 4, no. 3, pp. 208–212, 2009.

[52] S. Ross, G. J. Gordon, and J. A. Bagnell, ''A reduction of imitation learning and structured prediction to no-regret online learning,'' in *Proc. 14th Int. Conf. Artif. Intell. Statist.*, Fort Lauderdale, FL, USA, 2011, pp. 627–635.

[53] N. Rajaraman, L. Yang, J. Jiao, and K. Ramchandran, ''Toward the fundamental limits of imitation learning,'' in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2020, pp. 1–11.

[54] P. Abbeel and A. Y. Ng, ''Apprenticeship learning via inverse reinforcement learning,'' in *Proc. 21st Int. Conf. Mach. Learn.*, New York, NY, USA, 2004, pp. 1–8.

[55] U. Syed and R. E. Schapire, ''A game-theoretic approach to apprenticeship learning,'' in *Proc. 20th Int. Conf. Neural Inf. Process. Syst.*, Vancouver, BC, Canada, 2007, pp. 1449–1456.

[56] J. Merel, Y. Tassa, S. Srinivasan, J. Lemmon, Z. Wang, G. Wayne, and N. Heess, ''Learning human behaviors from motion capture by adversarial imitation,'' 2017, *arXiv:1707.02201*.

[57] X. Geng and B. H. Kang, *Trends in Artificial Intelligence* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2018, pp. 321–334.

[58] Y. Li, J. Song, and S. Ermon, ''InfoGAIL: Interpretable imitation learning from visual demonstrations,'' in *Proc. 30th Int. Conf. Neural Inf. Process. Syst.*, Long Beach, CA, USA, 2017, pp. 3815–3825.

[59] B. C. Stadie, P. Abbeel, and I. Sutskever, ''Third-person imitation learning,'' in *Proc. Int. Conf. Learn. Represent.*, Toulon, France, 2017, pp. 1–16.

[60] F. Torabi, G. Warnell, and P. Stone, ''Generative adversarial imitation from observation,'' 2018, *arXiv:1807.06158*.

[61] J. Song, H. Ren, D. Sadigh, and S. Ermon, ''Multi-agent generative adversarial imitation learning,'' in *Proc. 31th Int. Conf. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2018, pp. 7472–7483.

[62] T. Xu, Z. Li, and Y. Yu, ''Nearly minimax optimal adversarial imitation learning with known and unknown transitions,'' 2021, *arXiv:2106.10424*.

[63] A. Agarwal, S. M. Kakade, J. D. Lee, and G. Mahajan, ''Optimality and approximation with policy gradient methods in Markov decision processes,'' in *Proc. 33rd Annu. Conf. Learn. Theory (COLT)*, Graz, Austria, 2020, pp. 64–66.

[64] J. Ho and S. Ermon, ''Generative adversarial imitation learning,'' in *Proc. 29th Int. Conf. Neural Inf. Process. Syst.*, Barcelona, Spain, 2016, pp. 1–9.

[65] N. Rajaraman, Y. Han, L. F. Yang, K. Ramchandran, and J. Jiao, ''Provably breaking the quadratic error compounding barrier in imitation learning, optimally,'' 2021, *arXiv:2102.12948*.

[66] T. Xu, Z. Li, Y. Yu, and Z.-Q. Luo, ''On generalization of adversarial imitation learning and beyond,'' 2021, *arXiv:2106.10424*.

[67] B. Kryzhanovsky, W. Dunin-Barkowski, V. Redko, and Y. Tiumentsev, *Advances in Neural Computation, Machine Learning, and Cognitive Research* (Studies in Computational Intelligence). Cham, Switzerland: Springer, 2020, pp. 167–174.

[68] D. Winkler, S. Biffl, D. Mendez, and M. Wimmer, *Software Quality: Future Perspectives on Software Engineering Quality* (Lecture Notes in Business Information Processing). Cham, Switzerland: Springer, 2021, pp. 54–65.

[69] P. Kuderov and A. Panov, ''Planning with hierarchical temporal memory for deterministic Markov decision problem,'' in *Proc. 13th Int. Conf. Agents Artif. Intell.*, 2021, pp. 1073–1081.

[70] I. A. Surazhevsky, V. A. Demin, A. I. Ilyasov, A. V. Emelyanov, K. E. Nikiruy, V. V. Rylkov, S. A. Shchanikov, I. A. Bordanov, S. A. Gerasimova, D. V. Guseinov, N. V. Malekhonova, D. A. Pavlov, A. I. BELOV, A. N. Mikhaylov, V. B. Kazantsev, D. Valenti, B. Spagnolo, and M. V. Kovalchuk, ''Noise-assisted persistence and recovery of memory state in a memristive spiking neuromorphic network,'' *Chaos, Solitons Fractals*, vol. 146, May 2021, Art. no. 110890.

[71] S. Ilyuhin, A. Sheshkus, and V. L. Arlazarov, ''Recognition of images of Korean characters using embedded networks,'' in *Proc. 12th Int. Conf. Mach. Vis. (ICMV)*, Amsterdam, The Netherland, Jan. 2020, pp. 1–7.

[72] V. A. Demin, D. V. Nekhaev, I. A. Surazhevsky, K. E. Nikiruy, A. V. Emelyanov, S. N. Nikolaev, V. V. Rylkov, and M. V. Kovalchuk, ''Necessary conditions for STDP-based pattern recognition learning in a memristive spiking neural network,'' *Neural Netw.*, vol. 134, pp. 64–75, Feb. 2021.

[73] B. Goertzel, A. Panov, A. Potapov, and R. Yampolskiy, *Artificial General Intelligence* (Lecture Notes in Computer Science). Cham, Switzerland: Springer, 2020, pp. 172–182.

[74] A. Skrynnik, A. Staroverov, E. Aitygulov, K. Aksenov, V. Davydov, and A. I. Panov, ''Forgetful experience replay in hierarchical reinforcement learning from expert demonstrations,'' *Knowl.-Based Syst.*, vol. 218, Apr. 2021, Art. no. 106844.

[75] V. V. Kniaz, S. Y. Zheltov, F. Remondino, V. A. Knyaz, A. Bordodymov, and A. Gruen, ''Wire structure image-based 3D reconstruction aided by deep learning,'' *Int. Arch. Photogramm., Remote Sens. Spatial Inf. Sci.*, vol. 2020, pp. 435–441, Aug. 2020.

[76] R. Sun, ''Optimization for deep learning: Theory and algorithms,'' 2019, *arXiv:1912.08957*.

[77] Y. Nesterov, ''A method of solving a convex programming problem with convergence rate $O(1/k^2)$,'' *Sov. Math. Doklady*, vol. 27, pp. 372–376, Jan. 1983.

[78] Y. Nesterov, ''On an approach to the construction of optimal methods of minimization of smooth convex functions,'' *Matematicheskie Metody Resheniya Ekonomicheskikh Zadach*, vol. 24, pp. 509–517, Jan. 1988.

[79] Y. Nesterov, ''Smooth minimization of non-smooth functions,'' *Math. Program.*, vol. 103, no. 1, pp. 127–152, May 2005.

[80] Y. Nesterov, ''Gradient methods for minimizing composite functions,'' *Math. Program.*, vol. 140, no. 1, pp. 125–161, Aug. 2013.

[81] A. Beck and M. Teboulle, ''A fast iterative shrinkage-thresholding algorithm for linear inverse problems,'' *SIAM J. Imag. Sci.*, vol. 2, no. 1, pp. 183–202, Jan. 2009.

[82] A. Rohde and A. B. Tsybakov, ''Estimation of high-dimensional low-rank matrices,'' *Ann. Statist.*, vol. 39, no. 2, pp. 887–930, Apr. 2011.

[83] V. Koltchinskii, K. Lounici, and A. B. Tsybakov, ''Nuclear-norm penalization and optimal rates for noisy low-rank matrix completion,'' *Ann. Statist.*, vol. 39, no. 5, pp. 2302–2329, Oct. 2011.

[84] S. Negahban and M. J. Wainwright, ''Restricted strong convexity and weighted matrix completion: Optimal bounds with noise,'' 2010, *arXiv:1009.2118*.

[85] P. Jain, P. Netrapalli, and S. Sanghavi, ''Low-rank matrix completion using alternating minimization,'' in *Proc. 45th Annu. ACM Symp. Theory Comput.*, Palo Alto, CA, USA, Jun. 2013, pp. 665–674.

[86] M. Hardt and M. Wootters, ''Fast matrix completion without the condition number,'' in *Proc. 27th Conf. Learn. Theory*, Barcelona, Spain, 2014, pp. 638–678.

[87] T. Zhao, Z. Wang, and H. Liu, ''A nonconvex optimization framework for low rank matrix estimation,'' in *Proc. 28th Int. Conf. Neural Inf. Process. Syst.*, Montreal, QC, Canada, 2015, pp. 1–9.

[88] R. Sun and Z.-Q. Luo, ''Guaranteed matrix completion via non-convex factorization,'' *IEEE Trans. Inf. Theory*, vol. 62, no. 11, pp. 6535–6579, Nov. 2016.

[89] Q. Zheng and J. Lafferty, ''Convergence analysis for rectangular matrix completion using Burer–Monteiro factorization and gradient descent,'' 2016, *arXiv:1605.07051*.

[90] Y. Guan, J. Duan, S. Eben Li, J. Li, J. Chen, and B. Cheng, ''Mixed policy gradient: Off-policy reinforcement learning driven jointly by data and model,'' 2021, *arXiv:2102.11513*.

[91] R. Garnett, *Bayesian Optimization*. Cambridge, U.K.: Cambridge Univ. Press, 2023, pp. 87–200.

[92] E. D. Dolan and J. J. Moré, "Benchmarking optimization software with performance profiles," *Math. Program.*, vol. 91, no. 2, pp. 201–213, Jan. 2002.

[93] D. Li and D. Zhu, "An affine scaling interior trust-region method combining with line search filter technique for optimization subject to bounds on variables," *Numer. Algorithms*, vol. 77, no. 4, pp. 1159–1182, Apr. 2018.

ZAN YANG was born in Jilin, China, in 1984. He received the B.S. degree in information and computing science from Dalian Nationalities University, Dalian, Liaoning, China, in 2008, the M.S. degree in basic mathematics from Beihua University, Jilin, in 2014, and the Ph.D. degree in basic mathematics from Jilin University, Jilin, in 2017.

From 2008 to 2011, he was a Hardware Engineer with Headquarter of Datang Telecom, and a Lecturer and the Secretary of the General Communist Party Branch, Deputy of Science, Tongji Zhejiang College, from 2017 to 2022. Since 2022, he has been a Lecturer with the Public Teaching and Research Department, Huzhou College. He is the author of more than 80 articles. His research interests include algorithms on machine learning, reinforcement learning, and their applications.

Dr. Yang is a Lifetime Member of Chinese Association for Artificial Intelligence (CAAI) and the Operations Research Society of China (ORSC).

WEI NAI (Member, IEEE) was born in Nanjing, Jiangsu, China, in 1985. He received the B.S. degree in optical information science and technology and the M.S. degree in electromagnetic and microwave technology from Beijing University of Posts and Telecommunications (BUPT), Beijing, China, in 2007 and 2010, respectively, and the Ph.D. degree in transportation information engineering and control from Tongji University, Shanghai, China, in 2013.

From 2014 to 2016, he was a Postdoctoral Researcher with the Postdoctoral Research Station of Control Science and Engineering, Tongji University; and a Lecturer, an Associate Professor, the Principal of Sino-German exchanges, and the Vice Dean of the Department of Electronic and Information Engineering, Tongji Zhejiang College, from 2017 to 2022. Since 2022, he has been an Associate Professor with the School of Electronic Information, Huzhou College. He is the author of one book and over 130 articles. His current research interests include applied broadband communication and transportation behavior in information environment based on artificial intelligence algorithms.

Dr. Nai is a Lifetime Member of Chinese Association for Artificial Intelligence (CAAI). He is one of the "Leading Talents of Colleges and Universities (The Youth Talents)" in Zhejiang, the first batch of "Top-Notch Youth Talents in Education", the "Outstanding Talents", the "Excellent Teachers", and the first batch of "Innovation and Entrepreneurship Advisors" in Jiaxing.

DAN LI was born in Tonghua, Jilin, China, in 1988. She received the B.S. degree in mathematics and applied mathematics and the M.S. degree in basic mathematics from Beihua University, Jilin, in 2011 and 2014, respectively, and the Ph.D. degree in operational research and cybernetics from Shanghai Normal University, Shanghai, China, in 2017.

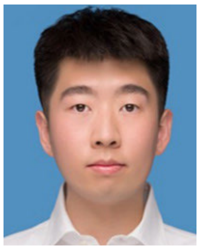
From 2017 to 2022, she was a Lecturer with the Deputy of Science, Tongji Zhejiang College. Since 2022, she has been a Lecturer with the Public Teaching and Research Department, Huzhou College. She is the author of more than 30 articles. Her research interests include artificial intelligence related methodologies and their applications, convex, and non-convex optimization theories.

Dr. Li is a Lifetime Member of Chinese Association for Artificial Intelligence (CAAI) and the Operations Research Society of China (ORSC). She is one of the "Leading Talents of Colleges and the Universities (The Youth Talents)" in Zhejiang.

LU LIU was born in Tieling, Liaoning, China, in 1988. He received the B.S. and M.S. degrees in transportation engineering from Tongji University, Shanghai, China, in 2009 and 2012, respectively, the M.S. degree in transportation engineering from Oregon State University, USA, in 2014, and the Ph.D. degree in marketing from Syracuse University, USA, 2020.

From 2019 to 2020, he was a Visiting Assistant Professor with Saginaw Valley State University. Since 2021, he has been an Assistant Professor of marketing with St. Bonaventure University. He is the author of ten articles. His research interests include statistical modeling, public policy, healthcare, and econometrics.

Dr. Liu is a member of American Marketing Association (AMA).



ZIYU CHEN was born in Jilin, China, in 1993. He received the Ph.D. degree in optics from Nankai University, Tianjin, China, in 2022.

Since 2022, he has been a Lecturer with the School of Electronic Information, Huzhou College. His research interests include applied broadband communication, rare earth luminescence, and flexible optoelectronics.