

Received 13 March 2025, accepted 2 April 2025, date of publication 15 April 2025, date of current version 23 April 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3560988

RESEARCH ARTICLE

A Design Framework for Scalable and Adaptive Multi-Agent Coordination in Dynamic Environments: Addressing Concurrent Agent and Environment Interactions

RAZA MUHAMMAD KAZIM¹, GUOXIN WANG^{1,2}, ZHENJUN MING^{1,2}, JINHUI CAO¹, JANET K. ALLEN³, (Member, IEEE), AND FARROKH MISTREE³

¹School of Mechanical Engineering, Beijing Institute of Technology, Beijing 100811, China

²Yangtze Delta Region Academy, Beijing Institute of Technology, Jiaxing, Zhejiang 314019, China

³Systems Realization Laboratory, The University of Oklahoma, Norman, OK 73019, USA

Corresponding author: Zhenjun Ming (zhenjun.ming@bit.edu.cn)

Zhenjun Ming acknowledges support from the National Natural Science Foundation of China (Grant No. 62373047) and the Beijing Municipal Science and Technology Foundation (Grant No. 3222020). Guoxin Wang acknowledges support from the National Natural Science Foundation of China (Grant No. 51975056). Janet K. Allen acknowledges funding from the John and Mary Moore Chair. Farrokh Mistree acknowledges funding from the L.A. Comp Chair at the University of Oklahoma.

ABSTRACT In dynamic environments, such as box-pushing tasks, multi-agent systems (MAS) face significant challenges in coordinating agents within high-density settings while managing uncertainties arising from fluctuations in agent configurations and environmental dynamics. In this study, we explore the integration of surrogate response surface modeling (SRSM) with optimization algorithms, comparing Stochastic Gradient Descent (SGD) with a fixed learning rate, and adaptive learning rate (ALR) optimizers—including Adaptive Moment Estimation (ADAM) and Adaptive Approximate Direction Method Algorithm (AADMA)—to enhance MAS performance metrics, such as Agent Collision Rate (ACR), Agent Movement Frequency (AMF), and Task Completion Time (TCT). Through systematic experimentation across five scenarios, SRSM is employed to uncover key trends in MAS performance and identify configurations that improve scalability and adaptability. From the analysis of simulation data, it has been observed that SGD struggles significantly in dynamic environments, while ADAM demonstrates moderate improvements. However, AADMA consistently outperforms both by reducing loss, lowering collision rates, increasing movement efficiency, and achieving shorter task completion times. Performance comparison charts and loss function graphs emphasize AADMA's superiority in addressing the complexities of real-time coordination and adaptability. Through this study, we highlight the critical role of combining SRSM with ARL to design an MAS that is capable of thriving in complex, dynamic, and high-density environments. By addressing key scalability and adaptability challenges, the proposed framework significantly advances MAS design, paving the way for improved multi-agent coordination in real-world applications.

INDEX TERMS Multi-agent reinforcement learning, adaptive learning rate, box-pushing task, multi-agent system, dynamic environments, scalability, adaptability.

LIST OF ABBREVIATIONS

ACR	Agent Collision Rate.
AADMA	Adaptive Approximate Direction Method Algorithm.
ADAM	Adaptive Moment Estimation.
ALR	Adaptive Learning Rate.

The associate editor coordinating the review of this manuscript and approving it for publication was Qiang Li¹.

AMF	Agent Movement Frequency.
MADDPG	Multi-agent Deep Deterministic Policy Gradient.
MADQN	Multi-agent Deep Q-Network.
MARL	Multi-agent Reinforcement Learning.
SGD	Stochastic Gradient Descent.
SRSM	Surrogate Response Surface Modeling.
TCT	Task Completion Time.

I. FRAME OF REFERENCE

A multi-agent system (MAS) is a set of agents that work together to achieve a common goal. This feature makes MAS particularly suitable for solving complex tasks that require coordination, communication, and decision-making among distributed agents [1]. The decentralized nature of MAS is particularly beneficial in such domains as robotics, autonomous vehicles, distributed sensing, and supply chain, where the cooperation of agents is critical [2]. For instance, in warehouses, robots can coordinate with each other to move and deliver products and simultaneously plan the best possible routes to avoid any obstacles. Likewise, in disaster response scenarios, MAS can also efficiently organize search and rescue missions in uncertain terrain without using the command-and-control hierarchy [3].

The overall performance of MAS depends on the scalability, stability, and adaptability of the agents, as well as on their capacity to respond to varying task requirements for the development of emergent behavior. As shown in Fig. 1, Multi-agent system emergent behavior. The phenomenon of MAS results from the local interactions of agents trying to solve a common problem without having a central controller [4]. Control processes allow the system to change and manage itself in response to external and internal conditions and improve performance over time [5]. This adaptive capability promotes emergent behaviors, which have been used in real-world problems such as the management of traffic flow.

One of the most important applications in the context of MAS research is the box-pushing task in which several agents have the common goal of moving a box from a starting position to a target position. This task is all about collaboration and synchronization, which are crucial for its success [6]. It serves as a benchmark in MAS research to examine the behaviors of agents as well as to design algorithms that enhance cooperation. The agents must synchronize their actions, for example, force and position, to slide the box and adjust to the changing environment, the position of the box, and other objects [7].

The box-pushing task is intrinsically cooperative and coordinated. If one agent cannot move a box on its own, it has to coordinate with other adjacent agents to apply the forces appropriately for constant cooperation. This task underscores the importance of algorithms that support emergent coordination, self-organization, and distributed decision-making [8]. Problems occur when agents act in dynamic environments

with stochastic and constantly changing obstacles that require the agent to mandate adaptations of real-time strategies. Significance researchers show that agents must be able to respond to such environmental changes to ensure that the system can be resilient and flexible. Nanjanath and Gini [9] point out that agents must respond to dynamic environments, thereby testing their robustness and adaptability. Recent approaches used for the box pushing employ machine learning and reinforcement learning that allows agents to optimize their behaviors through trial and error. Multi-agent reinforcement learning (MARL) is essential in the optimization of the agent's strategies, which in turn, enhances the box-pushing cooperation [10]. Kommey offers basic information on how to implement MARL to cooperative tasks like box pushing where agents learn cooperation strategies on their own [11].

Important aspects of the box-pushing task are the fact that agents are required to work in a coordinated manner to accomplish the task. Forces and positioning, in particular, have to be coordinated by agents to ensure their effectiveness. These issues have been well addressed by the recent development of multi-agent reinforcement learning (MARL), where agents learn from the outcomes of their actions. When it comes to box-pushing tasks, MARL methods like the MADDPG and MADQN have shown enhanced coordination approaches [12]. However, these approaches still face many challenges, especially concerning changes in the agent configurations and the environment, which, most of the time, require a lot of training. Recognizing this limitation, in our previous work [13], we proposed a framework leveraging surrogate modeling with MADDPG and design-of-experiments (DOE) techniques to address challenges related to agent configuration in multi-agent systems (MAS). While effective in optimizing agent-related variables, this framework neglected dynamic environmental factors, limiting its applicability to real-world scenarios with complex agent-environment interactions.

Building upon this foundation, in this study, we propose an enhanced framework that integrates adaptive learning rates (ALR) with surrogate response surface modeling (SRSM) to address critical challenges in multi-agent systems (MAS). This framework overcomes the limitations of previous methods by explicitly incorporating dynamic environmental factors, enabling MAS to adapt to non-linear interactions and uncertainties in both agent and environmental configurations. By addressing these complexities, the framework significantly enhances the applicability, scalability, and performance of MAS in dynamic, high-density environments. This integrated approach offers several direct benefits in tackling key challenges in MAS:

1) **Enhancing Adaptability:** Dynamic environments involve unpredictable variations in obstacle configurations, agent arrangements, and target reallocations, which require real-time adjustments [11]. The ALR-SRSM combination enables agents to quickly adapt to these dynamic shifts, improving their responsiveness to unforeseen changes and

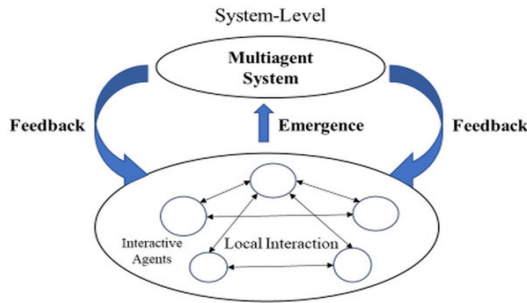


FIGURE 1. Multi-agent system emergent behavior [16].

ensuring seamless interactions during tasks such as box pushing.

2) **Improving Coordination:** In a dynamic environment, the risk of collisions increases as multiple agents operate simultaneously [14]. The ALR-SRSM framework facilitates both individual learning and emergent coordination, aligning agent behaviors toward collective goals and supporting synchronized actions in complex, multi-agent scenarios.

3) **Scalability:** As the number of agents increases, the complexity of coordination increases exponentially [15]. The ALR-SRSM framework promotes scalability by enabling systems to implement complex strategies without the need for retraining, making it ideal for tasks involving large agent groups and evolving configurations.

Through this research, we aim to address the core question: How can scalable multi-agent systems be designed to adapt rapidly to changes in agent configurations and dynamic environmental conditions while maintaining emergent behavior in cooperative tasks? By integrating ALR and SRSM, this study endeavors to fill the gaps that existing MARL methodologies leave, offering viable solutions to critical challenges in dynamic and cooperative multi-agent systems.

One of the feasible methods to address this challenge is using machine learning algorithms to train a system (with a specific configuration) so that the agents develop a strategy to collaboratively perform a task in the environment [17]. There are many algorithms to do this, such as MADDPG, MADQN, etc. Although these algorithms are useful for designers to explore the desirable system behavior in a given setting, they have clear limitations, such as being restricted to specific agent configurations and environmental settings [18]. When the agent configuration or the environment changes, the performance of the system declines, and the agents need to be retrained to adapt to a new environment. In our earlier work [19], we proposed a surrogate model-based method to tackle the issue of agent configuration variations and identification of the best agent configuration for a given task. The key idea is two-fold: The first is to train the multi-agent using MADDPG, then vary the agent configurations using DOE (Design of Experiments) techniques to generate sample inputs, and fit the response surfaces using surrogate model techniques to facilitate designers in identifying the best input that can produce the optimal performance. In this

study, we take a step further by addressing situations where variations occur in both agent configurations and environment settings. We develop an interactive design exploration and adaptive learning strategy to identify the key input design variables that lead to the most effective performance outcomes. The question that we address in this study is, “How can a scalable multi-agent system be designed to enable rapid adaptation to both agent configuration changes and dynamic environment variables while ensuring emergent behavior during the execution of cooperative box-pushing tasks in complex environments?”

We organize this paper as follows. In Section II, we present the related work consisting of the foundations of multi-agent systems and the design of the box-pushing multi-agent system, with research gaps and contributions. In Section III, we present the system architecture and design method for the development of our design framework consisting of the system architecture, integration of MADDPG with adaptive learning rate, and simulation environment. In Section IV, we present the experimental setup and the interpretation of the results. In Section V, we summarize our contributions through discussion along with closing remarks in Section VI.

II. RELATED WORK

The foundations of machine learning in box-pushing multi-agent systems provide critical insights into their coordination, scalability, and adaptability in dynamic environments. This section examines the design of these systems, focusing on the impact of surrogate modeling in achieving scalability and adaptability. Additionally, we discuss the role of adaptive learning rates (ARL) in enhancing multi-agent reinforcement learning (MARL) and address challenges related to scalability and adaptability in high-density, dynamic scenarios. Finally, a comparative analysis of prior work on modern MARL techniques highlights their strengths and limitations and how our approach advances the field.

A. MACHINE LEARNING IN BOX-PUSHING MULTI-AGENT SYSTEM

The Multi-Agent System (MAS) has been extensively studied in collaborative tasks, such as box-pushing, where agents must work together to move an object from one location to another. In these controlled environments, Multi-Agent Reinforcement Learning (MARL), particularly algorithms like MADDPG (Multi-Agent Deep Deterministic Policy Gradient), have successfully trained agents to adapt to dynamic tasks and collaborate towards a common goal [17].

However, while these studies demonstrate the potential of MARL in controlled environments, similar approaches have also been successfully deployed in real-world systems. For example, robot swarms used in search and rescue missions require agents to collaborate effectively and adapt to changing environments, just like in box-pushing tasks [18]. Similarly, fleets of autonomous vehicles in smart cities have implemented MAS frameworks to optimize traffic flow and vehicle coordination [19]. These systems utilize

reinforcement learning algorithms to handle dynamic and uncertain environments, much like how agents must adapt to box-pushing tasks. Multi-agent systems also face other challenges in practical applications, such as scalability to hundreds or thousands of agents and robustness to noisy sensors or unexpected obstacles. Nonetheless, these challenges highlight the practical feasibility of applying MAS to larger and more complex environments, such as dynamic environments in box-pushing tasks, where agents must quickly and effectively adapt to new conditions.

One of the basic algorithms in MARL, the Multi-Agent Deep Deterministic Policy Gradient (MADDPG), uses centralized training for decision-making with decentralized implementation, which is useful for cooperative tasks [17]. In this framework, a centralized critic network evaluates the joint action-value function, $Q(o, a)$, based on the observations and actions of all agents. Every agent produces actions based on their local observations; this makes the system work in real time and also helps in scalability and adaptability. While MADDPG performs well in small- to medium-scale multi-agent scenarios, it faces scalability issues when the number of agents increases and struggles with adaptability in dynamically changing environments. Although it remains effective for cooperative task execution, these limitations highlight the need for advancements to overcome these challenges [20].

The overview of the Multi-agent Deep Deterministic Policy Gradient (MADDPG) algorithm illustrated in Fig. 2 is a centralized training-decentralized execution scheme [21]. In this framework, each agent generates actions using its actor-network based on local observations, while a centralized critic network assesses the joint action-value function $Q(o, a)$, which considers the joint observations $o = [o_1, o_2, \dots, o_N]$ and joint actions $a = [a_1, a_2, \dots, a_N]$ of all agents. The critic updates its parameters using the temporal difference (TD) error, which derives the gradient to evaluate how an agent's action influences the total Q-value. Each actor then applies the chain rule to update its policy parameters based on the feedback. This structure facilitates coordination during training, as the critic has access to all agents' observations and actions. However, during the execution phase, agents act autonomously, making decisions based solely on their local information, which enhances the system's scalability and flexibility. MADDPG proves to be particularly effective in addressing challenges such as coordination, fault tolerance, and dynamic adaptation, making it well-suited for applications in swarm robotics and distributed systems.

Recent MARL algorithms, such as QMIX and MAPPO, have been developed to address the scalability and adaptability limitations of classic algorithms like MADDPG. QMIX, in particular, employs the factorization of the monotonic value function to efficiently compute the joint action value functions for large populations of agents [22]. Although it excels at synchronizing actions within complex environments, it performs poorly when agents must act

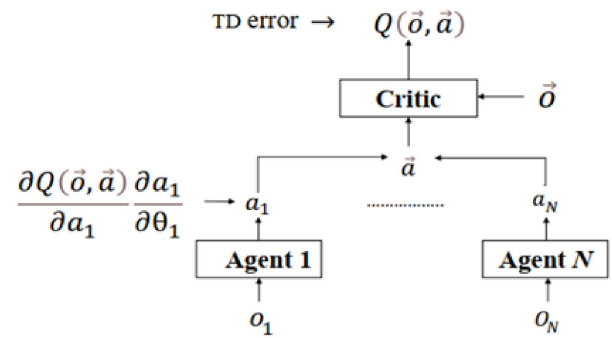


FIGURE 2. MADDPG [27].

independently or when the environment changes constantly. Multi-agent Proximal Policy Optimization (MAPPO), derived from Proximal Policy Optimization (PPO), offers effective policy optimization methods that balance exploitation and exploration, making it more adaptable to non-stationary environments [23]. However, MAPPO can become unstable when the number of agents and task demands is high and requires significant computational power [24].

Moreover, current MARL algorithms continue to face challenges in handling simultaneous agent-environment uncertainties, where both agents' actions and environmental states are stochastic and dynamic [25], [26]. In this study, we integrate surrogate response surface modeling (SRS) and adaptive learning rates (ALR) into the MARL framework. This combination enables agents to predict environmental changes and adapt in real time, reducing the need for retraining. Additionally, integrating ALR enhances the learning process, maintaining efficiency as the number of agents increases. In addition to capturing the dependencies between agents' actions and environmental states, SRS helps agents manage non-stationary situations. This integrated approach enhances MARL frameworks, making them more suitable for real-world applications and improving their scalability for complex, dynamic environments such as swarm robotics and distributed systems.

B. DESIGN OF A BOX-PUSHING MULTI-AGENT SYSTEM

A box-pushing multi-agent system (MAS) design focuses on enabling multiple independent agents to coordinate their actions to move a box from an initial position to a desired final position. This task highlights critical challenges such as coordination, cooperation, and the ability to operate in a constantly changing environment. The system leverages MARL to teach agents optimal strategies through trial and error, utilizing algorithms like MADDPG or its improved versions to enhance cooperation and decision-making [28]. Some key design features include equipping agents with sensors and actuators to perceive and interact with their environment, avoid obstacles, and communicate with other agents [29]. Coordination mechanisms are often incorporated

to facilitate effective communication and improve group performance.

Furthermore, the dynamic nature of the system, such as moving obstacles or shifting box positions, requires adaptability mechanisms, including real-time adjustments to learning rates and policy updates [30]. Simulation environments are essential for testing and refining the system before real-world implementation. They allow for estimating performance parameters such as success rates, coordination efficiency, and goal attainment by simulating various scenarios and challenges [31]. In conclusion, designing a box-pushing multi-agent system requires integrating agent architecture, decision-making methods, adaptability techniques, and simulation software to create a robust and efficient system. This approach captures the dynamics of cooperation among multiple agents and paves the way for advancements in algorithms and methodologies applicable to real-world scenarios.

C. SURROGATE MODELING

Surrogate modeling has emerged as a powerful tool in optimization tasks due to its ability to replace computationally expensive models with cost-effective approximations [32]. These models emulate the input-output mapping of high-fidelity simulations and are particularly valuable when design space exploration is constrained by time or resource limitations. Traditional methods, such as Response Surface Modeling (RSM), have been widely used in engineering and optimization for similar purposes. In surrogate modeling, RSM employs polynomial approximations to capture the relationships between design variables and performance metrics while accounting for computational efficiency. However, traditional models often struggle in complex multi-agent environments where agent interactions, adaptability, and flexibility are critical. They tend to assume constant parameters, making them ineffective when agents need to adapt or learn new behaviors dynamically [33].

Previous studies have explored surrogate models in Multi-Agent Reinforcement Learning (MARL) to enhance decision-making by adjusting agent behavior based on predicted outcomes [34]. However, most existing surrogate-based MARL methods are either limited in scope, tailored to specific scenarios, or lack the flexibility to address emerging uncertainties introduced by agents and dynamic environments [35]. This study addresses these gaps by integrating surrogate models with adaptive learning techniques, enabling agents to predict and adapt their policies in real time [36]. This integrated mechanism enhances the agent's ability to respond to environmental changes, improves the scalability of the learning process, and ensures that MARL systems remain both effective and efficient in dynamic and complex environments [37].

Current optimization and modeling techniques have contributed significantly to design optimization and performance evaluation. In Table 1, we summarize a comparison of

traditional modeling techniques for multi-agent system design, including brute-force simulations, grid search, and surrogate-based models like Kriging and Radial Basis Functions (RBFs). While these methods are useful, they have notable limitations. Brute-force simulations are computationally expensive and impractical for large design spaces due to exponential growth in resource demands. Grid search struggles with scalability in high-dimensional problems, and surrogate models often lack interpretability and generalizability, making their predictions difficult to replicate across diverse contexts.

The surrogate response surface modeling (SRS) approach presents significant advantages over these traditional methodologies. It enhances efficiency by requiring fewer simulations to explore the design space accurately, leveraging advanced sampling techniques like the central composite design analysis. SRS accurately captures complex, non-linear relationships between input parameters and performance metrics. For instance, studies have shown its predictive reliability to be highly robust, with $R^2 > 95$ percent. Furthermore, the polynomial models used in SRS are computationally lightweight and interpretable, facilitating their seamless integration into real-world applications. These features make SRS a practical and effective tool for improving decision-making processes and optimizing complex systems.

D. ADAPTIVE LEARNING RATE (ALR) IN MARL

The adaptive learning rate (ALR) is crucial for training reinforcement learning (RL) models, as it adjusts the learning rate based on the optimization landscape [42]. This adaptability is particularly relevant in multi-agent reinforcement learning (MARL), where agent-environment interactions introduce significant challenges like coordination, non-stationarity, and scalability issues. By incorporating ALR, MARL systems gain the flexibility to adapt to dynamic conditions and improve overall performance [43].

Various ALR methods have been proposed to increase the stability and efficiency of RL training.

- **ADAM:** This method combines RMSProp and AdaGrad, calculating the learning rate for each parameter using the first and second moments of the gradients [44]. Adam is widely used due to its efficiency and significant improvements in convergence speed and stability compared to Stochastic Gradient Descent (SGD) [45].

- **ADAMA:** Specifically designed for multi-agent settings, this method assigns each agent its learning rate based on its contribution to the overall value estimate. This approach enables fine-grained learning, improving adaptability and performance in multi-agent environments [46].

Additional techniques, such as RMSProp and AdaGrad, are also utilized in various RL architectures. RMSProp stabilizes the learning rate by using a moving average of squared gradients, while AdaGrad adapts learning rates based on the magnitude of past gradients. However, AdaGrad faces

TABLE 1. Comparison of traditional modeling techniques for multi-agent system design.

Methodology	Advantages	Limitations	SRSM Advantages Over Alternatives
Brute-Force Simulations [38]	- Exhaustively evaluates all configurations.	- Extremely high computational cost. - Impractical for large-scale or high-dimensional problems.	- SRSM reduces computational costs by requiring fewer simulations. - Efficient sampling (e.g., CCD analysis).
Grid Search [39]	- Systematic exploration of hyperparameters.	- Scalability issues in high-dimensional design spaces. - Computationally expensive.	- SRSM efficiently explores the design space with fewer iterations. - Handles non-linear relationships effectively.
Kriging Models [40]	- Excellent for interpolation. - Handles complex relationships.	- Lack interpretability. - Computational overhead increases with larger datasets.	- SRSM uses computationally lightweight polynomial models. - Provides better interpretability.
Radial Basis Functions (RBFs) [41]	- Suitable for function approximation and optimization.	- Generalization issues in dynamic and diverse environments. - Limited scalability for high-dimensional problems.	- SRSM delivers reliable predictions with high accuracy ($R^2 > 95\%$). - Flexible for integration with adaptive learning.
Traditional RSM (Response Surface Modeling)	- Captures relationships between variables and performance metrics. - Reduces computational burden compared to brute-force methods.	- Struggles with complex, non-linear interactions. - Assumes constant parameters, limiting adaptability.	- SRSM captures complex, non-linear relationships between variables. - Highly adaptable to dynamic environments.

challenges due to excessive decay over time, which can hinder long-term training [47].

Integrating ALR with multi-agent reinforcement learning (MARL) will enhance system adaptability by dynamically adjusting learning rates based on feedback from the environment and other agents. This approach is well-suited to the variability of multi-agent systems, as it enables agents to learn efficiently while maintaining stability in the learning process. The proposed framework is also expected to outperform conventional approaches by fine-tuning for diverse scenarios with minimal retraining. By addressing issues related to agent interactions, ALR improves both the quality and stability of training, representing a significant advancement in the field of MARL.

E. SCALABILITY AND ADAPTABILITY

Scalability and adaptability remain significant challenges in MARL, particularly in large-scale systems involving many agents and dynamic environments. Initiatives like SMAC and SMAC2 have made progress in enhancing scalability, enabling the management of tens or even hundreds of agents. However, extending the number of agents to thousands remains challenging due to computational complexity and fluctuating policies [48]. Adaptive learning rates (ALR) address these challenges by dynamically adjusting the learning rate based on the number of agents and their level of interaction. This approach ensures that the update process remains efficient as the system scales, preventing the degradation of learning caused by increased agent interactions. The integration of MARL with ALR in distributed and decentralized systems demonstrates that these issues can be mitigated, enabling agents to operate smoothly in the

presence of obstacles while cooperating to achieve common goals [49].

Adaptability, which focuses on equipping agents to handle dynamic environments, has advanced through hierarchical reinforcement learning and meta-learning methods. However, these approaches are often ineffective in managing simultaneous uncertainties and extrapolating to new situations [50]. Adaptive learning rates (ALR) address these limitations by employing hyperparameters that dynamically adjust the learning rate based on environmental changes, thereby increasing system flexibility. Momentum facilitates more informed adjustments, resulting in smoother and faster convergence. Meanwhile, epsilon ensures that the adaptation process remains stable and avoids disruption due to numerical instabilities during adaptive changes [51].

To address the challenges of scalability and adaptability in Multi-Agent Systems (MAS), the proposed framework integrates the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm with adaptive learning rate (ALR) techniques. ALR dynamically adjusts learning rates, enhancing flexibility in dynamic environments while reducing computational complexity, thus improving scalability. This integration effectively balances adaptability and scalability, making the framework suitable for large-scale, dynamic, and high-density MAS applications.

Our previous study [13] developed a framework combining cDSP, MADDPG, and Design of Experiments (DOE) techniques, focusing exclusively on agent-related design variables. While effective in static environments, it lacked consideration of dynamic environmental factors, limiting its relevance for complex real-world agent-environment interactions. The current framework addresses these gaps

TABLE 2. Comparative analysis of MADDPG with ALR against modern reinforcement learning techniques.

Criteria	MADDPG with ALR	QMIX	Proximal Policy Optimization (PPO)	Model-Based Methods (e.g., MuZero)
Stability and Convergence	Enhanced stability through ALR; effectively addresses scalability challenges in dynamic environments [52].	Exhibits high stability and convergence in cooperative tasks; scalability supported through Q-value decomposition [22].	Offers impressive stability due to clipping and adaptive strategies, but complex in multi-agent scenarios [53].	Highly stable; adapts using modeled predictions to enhance convergence and learning efficiency across diverse tasks [54].
Learning Efficiency	Adaptive learning rates contribute to effective learning, but may lag in sample efficiency [55].	Efficient in learning from joint actions; communication enhances scalability across agents [56].	High sample efficiency with adaptive exploration, though stressed by added complexity [57].	Superior sample efficiency; utilizes environmental models to enhance learning from limited interactions [58].
Coordination and Communication	Centralized training fosters agent coordination, improving scalability in multi-agent systems [59].	Excellent coordination through joint value optimization; strong in cooperative setups [60].	Good coordination, particularly with shared information; challenges in decentralized contexts [61].	Dynamic communication capabilities; enhances coordination through learned interactions within environments [62].
Generalization and Robustness	Robust adaptability to varying conditions, though may struggle with generalization in unseen tasks [63].	Strong generalization across various environments; excels in dealing with non-stationarity [64].	Generally good at generalization but may falter in highly competitive or complex settings [65].	Highly robust; designed for adaptability, allowing effective decision-making in diverse and unforeseen circumstances [66].
Limitations	May experience performance degradation in competitive environments; overall complexity can hinder scalability [63].	Reliant on cooperative frameworks; may encounter limitations in independent settings [67].	Complexity in implementation can be high; suffers from instability in extreme conditions [68].	Computationally intensive; demands significant resource allocation for building accurate environmental models [69].
Applications	Well-suited for cooperative multi-agent tasks in robotics and simulations, especially where adaptability is crucial.	Applied in cooperative strategic games and scenarios requiring joint value assessment.	Used extensively across various domains including robotics, game AI, and financial modeling due to its robust adaptability.	Effective in complex game environments (e.g., Go, Chess) and dynamic scenarios where agents must adapt quickly.

by incorporating ALR and Surrogate Response Surface Modeling (SRSM), enabling MAS to adapt effectively to non-linear interactions and uncertainties, significantly enhancing scalability and adaptability.

In Table 2, we present a comparative analysis of MADDPG with ALR against modern reinforcement learning techniques, including QMIX, PPO, and model-based methods such as MuZero. QMIX shows scalability in cooperative tasks but struggles with independent agents. PPO adapts well to dynamic environments but faces challenges in large-scale scenarios due to its reliance on centralized training. Model-based approaches, such as MuZero, excel in robustness and sample efficiency but are hindered by high computational costs and limited scalability. MADDPG with ALR builds upon these methodologies by employing dynamic learning rates to handle high-density environments, relocating targets, and moving obstacles more effectively. While effective, further refinements are necessary to optimize performance in ultra-large-scale and highly competitive settings.

The proposed framework, which combines MADDPG with ALR and SRSM, represents a significant advancement over prior approaches. SRSM enhances computational efficiency by effectively sampling the design space, achieving high

prediction accuracy ($R > 95$ percent) for non-linear relationships between agent parameters and performance metrics. Meanwhile, ALR techniques like ADAM and AADMA boost real-time adaptability, enabling MAS to manage relocating targets, moving obstacles, and high agent densities with improved efficiency. Performance metrics, including Agent Collision Rate (ACR), Agent Movement Frequency (AMF), and Task Completion Time (TCT), consistently show improvement under this framework. The proposed MADDPG framework with ALR and SRSM provides a robust solution for medium- to large-scale MAS applications in dynamic and cooperative environments. With continued development, it holds strong potential to address challenges in ultra-large, competitive, and diverse real-world scenarios, paving the way for advancements in practical MAS design.

III. SYSTEM ARCHITECTURE AND DESIGN

The design of a box-pushing multi-agent system demands adaptability and scalability, particularly in complex environments. This system consists of multiple autonomous agents working collaboratively toward a common goal. Multi-agent systems are essential for solving large distributed problems by leveraging agents' interactions. These systems

are trained using reinforcement learning to improve the collective efficiency of the agents. Additionally, surrogate modeling handles diverse scenarios, enhancing the system's stability and versatility. Adaptive learning mechanisms are further integrated into the multi-agent reinforcement learning framework to improve system performance and its applicability to real-world problems.

A. OVERALL FRAMEWORK FOR DESIGN OF MULTI-AGENT SYSTEM

The proposed framework effectively integrates MADDPG with adaptive learning rates (ALR) and surrogate response surface modeling (SRSM) to address scalability and adaptability challenges in multi-agent systems (MAS) operating in dynamic environments. It leverages agent-specific parameters (e.g., number of agents, size, velocity) and environment-specific parameters (e.g., dynamic obstacles, target positions) to manage simultaneous uncertainties arising from agent interactions and environmental changes. The overall framework for scalable and adaptive multi-agent system design is illustrated in Fig. 3

At the framework's core, MADDPG enables centralized training with decentralized execution, allowing agents to coordinate through a centralized critic during training. The critic evaluates the value of joint actions using the observations and actions of all agents. During execution, the decentralized structure empowers agents to act autonomously based solely on local feedback. While MADDPG performs well in cooperative tasks, it faces inherent limitations, including low scalability as the number of agents increases and difficulty adapting to dynamic environments. The integration of SRSM and ALR addresses these challenges.

The Surrogate Response Surface Model (SRSM) is important in estimating the relationship between agents, environmental variables, and performance metrics such as collision rate, movement frequency, and task completion time. By reducing the need for extensive simulations during training, SRSM significantly lowers computational costs. Furthermore, SRSM enables agents to predict the effects of their actions in real time, facilitating dynamic policy updates without retraining—an especially valuable capability in unstable or non-stationary conditions.

The framework incorporates advanced adaptive learning rate (ALR) optimizers to enhance adaptability and scalability. ALR dynamically adjusts the learning process based on environment and agent models feedback, ensuring efficient learning under diverse conditions. By tailoring learning rates to specific interactions between agents and their environment, ALR improves training stability and effectively scales the framework to handle larger agent populations and more complex environments. This adaptation mechanism enables agents to respond quickly to environmental changes and evolving agent behaviors, ensuring robust performance in dynamic scenarios.

The combination of MADDPG, SRSM, and ALR effectively solves the critical challenges of scalability and

adaptability in MAS. Unlike previous MARL methods, such as QMIX and MAPPO, which primarily address agent-environment conflict problems, this framework enables agents to adapt their plans while maintaining operational flexibility dynamically. As a result, the proposed framework is well optimized for real-world multi-agent applications, including swarm robotics and distributed systems. It not only overcomes the limitations of existing methods but also serves as a powerful and scalable tool for designing multi-agent systems in complex and dynamic environments.

The framework comprises three interrelated components that collectively address scalability, resilience, and adaptability in dynamic environments. Each stage builds upon the previous one, employing key techniques such as MADDPG, network optimization, and adaptive learning rate optimization. Fig. 4 illustrates the system-level diagram of the multi-agent system design framework, highlighting how these components interact to optimize agent behavior and enhance system performance across diverse scenarios. This design supports efficient detection, real-time updates, and reliable task executions.

To evaluate the individual contributions of the SRSM and ALR optimizations to MADDPG performance, we conducted experiments that independently isolated and assessed the effects of each optimization technique. Specifically, the experiments were designed to evaluate the following configurations:

1. MADDPG with SGD (baseline): This configuration is the baseline, with no additional optimizations applied. It establishes the performance without the influence of SRSM or ALR.
2. MADDPG + SRSM: This configuration applies Surrogate Response Surface Modeling (SRSM) to enhance learning efficiency and agent performance, building on the baseline results from MADDPG with SGD.
3. MADDPG + ALR (ADAM, AADMA): In this configuration, we incorporate adaptive learning rate optimizations (ADAM and AADMA) to assess their effects on training stability, convergence speed, and overall system performance.

By evaluating these configurations independently, we effectively isolate the contributions of SRSM and ALR optimizations, providing a detailed understanding of each technique's impact on MADDPG performance before integrating them into the final framework. This ensures that performance improvements are attributed to individual optimization techniques.

Step 1: Baseline Analysis with MADDPG + SGD

In the first step, the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) algorithm is analyzed using Stochastic Gradient Descent. This step focuses on understanding agent behaviors based on agent-related variables (e.g., the number, size, and velocity of agents) and environmental variables (e.g., varying obstacle configurations and target reallocations). The initial performance of the multi-agent system is evaluated through output performance metrics such as Agent Collision Rate (ACR), Agent Movement Frequency

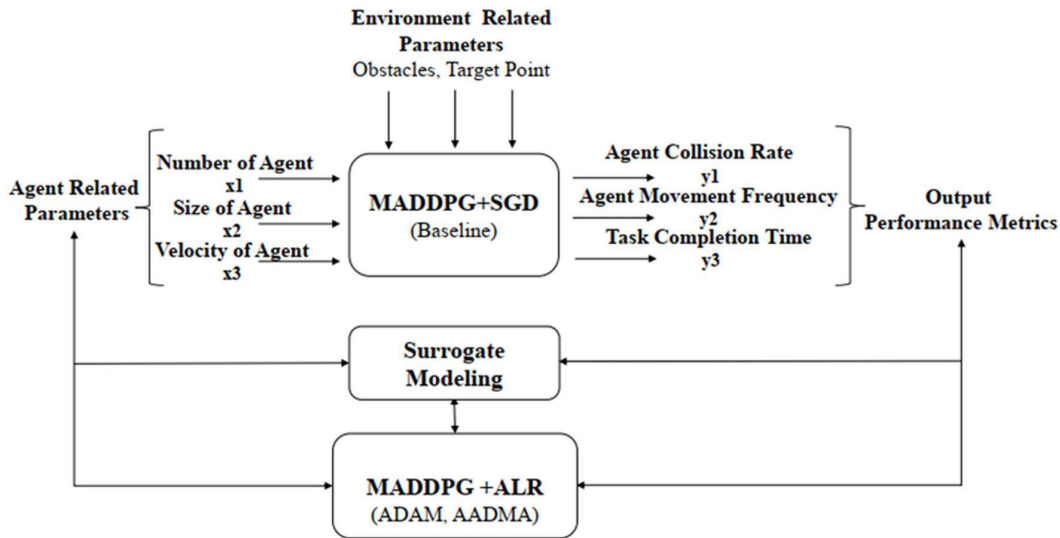


FIGURE 3. Overall framework for scalable and adaptive multi-agent system design.

(AMF), and Task Completion Time (TCT). These metrics establish a performance baseline and provide a consistent framework for analyzing and improving agent behavior in scenarios requiring continuous cooperation and adaptability.

Step 2: Surrogate Response Surface Modeling

The second step introduces surrogate response surface modeling (SRS) to estimate the complex interactions between agents and their environment. This approach leverages the results of Step 1 to simulate different agent configurations, optimizing agent performance while enhancing flexibility and scalability. Surrogate modeling reduces computational demands while allowing for better adaptability to dynamic environments.

Step 3: Integration of MADDPG with ADAM and AADMA

In the final step, we explored MADDPG with adaptive learning rate (ALR) optimizers, specifically ADAM and AADMA, guided by the insights gained from the surrogate model. This step includes developing an integrated design and learning framework, referred to as “IDEAL,” which dynamically adjusts learning rates based on environmental conditions or agent interactions. This integration enhances the system’s ability to adapt to changing environments, resulting in a highly efficient and scalable multi-agent system.

1) PROBLEM DEFINITION AND GOALS

The box-pushing task involves multiple agents collaborating to push a box toward a shifting target position. These agents must navigate confined spaces, avoid becoming stuck, and ensure the box reaches its destination. This task requires agents to self-organize, coordinate actions, and adapt in real time as the environment and associated constraints develop. Designing a multi-agent push-box system for dynamic environments involves several critical steps to address the

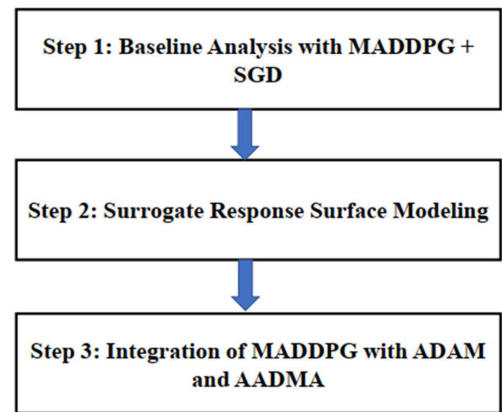


FIGURE 4. System-level diagram of the multi-agent system design framework.

complexity and variability of authentic scenarios effectively. First, the system must be flexible and capable of handling varying agent configurations and environmental conditions without significant performance degradation. It should adapt to changes in agent characteristics, such as number, size, and velocity, as well as environmental variables like obstacle configurations and target position [31].

Additionally, the system must respond in real time to dynamic constraints, moving obstacles, and shifting agent configurations. Agents should act autonomously, making decisions based on environmental information without relying on centralized control. This decentralized approach enhances adaptability and robustness, enabling each agent to adapt to its surroundings and coordinate effectively with others. Such a design ensures robust performance in complex, dynamic environments.

We formulate three output performance metrics to evaluate the effectiveness of the proposed multi-agent system design framework.

- **Agent Collision Rate (ACR):** This metric measures the number of collisions between agents and obstacles or between agents themselves. Reducing the collision rate reflects better coordination and more efficient movement, essential for improving system performance and preventing congestion.

- **Agent Movement Frequency (AMF):** This metric evaluates the agents' ability to navigate the environment while minimizing unnecessary movement. The objective is to ensure agents follow the most efficient routes with minimal repetitive operations, thereby enhancing overall system planning and operational efficiency.

- **Task Completion Time (TCT):** This metric assesses the time for agents to move the box to the specified target location. A shorter completion time indicates improved collaboration between agents and their ability to adapt to changing environmental conditions.

These metrics—ACR, AMF, and TCT—serve as the primary criteria for evaluating system performance and guiding the design and improvement of the multi-agent scheme.

2) AGENT AND ENVIRONMENT-RELATED PARAMETERS CONFIGURATIONS

Multi-agent tasks are designed to manage and evaluate the interactions between agents and their environments. This approach focuses on how agents collaborate to achieve a shared goal, such as moving a box to a target in a dynamic environment. Agents must coordinate their efforts to apply force collectively on an object (the box). Effective collaboration minimizes task completion time and reduces unnecessary movements. Each agent makes decisions based on its observations of the environment, the target position, and other agents' actions. The proposed framework is designed to handle many scenarios, from simple static environments to complex, dynamic environments with high variability and evolving structures. System flexibility was evaluated by varying both agent configurations and environmental conditions.

Agent-related Variables:

- **Number of Agents:** Varying the number of agents helps evaluate how changes in group size impact planning efficiency and task completion.
- **Size of Agents:** Testing agents of different sizes helps determine how physical dimensions affect movement and collision rates.
- **Velocity of Agents:** Optimizing agent speed helps evaluate the trade-off between faster movement and accuracy in avoiding obstacles.

Environment-related variables:

- **Number and size of obstacles:** Varying the number and size of obstacles changes the task's difficulty by increasing or decreasing collision risks.
- **Obstacle Orientation:** Adjusting the orientation of obstacles introduces varying constraints on agent movement, ranging from fixed to dynamic configurations.

- **Target Reallocation:** Changing the target's position evaluates how hard-to-reach locations impact task completion times and agent behavior.

3) MULTI-AGENT SIMULATION ENVIRONMENT

A 2D simulation environment is developed in Python to evaluate multi-agent tasks, including obstacle avoidance, coordination, and target interactions. The framework simulates realistic dynamics, focusing on collision rates, task completion times, and system efficiency while allowing flexibility to test various configurations. It provides insights into agent behavior and adaptability in dynamic environments, aiding the refinement of algorithms for complex scenarios. Designed to simulate real-world conditions, the environment enables agents to navigate obstacles and coordinate their movements toward a shared goal. This framework supports the analysis of agent behavior, evaluates obstacle-avoidance mechanisms, and identifies areas for system improvement, ensuring flexibility in testing diverse approaches.

As shown in Fig. 5, Scenario 1 represents a static environment where obstacles (depicted in green) are scaled to 10 units. Agents are expected to navigate this environment without colliding with static objects. The 'Adversary (Box),' represented in blue, is also scaled to 10 units, introducing an additional challenge that requires agents to coordinate their movements while avoiding obstacles. The target point, also in gray, is scaled to 10 units. This target creates a dynamic interaction as agents move the box toward it, testing their efficiency and cooperation. This setup highlights the agents' ability to minimize collisions, reduce unnecessary movements, and optimize task completion times. The simulation results are critical for understanding how multi-agent systems perform under controlled conditions and for refining algorithms and frameworks to address more complex and dynamic scenarios in the future.

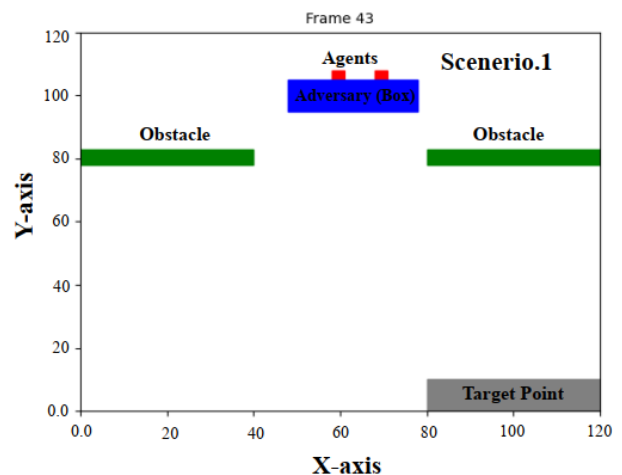


FIGURE 5. Multi-agent particle simulation environment.

To evaluate the design variables of the multi-agent box-pushing system, we investigated five distinct scenarios, each

TABLE 3. Scenarios key configuration.

Scenario	Description	Key Configurations	Expected Outcomes
1	Limited Number of Obstacles	- Number of obstacles = 2 - Size of obstacles = 10 - Random obstacle placement - Fixed target position	Agents focus on basic navigation with minimal obstacle interference.
2	Increasing the Number of Obstacles and Adjusting Their Position	- Same as Scenario 1 - Number of obstacles = 3	Tests agent adaptability to increasingly cluttered environments.
3	Increasing the Size of Obstacles	- Same as Scenario 1 - Size of obstacles = 13	Assesses agents' ability to adjust strategies for larger obstacles.
4	Changing the Position of the Target Point	- Same as Scenario 1 - Target relocates	test agents' ability to adapt to a moving target.
5	Moving Obstacles	- Same as Scenario 1 - Obstacles move dynamically	Evaluates agents' adaptability to moving obstacles in real-time.

with unique configurations. As shown in Table 3, these scenarios represent both common and challenging conditions to test system performance:

- Scenario 1 serves as the baseline, featuring four randomly placed obstacles in a relatively predictable environment.
- Scenario 2: The number of obstacles was increased to six, creating a denser environment with long and nonlinear paths, testing agents' ability to navigate congestion.
- Scenario 3: The Obstacle size was increased, requiring precise movements and challenging the system's ability to handle physical constraints effectively.
- Scenario 4: A dynamic mechanism was introduced, continuously changing obstacle positions during the simulation, requiring agents to make real-time adjustments to their plans.
- Scenario 5: Moving obstacles were added, introducing environmental variations and testing agents' ability to adapt to constantly shifting obstacle positions.

These scenarios collectively provide a comprehensive framework for assessing multi-agent systems' flexibility, adaptability, and robustness under increasingly complex and dynamic conditions.

To evaluate different configurations of a multi-agent system, we investigated five distinct scenarios, each with unique challenges. The training parameters for the system are outlined in Table 4. Key parameters include 32 mini-batches per learning phase to enhance performance, a maximum replay buffer of 2000 for learning trials, and a learning rate of 0.001 to manage increasing task difficulty. A discount rate of 0.95 was used to prioritize long-term rewards. The training consists of 20,000 steps, representing a complete simulation. An, μ -greedy behavior selection algorithm was implemented, beginning with an exploration rate of 1.0, which decays to 0.005 over 100,000 steps, facilitating the transition to learned strategies.

Simulations were conducted in Python 3.12 using PyTorch-based libraries for multi-agent detection platforms (MADDPG). NumPy handled computational tasks, while Matplotlib visualized regression data. The environment, built using an Open AI Gym, featured dynamic challenges and obstacles. The stochastic gradient descent (SGD) optimizer

TABLE 4. Simulation parameters.

Simulation Parameter	Description
Mini-Batches per Training Step	32
Replay Buffer Size	2000
Base Learning Rate	0.001
Discount Factor	0.95
Total Training Episodes	20,000
Action Selection Approach	e-Greedy
Initial Exploration Rate (ϵ)	1.0
Final Exploration Rate (ϵ)	0.005
Annealing Duration	100,000 steps

and adaptive learning rate (ALR) mechanism balanced neural network weights to enhance performance, ensuring stability and faster convergence. An NVIDIA GPU supported over 20,000 training iterations, enabling dynamic adaptation to varying environments.

Cooperative agents (highlighted in green) are trained using a combination of MADDPG with adaptive learning rate (ARL) optimizers like SGD, ADAM, and AADMA to interact effectively with both cooperative and adversarial agents (the box). These agents coordinate their movements to push the adversary, surround it, and apply force from multiple directions to drive it toward the goal. As the adversary attempts to escape or counter the push, the green agents dynamically adjust their strategies, reposition themselves, or shift their focus to maintain control. ALR enables agents to optimize their force application plans, blocking the opponent's escape paths while pushing them in the desired direction. The green agents continuously refine their strategies in real time, adapting to changes in opponent behavior and environmental conditions. This balance between exploration (testing new strategies) and exploitation (applying the most effective tactics) is critical for success. Agents begin training with a random policy, which is refined over time through iterative training sessions based on performance improvements. This adaptive training process enhances system efficiency, ensuring the agents' strategies

become increasingly effective in dynamic and competitive scenarios.

B. STEP 1: BASELINE ANALYSIS WITH MADDPG + SGD

In this section, we combine the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) and Stochastic Gradient Descent (SGD) optimizer to train our multi-agent box-pushing system. MADDPG is a reinforcement learning method specifically designed for multi-agent environments, offering a robust framework for handling complex planning tasks. Fig. 2 illustrates the actor and critic network architecture of MADDPG. Although MADDPG provides a flexible approach for learning and adapting rules in multi-agent systems, SGD analyzes regulations and trains key networks with stable learning rates [28]. This combination supports the design of space exploration strategies when evaluating system performance under controlled conditions.

Although SGD with fixed learning rates ensures stable performance, it lacks the flexibility of adaptive training methods. As such, this approach serves as a baseline for evaluating the impact of adaptive learning rates on scalability and performance. By starting with SGD, we establish a foundation for exploring the benefits of variable learning rates in improving system robustness and adaptability.

This framework outlines a clear path from initial design to implementation, addressing real-time transitions and system deployment challenges. Developers can evaluate the performance of MADDPG with SGD using metrics such as learning rate to identify potential areas for improvement. Additionally, this evaluation highlights the effectiveness of existing policies and strategies in planning and executing tasks within a multi-agent box-pushing environment. Ultimately, this analysis provides valuable insights into how user goals and environmental factors influence the learning process, supporting adaptive model development and analysis

C. STEP 2: IMPLEMENTATION OF SURROGATE MODELING FOR OPTIMIZATION

The Design of Experiments (DOE) methodology addresses the complexity of multivariate problems by optimizing the interactions between variables, thus reducing cost and time while improving reliability. Surrogate models, such as polynomial regression, approximate the relationships between variables and system responses, enabling rapid evaluations and iterative optimizations in multi-agent systems (MAS) [70]. DOE's reputation stems from its efficiency and reliability in solving complex problems involving numerous variables and tests. This approach establishes a comprehensive design strategy that significantly reduces development costs and time while enhancing the reliability and adaptability of system designs.

In this context, surrogate models are integrated with the Multi-Agent Deep Deterministic Policy Gradient (MADDPG) to analyze user behavior and system performance in dynamic environments. The central composite design

analysis approximates planning functions by fitting a quadratic space with minimal experimental trials. This approach facilitates efficient parameter space exploration, reduces computational costs, and supports the development of robust MAS designs suitable for authentic applications [71].

Integrating DOE with surrogate models and the central composite design method allows for efficient exploration of the parameter space and rapid iteration of system configurations. By adopting this approach, we can optimize agent interactions, minimize the need for computationally expensive simulations, and achieve a robust MAS design that can effectively adapt to dynamic real-world environments. Surrogate models reduce the number of trials required. At the same time, DOE ensures the identification and optimization of the most critical parameters, ultimately leading to significant cost savings, improved time efficiency, and enhanced system performance.

D. STEP 3: ADVANCED OPTIMIZATION WITH MADDPG + ALR (ADAM AND AADMA)

To enhance the multi-agent reinforcement learning (MARL) framework, we integrate two adaptive learning rate (ALR) optimizers, specifically Adaptive Moment Estimation (ADAM) and Adaptive Approximate Direction Method (AADMA). ADAM optimizes the learning rate using first- and second-order moment estimates of the gradient, improving recovery in mixed or clustered environments. AADMA extends ADAM by incorporating acceleration and learning algorithms that dynamically adjust parameters based on gradient differences and processing conditions, enabling more precise updates [72].

The mathematical framework of the ALR algorithm both for ADAM and AADMA is illustrated in Fig. 6. In this formulation, O and t represent the initial and updated learning rates at step t , determining the magnitude of parameter updates. The parameters 1 and 2 are typically set at 0.9 and 0.999, respectively, to indicate the decay rates for the first and second moments. The constant μ prevents division by zero, while t denotes the current time step of the optimization process. These mechanisms ensure that ADAM can effectively optimize the system, especially when gradient behavior changes at different stages of training. AADMA extends the functionality of ADAM by incorporating advanced features such as acceleration and dynamic learning rate adjustment. AADMA responds to gradient differences and environmental conditions, enabling more precise updates by adjusting the learning and decay rates in real time. This allows for more effective adaptation to varying training conditions and improved optimization in dynamic settings. In the ALR framework, the gradient gt represents the loss function's acceleration at timestep t , while mt and vt are the first- and second-order moment estimates, respectively. These bias-corrected estimates provide more stable gradient updates. The variable yt captures the difference between

```

Step 0: Set  $k = 0$ . Given initial learning rate  $\mu = 0.001$ ,  $\beta_1, \beta_2, m_0$  and  $v_0$  are initialized to 0 and time step  $t = 0$ .
Step 1: Set  $t = t + 1$ 
Step 2: Update Biased first and second Moment Estimates
 $m_t = \beta_1 m_{t-1} + (1 - \beta_1) g_t$  (1)  $v_t = \beta_2 v_{t-1} + (1 - \beta_2) g_t^2$  (2)
Step 3: Compute Bias-Corrected First and Second Moment Estimates
 $\hat{m}_t = m_t / (1 - \beta_1)$   $\hat{v}_t = v_t / (1 - \beta_2)$  Here,  $\beta_1$  and  $\beta_2$  are the exponential decay rate for ADAM optimizer.
 $\hat{m}_t = \beta_{1,t} \cdot m_{t-1} + (1 - \beta_{1,t}) g_t$   $\hat{v}_t = \beta_{2,t} \cdot v_{t-1} + (1 - \beta_{2,t}) g_t^2$  Here,  $\beta_{1,t}, \beta_{2,t}$  are adaptive momentum decay rates for AadMa optimizer,  $\beta_{1,t} = \beta_1 (1 - \lambda_t)$ ,  $\beta_{2,t} = \beta_2 (1 - \lambda_t)$ ,  $\lambda_t = \max(|g_t|, |g_{t-1}|) |g_t - g_{t-1}|$ 
Step 4: Compute Differences  $y_t = \hat{g}_t - \hat{g}_{t-1}$  and  $s_t = \hat{g}_t - \hat{g}_{t-1}$ 
Step 5: Compute Bk, Scaling factor
 $B_{k+1}^{(1)} = \frac{1}{1 + \omega(\hat{y}_t^{(k)})}$  For ADAM  $B_k^{(1)} = \frac{1}{1 + \omega(\hat{y}_t^{(k)})}$  For AadMa
Step 6: Adjust Learning Rate if  $t = 1$ ,  $\mu_t = \mu_0$ , else  $\mu_t = \mu_0 B_t^{(1)}$  For ADAM,  $\mu_t = \mu_0 (1 - \eta_0)$  For AadMa
Step 7: Update Parameters
 $\theta_{t+1} = \theta_t - (\mu_t \times \hat{m}_t / \sqrt{\hat{v}_t + \epsilon})$  For ADAM,  $\theta_{t+1} = \theta_t - \alpha_t \cdot m_t / \sqrt{v_t + \epsilon + B_k \cdot s_t}$  For AadMa
Step 8: Increment Iteration Counter Set  $k = k + 1$  and go to Step 1

```

FIGURE 6. Mathematical formulation of MADDPG + ALR implementations [44], [46].

the current and previous gradients, while s_t represents the difference between the current and prior parameters. Additionally, B_k is a scalar or skew matrix that adjusts the learning rate based on parameter difference calculations. The parameters t and $t+1$ represent the model states at time steps t and $t+1$, respectively, reflecting updates in the optimization process.

Built on the foundation of ADAM, the AadMa processor introduces advanced features such as adaptive motion degradation and dynamic learning rate adjustment, enabling fine-grained control of new parameters. AadMa automatically adjusts the learning rate t in response to varying gradients and task conditions, ensuring stable updates even during sudden changes. Specifically, AadMa incorporates the variation t to dynamically adjust the decay rates $1, t$ and $2, t$. It also accounts for gradient variance (y_t) and parameter variance in its B_k component, enabling precise updates and transformations. These features significantly enhance stability and accuracy, particularly in rapidly changing environments [73]. These innovations make AADMA especially effective in rapidly changing environments where task conditions may fluctuate unpredictably, ensuring a more stable and responsive optimization process.

When integrated into the MADDPG framework, these optimizers enable real-time adaptation to dynamic environments, reducing the need for manual tuning and accelerating convergence. ADAM ensures reliable gradient-based updates, while AADMA improves performance by incorporating gradient and parameter variations into the optimization process. This integration enhances scalability and adaptability, addressing key challenges in multi-agent systems by reducing the number of required workers, improving movement efficiency, and decreasing task completion time. Compared to traditional MADDPG-SGD, this combination significantly enhances system performance, particularly in complex and dynamic environments, providing a more robust, adaptive, and efficient platform for multi-agent systems [74].

IV. EXPERIMENTS AND RESULTS

A series of experiments were conducted to evaluate the scalability and adaptability of a multi-agent system (MAS) in a simulation environment. The experimental setup involved agents (shown as green circles) pushing an adversary (shown by a red circle) toward a target while avoiding obstacles (represented as blue and grey circles). The results highlight the impact of various design variables on the flexibility and performance of the self-organizing MAS, demonstrating its ability to adapt rapidly to dynamic environments. Performance outcomes are presented through tables, comparison charts, and loss function graphs, providing a detailed analysis of the characteristics and effects of each variable.

A. EXPERIMENT SETUP

To address the gaps in scalability and adaptability in multi-agent systems (MAS), we developed a comprehensive experimental framework using a multi-agent simulation environment for a push-box task. Building on predefined scenarios, our experiments systematically evaluate system performance under varying environmental conditions and constraints. This framework enables an in-depth analysis of factors such as obstacle configurations and dynamic target switching and their effects on agent behavior and overall system adaptability. By leveraging these scenarios, the framework provides a robust foundation for assessing the scalability, flexibility, and robustness in complex, dynamic environments while offering valuable insights into optimizing MAS for real-world challenges.

The experimental work is conducted in three steps:

1. Step 1: Initial simulations were performed with varying agent-related parameters, number of agents (2, 4, 8, 16, 24, 32, 64, 100, and 120), agent sizes (1,4 and 9 unit), and agent speeds (0.5, 1.5, and 3.5 units). These simulations analyzed the physical properties of the system, with metrics such as Agent Collision Rate (ACR), Agent Movement Frequency (AMF), and Task Completion Time (TCT) recorded to evaluate agent-environment interactions under different conditions.

2. Step 2: Surrogate Response Surface Modeling (SRSM) was applied to visualize and evaluate design strategies. The objective was to identify optimal design configurations that enhance adaptability in complex environments.

3. Step 3: The simulations were rerun using the optimized parameters identified in Step 2, under the same conditions as in Step 1, to evaluate and verify the resilience and adaptability of the system frequently.

This comprehensive testing framework addresses scalability by evaluating system performance under challenging environmental constraints, such as higher obstacle density, larger obstacles, and varying agent configurations. It also tackles adaptability by assessing the system's ability to maintain high performance in dynamic environments where obstacles and target positions change in real time. By systematically analyzing output performance metrics (ACR, AMF, and TCT)

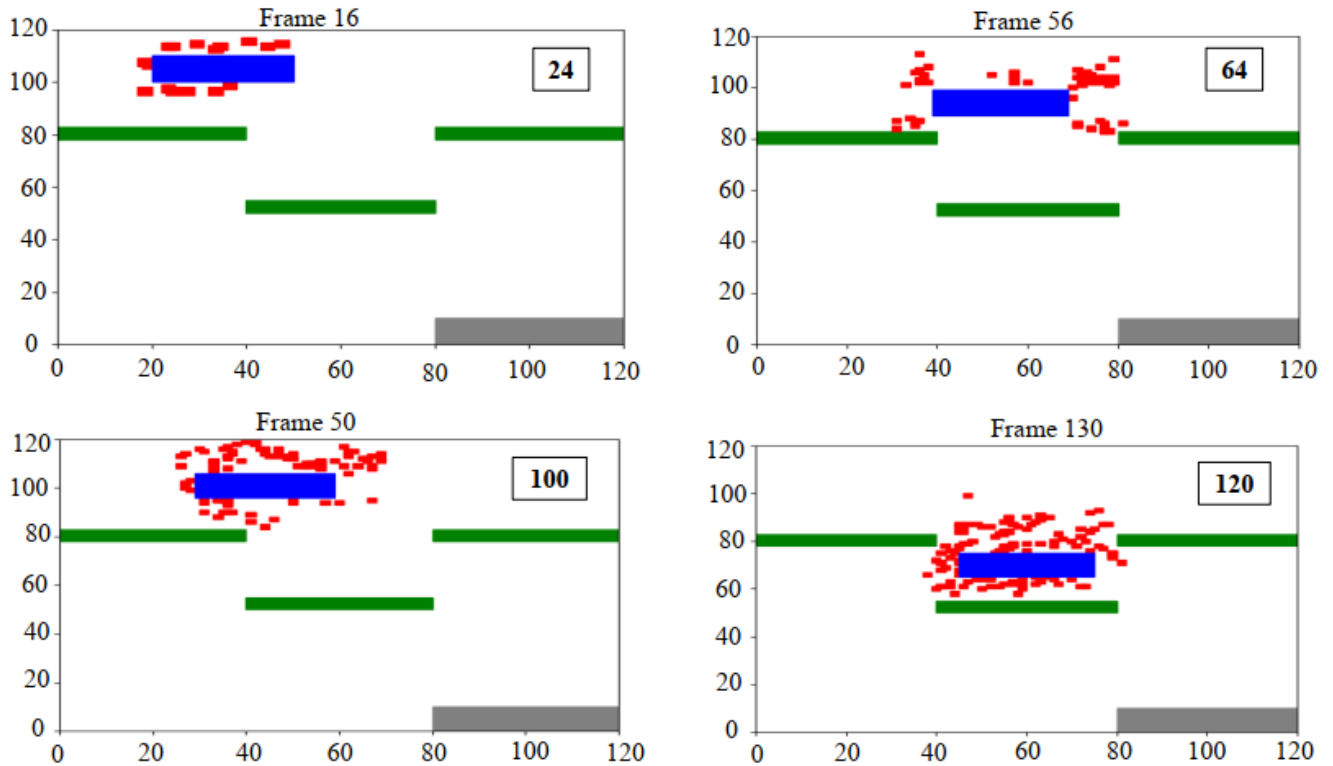


FIGURE 7. Agents interaction across different configurations including 24, 64, 100, and 120 agents.

across various scenarios—including novel dynamic setups—our approach demonstrates significant improvements in scalability and adaptability. Furthermore, the integration of dynamic features, such as moving obstacles, ensures that the framework remains robust and responsive to evolving real-world conditions.

B. RESULT

In this section, we analyze the performance metrics across multiple experimental scenarios, primarily focusing on Scenario 1, Scenario 4, and Scenario 5. These scenarios were designed to evaluate the impact of various environmental factors: Scenario 1 examines the influence of a limited number of obstacles, Scenario 4 explores the effects of target transition states, and Scenario 5 investigates the impact of moving obstacle positions. The analysis reveals how key design variables—such as agent's number, size, and velocity—affect output performance metrics. A central concern in our study was the scalability and adaptability of the system, particularly as the number of agents increases. As the number of agents grows, the interactions between them and their ability to adapt to the dynamic environment become more complex. To illustrate these challenges, Fig. 7 presents visual representations of agent interactions across configurations, including 24, 64, 100, and 120 agents. These visuals demonstrate how agent behavior and coordination evolve as the agent count increases, with the complexity of movement and coordination growing accordingly. This

highlights the scalability challenges in multi-agent systems as their size increases.

The results for each configuration are presented in Tables 5, 6, and 7, where the performance of MADDPG under different optimization techniques is compared:

- Table 5: Simulation results for scenarios 1, 4, and 5 (S1, S4, S5)-MADDPG with SGD serves as the baseline configuration, providing a reference for comparison. The performance metrics indicate that this configuration exhibited slower convergence or higher collision rates than optimized configurations.
- Table 6: Simulation results for scenarios 1, 4, and 5 (S1, S4, S5)-MADDPG with SRSM demonstrated significant improvements, notably in ACR and TCT, addressing scalability and adaptability challenges as the number of agents increased.
- Table 7: Simulation result for scenarios 1, 4, and 5 (S1, S4, S5)-MADDPG with ALR (ADAM and AADMA), led to improvements in faster convergence, better stability, and reduced collision rates, highlighting the critical role of adaptive learning in improving overall system performance.

The discussion of these results underscores how changes in key design variables influence the system's scalability, adaptability, and performance under varying environmental conditions.

In Step 1, we collected baseline performance metrics across various agent configurations. This initial step established a foundational framework for understanding the

trends in system performance, particularly about scalability and adaptability under a dynamic environment. In Step 2, we applied Surrogate Response Surface Modeling (SRSM) to evaluate the influence of different parameters on system behavior. This step generated regression equations that enabled us to model and adjust the system's performance, identifying effective design strategies that address key challenges in scalability and adaptability. It provided valuable insights into the relationship between multi-agent system (MAS) design choices and performance metrics. Finally, in Step 3, we validated the optimized design strategies through simulations, using the experimental setups from Step 1. The results showed significant performance improvements, including reduced collision rates, higher agent movement frequencies, and shorter task completion times. These findings confirm the effectiveness of the optimization strategies. Additionally, performance comparison charts and loss function graphs were presented to provide clear, quantitative representations of improvements in key performance metrics, demonstrating how the proposed framework enhances the scalability and adaptability of multi-agent systems.

1) STEP 1: BASELINE ANALYSIS WITH MADDPG + SGD

The experimental results for Step 1 evaluate MADDPG using a fixed learning rate with a Stochastic Gradient Descent (SGD) optimizer across Scenarios 1, 4, and 5 (S1, S4, and S5), as shown in Table 5.

- **Scenario 1 (Baseline: limited obstacles):** This scenario represents a relatively simple environment with a limited number of static obstacles and a fixed target. The ACR remains low, indicating that the agents can effectively coordinate their actions with minimal collisions in an environment with few obstacles and no dynamic elements. AMF is high because the agents make fewer adjustments and can freely navigate to the target without interference. TCT is short, as agents can reach the target quickly, especially when there are fewer agents and moderate or high speeds. As the number of agents or their speed increases, ACR and TCT slightly increase, indicating that scalability starts to be affected. Since the environment remains static, adaptability is not fully tested here.

- **Scenario 4 (Target relocation):** This scenario introduces dynamic target repositioning, requiring the agents to adjust their navigation strategies in real time. The agents need to adjust their paths to follow the moving target frequently. ACR increases due to the need to adopt the paths more regularly, leading to more collisions. Compared to Scenario 1, AMF slightly decreases because agents spend more time recalibrating their movements to chase the dynamic target. TCT increases, and with more agents or greater speeds, TCT increases even more, especially in congested situations. Scenario 4 tests MADDPG's adaptability. The ACR, AMF, and TCT increases reflect the growing challenges of adapting to moving targets. The fixed learning rate of MADDPG limits its scalability and adaptability under dynamic conditions,

with performance deterioration becoming more evident as the number of agents increases.

- **Scenario 5 (Moving obstacles):** The most complex scenario, involving dynamic moving obstacles. The complexity increases significantly as agents must navigate moving obstacles and dynamic targets. ACR peaks in this scenario as agents need to constantly adjust their paths to avoid other agents and moving obstacles. The dynamic nature of the obstacles makes it harder for agents to maintain safe distances. Compared to Scenario 1 and Scenario 4, AMF decreases because agents reduce speed and adjust more frequently to avoid collisions with moving obstacles and other agents. TCT is the longest among the three scenarios, as agents spend more time avoiding obstacles and adapting to the constantly changing environment. Scenario 5 demonstrates MADDPG's scalability and adaptability limitations in highly dynamic environments. As the number of agents increases, ACR and TCT grow significantly, indicating that the system struggles to scale with the increasing number of agents and dynamic obstacles. Moreover, the fixed learning rate reduces adaptability, as the agents cannot dynamically adjust their strategies to cope with the growing complexity of moving obstacles.

Additional results from Scenario 2 (higher obstacle density) and Scenario 3 (increased obstacle size) further reinforce these findings:

- **Scenario 2 (Higher obstacle density):** This scenario increases the obstacle density by adding more obstacles (six), creating a more chaotic environment. Due to the need to navigate around more obstacles, ACR increases moderately. AMF decreases because agents make more cautious adjustments to avoid collisions with the additional obstacles. TCT is longer than in Scenario 1, as agents take more time navigating the increased obstacles. Scenario 2 shows that higher obstacle density moderately impacts ACR, AMF, and TCT. While performance degradation is not as severe as in Scenario 4 or Scenario 5, it does highlight the impact of environmental complexity on scalability. With more obstacles introduced, agents need to adapt, but MADDPG with a fixed learning rate can still handle the environment with relatively good performance.

- **Scenario 3 (Increased obstacle size):** This scenario introduces larger obstacles (13 units), making agent navigation more challenging. The increase in ACR is more significant compared to Scenario 2, as the larger obstacles occupy more space, making it harder for agents to avoid collisions. AMF decreases because the larger obstacle sizes force agents to adjust their paths more frequently, slowing their movement. TCT increases significantly because agents need more time navigating the larger obstacles. Scenario 3 highlights the performance decline caused by larger obstacles. As expected, compared to Scenario 2, ACR increases, AMF decreases, and TCT increases. This demonstrates how the size of obstacles impacts an agent's ability to complete tasks effectively and suggests that MADDPG with a fixed learning rate is affected by environmental complexity, which may limit

TABLE 5. Simulation results for scenarios 1, 4, and 5 (S1, S4, S5)-MADDPG with SGD.

No. of Agents	Size of Agents	Velocity of Agents	ACR			AMF			TCT		
			S1	S4	S5	S1	S4	S5	S1	S4	S5
2	1	0.5	0.18	0.24	0.28	0.18	0.21	0.27	0.15	0.18	0.18
2	4	1.5	0.22	0.28	0.32	0.225	0.27	0.33	0.12	0.16	0.16
2	9	3.5	0.25	0.32	0.35	0.27	0.3	0.375	0.1	0.14	0.14
4	4	1.5	0.12	0.18	0.22	0.12	0.15	0.18	0.14	0.2	0.24
4	9	3.5	0.14	0.2	0.24	0.15	0.18	0.21	0.11	0.18	0.22
4	1	1.5	0.16	0.22	0.26	0.165	0.225	0.24	0.16	0.22	0.26
8	4	0.5	0.28	0.34	0.36	0.3	0.33	0.39	0.2	0.24	0.28
8	9	1.5	0.3	0.36	0.38	0.33	0.39	0.45	0.18	0.22	0.26
8	1	3.5	0.32	0.38	0.4	0.36	0.42	0.48	0.19	0.23	0.27
16	4	0.5	0.35	0.42	0.45	0.39	0.45	0.51	0.24	0.28	0.32
16	1	1.5	0.37	0.44	0.47	0.42	0.48	0.57	0.22	0.26	0.3
16	9	3.5	0.4	0.46	0.5	0.45	0.51	0.6	0.23	0.27	0.31
32	4	3.5	0.45	0.5	0.55	0.525	0.6	0.75	0.3	0.34	0.38
32	9	0.5	0.47	0.52	0.57	0.495	0.57	0.72	0.28	0.32	0.36
32	1	3.5	0.5	0.55	0.6	0.57	0.63	0.78	0.32	0.36	0.4
64	4	1.5	0.55	0.6	0.65	0.6	0.675	0.9	0.4	0.42	0.48
64	9	0.5	0.58	0.63	0.67	0.63	0.72	0.93	0.42	0.44	0.5
64	1	3.5	0.6	0.65	0.7	0.675	0.765	0.975	0.45	0.46	0.52
100	9	1.5	0.65	0.7	0.75	0.72	0.78	1.05	0.5	0.5	0.56
100	1	3.5	0.7	0.73	0.78	0.75	0.81	1.08	0.53	0.53	0.59
100	4	1.5	0.72	0.75	0.8	0.78	0.84	1.125	0.55	0.55	0.61
120	1	0.5	0.75	0.78	0.83	0.825	0.87	1.17	0.6	0.6	0.66
120	4	1.5	0.78	0.8	0.85	0.87	0.9	1.2	0.62	0.62	0.68

its adaptability in highly cluttered or dense spaces. It has been observed that as the number of agents increases, ACR and TCT show a significant increase in all scenarios. This reflects the scalability challenges. MADDPG faces a fixed learning rate in larger, more crowded environments. Scenario 5 emphasizes the scalability issue, where, as the number of agents increases, the task completion time becomes unreasonably long, and collision rates rise sharply due to congestion and dynamic obstacles. More agents and higher speeds lead to increased ACR and extended TCT, reflecting the pressure on MADDPG's ability to manage agent interactions effectively as the system scales. While Scenarios 4 and 5 highlight the importance of adaptability. As the environment becomes more dynamic with constantly moving targets and obstacles, MADDPG shows its limitations, particularly when using a fixed learning rate. Scenario 5 demonstrates MADDPG's difficulty adapting to the dynamic environment changes, leading to longer task completion times and higher collision rates. This suggests that MADDPG with a fixed learning rate may not be sufficient for complex environments. An adaptive learning rate is needed to improve performance.

The experimental results show that MADDPG with a fixed learning rate performs well in simpler environments like Scenario 1 but faces significant challenges in more complex environments like Scenarios 4 and 5. As the environment

becomes increasingly dynamic with moving targets and obstacles, the scalability and adaptability of the system become key limitations. These findings suggest that to scale and adapt effectively in such environments, adaptive learning mechanisms (e.g., adaptive learning rates) are essential for handling the growing complexity of multi-agent systems. The need for such improvements is especially apparent in environments with large numbers of agents and complex tasks. To address these issues, the next step is to utilize Surrogate Response Surface Models (SRSM), which will be key in optimizing agent-environment interactions and further enhancing scalability and adaptability.

2) STEP 2: SURROGATE RESPONSE SURFACE MODELING

In Step 2, we applied a central composite design method for the Surrogate Response Surface Modeling (SRSM) to address the scalability and adaptability challenges identified in Step 1. The goal was to explore the complex design space of agent-related variables and their impact on system performance. By fitting second-order polynomial models to data from the experimental runs, we captured the non-linear relationships between agent number (x), size (x), and velocity (x) and the key performance metrics—Agent Collision Rate (ACR), Agent Movement Frequency (AMF), and Task Completion Time (TCT). These relationships are expressed

TABLE 6. Simulation results for scenarios 1, 4, and 5 (S1, S4, S5) - MADDPG with SRSM.

ACR			AMF			TCT		
S1	S4	S5	S1	S4	S5	S1	S4	S5
0.16	0.22	0.26	0.165	0.195	0.24	0.14	0.17	0.18
0.2	0.26	0.3	0.21	0.24	0.28	0.11	0.14	0.15
0.23	0.29	0.33	0.255	0.285	0.345	0.09	0.12	0.13
0.1	0.16	0.2	0.105	0.135	0.165	0.12	0.18	0.23
0.12	0.18	0.22	0.135	0.165	0.195	0.1	0.16	0.21
0.14	0.2	0.24	0.15	0.18	0.225	0.15	0.19	0.25
0.26	0.3	0.34	0.27	0.3	0.39	0.18	0.22	0.27
0.28	0.32	0.36	0.3	0.33	0.42	0.17	0.2	0.25
0.3	0.34	0.38	0.33	0.36	0.45	0.18	0.21	0.26
0.33	0.38	0.42	0.36	0.39	0.48	0.22	0.25	0.3
0.35	0.4	0.44	0.39	0.42	0.51	0.19	0.23	0.28
0.37	0.42	0.46	0.42	0.45	0.54	0.2	0.24	0.29
0.4	0.45	0.5	0.48	0.54	0.66	0.26	0.3	0.36
0.42	0.47	0.52	0.45	0.51	0.63	0.24	0.28	0.34
0.45	0.5	0.55	0.525	0.57	0.69	0.28	0.32	0.38
0.5	0.55	0.6	0.57	0.63	0.78	0.36	0.38	0.46
0.52	0.57	0.62	0.6	0.66	0.81	0.38	0.4	0.48
0.55	0.6	0.65	0.63	0.69	0.855	0.4	0.42	0.5
0.6	0.64	0.7	0.69	0.75	0.93	0.44	0.46	0.54
0.65	0.67	0.69	0.72	0.78	0.96	0.47	0.49	0.57
0.68	0.7	0.72	0.75	0.81	0.99	0.49	0.51	0.59
0.7	0.73	0.75	0.78	0.84	1.02	0.53	0.56	0.64
0.73	0.75	0.78	0.825	0.87	1.05	0.55	0.58	0.66

in Equations 1–3, providing insights into the trade-offs in system design.

These models reveal the complex trade-offs in system design, such as the positive relationship between agent velocity and ACR, highlighting the increased likelihood of collisions at higher speeds. This underscores the importance of balancing agent speed and density in large systems. With R^2 values exceeding 95 percent and all terms statistically significant ($p < 0.05$), these models demonstrate a robust fit, offering reliable predictions for system performance across various configurations. In this context, the models are instrumental in optimizing multi-agent systems for scalability and adaptability as they enable rapid performance evaluation without the need for exhaustive simulations.

$$\begin{aligned}
 \text{Collision Rate} = & 0.2609 + 0.0076x_1 - 3.3333x_2 \\
 & - 0.0076x_3 - 0.000035x_1^2 \\
 & + 24.706034x_2^2 \\
 & + 0.001024x_3^2 + 0.0108x_1 \cdot x_2 \\
 & - 0.0001x_1 \cdot x_3 + 0.1400x_2 \cdot x_3
 \end{aligned} \quad (1)$$

$$\begin{aligned}
 \text{Movement Frequency} = & 0.1173 + 0.0139x_1 \\
 & + 4.2472x_2 + 0.0304x_3 \\
 & - 0.000064x_1^2 - 29.282978x_2^2
 \end{aligned}$$

$$\begin{aligned}
 & - 0.008209x_3^2 - 0.0149x_1 \cdot x_2 \\
 & - 0.0002x_1 \cdot x_3 + 0.3242x_2 \cdot x_3
 \end{aligned} \quad (2)$$

$$\begin{aligned}
 \text{Task Completion Time} = & 0.1673 + 0.0063x_1 \\
 & + 0.5196x_2 - 0.0261x_3 \\
 & - 0.000021x_1^2 - 4.299454x_2^2 \\
 & - 0.000318x_3^2 - 0.0020x_1 \cdot x_2 \\
 & + 0.0001x_1 \cdot x_2 + 0.3886x_1 \cdot x_3
 \end{aligned} \quad (3)$$

In Table 6, we present the predicted values for ACR, AMF, and TCT after applying SRSM optimization to different agent configurations (number of agents, agent size, and speed). The optimization focuses on reducing ACR, increasing AMF, and minimizing TCT. SRSM optimization decreases ACR for most agent configurations, particularly in Scenario 5 (S5), where the environment is most dynamic with moving obstacles. For example, in Scenario 5, for the configuration of 2 agents, size 9, and speed 3.5, the predicted ACR decreases from 0.25 (before SRSM) to 0.18 (after SRSM). Reducing ACR is a key improvement, highlighting how SRSM helps agents navigate the environment more effectively, even in complex environments with dynamic elements like moving obstacles or relocating targets, thereby

avoiding collisions. This decrease also underscores the role of SRSM in optimizing agent-environment interactions, enhancing scalability in larger environments.

After optimization, AMF shows notable improvement. For instance, in Scenario 5, with 2 agents, size 9, and speed 0.5, AMF increases from 0.20 to 0.25. This indicates that SRSM optimization enhances the agents' movement, allowing them to navigate the environment more efficiently, especially when dealing with dynamic obstacles. The increase in AMF reflects that SRSM reduces collision rates and helps agents move more effectively, addressing the challenge of adaptability where agents must continuously adjust their paths to avoid obstacles while heading toward their goals. One of the most significant outcomes of SRSM optimization is the reduction in TCT, especially in more complex environments. For example, in Scenario 5, with 2 agents, size 9, and speed 0.5, TCT decreases from 0.18 (before SRSM) to 0.15 (after SRSM). This reduction suggests that SRSM optimizes the agents' movements, allowing them to complete tasks more quickly while avoiding obstacles and adapting to dynamic targets. The decrease in TCT improves task efficiency and boosts scalability, enabling the system to handle a higher number of agents or more complex tasks in a shorter time.

SRSM optimization significantly improves the system's scalability, particularly in larger and more complex environments. As shown in the table, for higher agent configurations (e.g., 64, 100, and 120 agents), the reduction in ACR and TCT demonstrates that SRSM enables MADDPG to manage a larger number of agents without a substantial decrease in performance. In the absence of SRSM, scalability may be compromised, as the increase in agent numbers leads to higher collision rates and longer task completion times. By optimizing AMF and reducing both ACR and TCT, SRSM greatly enhances the system's adaptability. After optimization, agents are more responsive to dynamic changes in the environment, especially in Scenario 5, where agents must adapt to moving obstacles and dynamic targets. SRSM allows agents to adjust their movement in real time, thereby improving task completion efficiency, reducing collisions, and enhancing the system's adaptability in dynamic environments.

The response surface plots in Fig. 8 provide clear visualizations of the relationships between performance metrics, offering insights into how these metrics vary across the design space. These plots are particularly valuable for highlighting critical interactions influencing system performance under varying conditions. In Scenario 1, the focus on agent numbers and velocities offers foundational insights into how the system scales in a relatively controlled environment, serving as a baseline for comparison. Scenario 4, which introduces relocating targets, emphasizes the interaction between agent size and agent count in a dynamic environment. This scenario captures the challenges of maintaining adaptability and coordination when responding to shifting goals. By contrast, Scenario 5, characterized by moving obstacles, represents the most complex setup. Here, the interplay between obstacle

velocity and agent count highlights the critical challenges of dynamic, non-stationary obstacles.

While the selected plots provide detailed visualizations, additional response surface plots—such as those depicting movement frequency or varying obstacle sizes—serve as references, corroborating and complementing the primary insights. Together, these visualizations illustrate a progressive increase in complexity, offering actionable insights for optimizing multi-agent systems in terms of scalability and adaptability across diverse scenarios. By focusing on key plots while considering others for context, the analysis remains targeted yet comprehensive.

The response surface plots succinctly demonstrate how agent-related variables—such as number of agent, velocity, and size of agent affects the output performance metrics across scenarios 1, 4, and 5. These insights highlight the trade-offs between scalability and adaptability in static and dynamic environments. In Scenario 1, the simple static setup shows that ACR moderately increases as agent numbers and velocities rise, reflecting coordination challenges as agent density and speed increase, even in less complex environments. In Scenario 4, where targets relocate dynamically, ACR increases sharply, particularly at higher agent densities and velocities. This reflects the added difficulty of collision-free navigation as agents adjust to shifting targets, underscoring the need for enhanced real-time adaptability mechanisms. In Scenario 5, with moving obstacles, ACR peaks, demonstrating that dynamic obstacles significantly increase the risk of collisions, especially at elevated agent velocities. This scenario underscores the compounded challenges of scalability and adaptability in highly dynamic environments.

For TCT, the trends reveal further trade-offs. In Scenario 1, TCT decreases with an increasing number of agents, signaling improved efficiency in static environments. However, higher agent velocities slightly increase TCT due to higher collision rates. In Scenario 4, TCT rises as relocating targets introduce complexity, with efficiency improving at higher agent densities only up to a threshold. Beyond this threshold, higher velocities exacerbate coordination issues, further delaying task completion. In Scenario 5, TCT increases sharply at higher agent densities and velocities as agents navigate moving obstacles, emphasizing the need for adaptive strategies to maintain task efficiency in dynamic environments.

The insights gained from Step 2, leveraging surrogate modeling and Response Surface Modeling (RSM), offer a robust understanding of the complex relationships between agent-related variables and system performance metrics across diverse scenarios. Surrogate modeling facilitates efficient design space exploration, addressing computational constraints while optimizing scalability and adaptability. However, the limitations observed in Step 1, where MADDPG with Stochastic Gradient Descent (SGD) struggled in dynamic scenarios—particularly in environments with relocating targets (Scenario 4) and moving obstacles

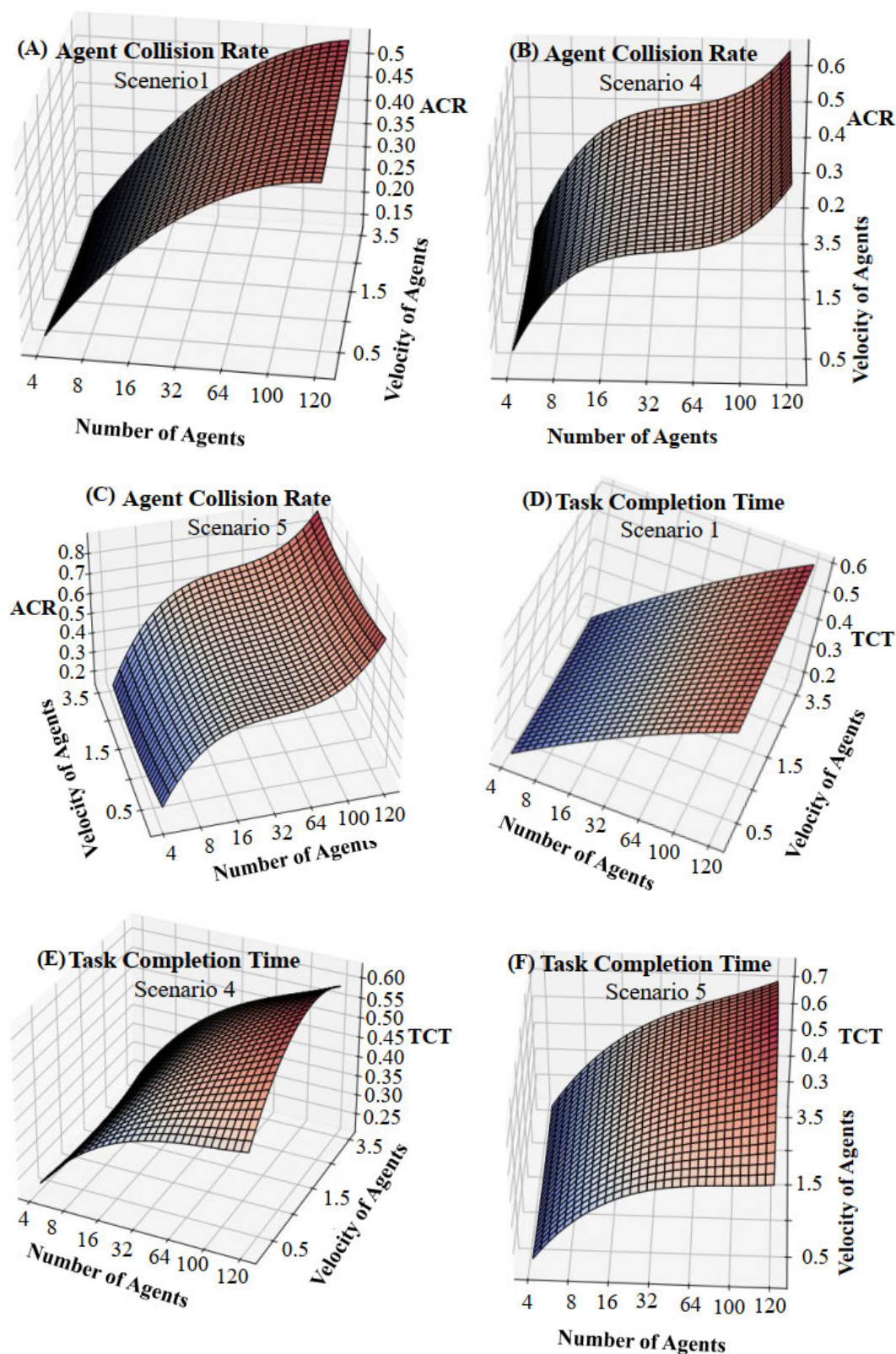


FIGURE 8. Response surface analysis of performance metrics across scenarios 1,4,5 (A-C Agent collision Rate), (D-F Task completion time).

(Scenario 5)—highlight the need for further advancements. The fixed learning rate of SGD fails to adapt to non-stationary environments and evolving dynamics, leading to suboptimal performance as the environment becomes more complex.

Building on the optimization insights from Step 2, Step 3 integrates MADDPG with an adaptive learning rate (ALR) to further enhance the system's scalability with increasing agent density and adaptability to real-time dynamic changes. ALR dynamically adjusts the learning rates based on gradient magnitudes and training conditions, addressing the rigidity of fixed-rate optimizers and enabling better performance in complex environments. This integration leverages the results from Step 2, where SRSF helped identify key agent configurations and optimization strategies. By combining these insights with ALR, Step 3 demonstrates how MADDPG can improve scalability and adaptability, leading to superior system performance in real-world, high-density, dynamic multi-agent environments.

3) STEP 3: ADVANCED OPTIMIZATION WITH MADDPG + ALR (ADAM AND AADMA)

Based on the limitations of MADDPG identified using the SGD optimizer in Step 1 and the optimization insights derived from the surrogate model in Step 2, this section introduces the integration of MADDPG with adaptive learning rate (ALR) optimizers, specifically ADAM and AADMA, to overcome scalability and adaptability challenges in complex multi-agent environments. ADAM and AADMA dynamically adjust the learning rate based on the gradient magnitude, promoting faster convergence and improved performance in dynamic environments. We evaluate the performance of MADDPG with ALR (using ADAM and AADMA) across the same five scenarios presented earlier, focusing on key performance indicators: ACR, AMF, and TCT. By comparing these results with those obtained using MADDPG with SGD, we demonstrate the superior scalability and adaptability achieved by integrating ALR. The results summarized in Table 7 particularly highlight the performance differences across Scenario 1 (S1), Scenario 4 (S4), and Scenario 5 (S5).

The ACR values in Scenario 1 (S1), Scenario 4 (S4), and Scenario 5 (S5) show that AADMA generally performs better than ADAM in reducing collisions. For example, in Scenario 5 (S5), with 2 agents, size 4, and speed 1.5, the ACR drops from 0.16 (with ADAM) to 0.15 (with AADMA), indicating fewer collisions with AADMA. As the number of agents increases, AADMA helps maintain a stable or slightly reduced ACR, whereas ADAM gradually increases, especially in dynamic environments with moving obstacles. This suggests that AADMA is more effective at mitigating collisions, particularly as environmental complexity increases.

As the number of agents and their speed increase, ADAM and AADMA show an upward trend in AMF, with AADMA optimizing AMF to higher values, indicating improved movement efficiency. In Scenario 5 (S5), with 2 agents, size 4, and speed 1.5, AMF increases from 0.22 (with ADAM) to

0.28 (with AADMA), reflecting that AADMA helps agents move more efficiently in dynamic environments. AMF is particularly important in scenes like S5, where agents must constantly adapt to moving obstacles and targets. The higher AMF value for AADMA indicates that it allows agents to adjust their actions more effectively, thus improving overall task performance.

From the results, we have observed that AADMA reduces TCT across all scenarios compared to ADAM. For example, in Scenario 5 (S5), with 2 agents, size 4, and speed 1.5, the TCT decreases from 0.16 (with ADAM) to 0.14 (with AADMA), indicating that AADMA optimizes task completion time, allowing agents to complete tasks more efficiently, even in complex environments. As agent density and speed increase, the TCT for both optimizers increases, but AADMA consistently provides lower TCT values, suggesting better scalability and efficiency when handling larger tasks or more agents.

Across all three performance metrics, AADMA outperforms ADAM, especially as the number of agents and environmental complexity (e.g., dynamic obstacles and moving targets) increases. AADMA enables agents to navigate more efficiently, avoid collisions, and complete tasks faster, particularly in more dynamic environments. In Scenario 5 (with moving obstacles), the gap between ADAM and AADMA becomes even more pronounced, highlighting AADMA's superior ability to adapt to increasingly complex environments.

The comparison of performance metrics further emphasizes the differences observed when integrating MADDPG with ADAM and AADMA in the selected scenarios. Regarding ACR, ADAM maintains a lower collision rate in simple environments, such as Scenario 1 (S1), where static obstacles create a predictable environment. However, in Scenario 4 (repositioning targets) and Scenario 5 (moving obstacles), ACR increases sharply with ADAM, reflecting its limited adaptability to dynamic environments. In contrast, AADMA performs significantly better in these complex scenarios, dynamically optimizing agent paths and interactions to reduce collisions. Examining AMF, we find that ADAM achieves a higher movement frequency in simpler environments like S1, where coordination among agents is easier. However, as the complexity increases in Scenario 4 and Scenario 5, ADAM's AMF decreases, indicating a reduction in agent activity and coordination. On the other hand, AADMA maintains higher AMF values in these dynamic scenarios, showing its effective use of adaptive learning rates to optimize agent movement and maintain consistent and efficient behavior.

In general, the results highlight the strengths and limitations of both ADAM and AADMA. While ADAM provides a strong baseline performance in simpler environments, it becomes less effective when scaling and adapting to more complex multi-agent systems. In contrast, AADMA demonstrates exceptional scalability and adaptability, particularly in dynamic and complex environments. Improves

TABLE 7. Simulation result for scenarios 1, 4, and 5 (S1, S4, S5) -MADDPG with ALR (ADAM and AADMA).

ACR						AMF						TCT					
ADAM			AADMA			ADAM			AADMA			ADAM			AADMA		
S1	S4	S5	S1	S4	S5	S1	S4	S5	S1	S4	S5	S1	S4	S5	S1	S4	S5
0.14	0.18	0.22	0.15	0.2	0.24	0.15	0.18	0.24	0.165	0.195	0.255	0.14	0.17	0.17	0.12	0.16	0.16
0.17	0.22	0.26	0.18	0.24	0.28	0.195	0.225	0.285	0.21	0.24	0.3	0.1	0.14	0.14	0.1	0.13	0.14
0.2	0.26	0.3	0.21	0.27	0.32	0.24	0.27	0.33	0.255	0.285	0.33	0.08	0.12	0.12	0.08	0.11	0.12
0.08	0.14	0.18	0.09	0.15	0.19	0.105	0.135	0.165	0.09	0.12	0.18	0.13	0.18	0.22	0.12	0.17	0.21
0.1	0.16	0.2	0.11	0.17	0.21	0.12	0.165	0.195	0.12	0.15	0.21	0.1	0.15	0.2	0.09	0.14	0.19
0.12	0.18	0.22	0.13	0.19	0.23	0.15	0.18	0.225	0.135	0.18	0.24	0.14	0.18	0.23	0.13	0.18	0.23
0.22	0.28	0.32	0.23	0.29	0.33	0.27	0.33	0.42	0.255	0.3	0.42	0.16	0.2	0.24	0.15	0.2	0.24
0.24	0.3	0.34	0.25	0.31	0.35	0.3	0.36	0.45	0.285	0.33	0.45	0.15	0.18	0.22	0.14	0.18	0.22
0.26	0.32	0.36	0.27	0.33	0.37	0.33	0.39	0.48	0.3	0.36	0.48	0.14	0.17	0.23	0.13	0.17	0.23
0.28	0.35	0.4	0.3	0.36	0.41	0.36	0.42	0.51	0.33	0.39	0.51	0.19	0.22	0.27	0.18	0.21	0.26
0.3	0.37	0.42	0.32	0.38	0.43	0.39	0.45	0.54	0.36	0.42	0.54	0.18	0.21	0.26	0.17	0.2	0.25
0.32	0.39	0.44	0.34	0.4	0.45	0.42	0.48	0.57	0.39	0.45	0.57	0.19	0.22	0.27	0.18	0.21	0.26
0.35	0.42	0.47	0.36	0.43	0.48	0.465	0.57	0.675	0.495	0.54	0.69	0.24	0.26	0.32	0.22	0.25	0.31
0.37	0.44	0.5	0.38	0.45	0.5	0.435	0.54	0.63	0.465	0.51	0.66	0.22	0.24	0.3	0.21	0.23	0.29
0.4	0.47	0.53	0.4	0.48	0.53	0.51	0.6	0.705	0.525	0.57	0.72	0.25	0.27	0.33	0.24	0.26	0.32
0.42	0.5	0.56	0.43	0.52	0.57	0.54	0.63	0.765	0.57	0.615	0.78	0.32	0.32	0.4	0.3	0.31	0.39
0.44	0.52	0.58	0.45	0.54	0.59	0.585	0.675	0.795	0.6	0.645	0.81	0.33	0.33	0.42	0.31	0.32	0.41
0.46	0.54	0.6	0.47	0.56	0.61	0.615	0.705	0.825	0.63	0.675	0.855	0.36	0.36	0.44	0.34	0.35	0.43
0.5	0.57	0.64	0.5	0.6	0.65	0.675	0.765	0.885	0.69	0.735	0.93	0.38	0.39	0.48	0.37	0.38	0.47
0.53	0.6	0.67	0.52	0.63	0.68	0.705	0.795	0.915	0.72	0.765	0.96	0.41	0.42	0.51	0.4	0.41	0.5
0.55	0.62	0.7	0.54	0.65	0.7	0.735	0.825	0.945	0.75	0.795	0.99	0.44	0.45	0.53	0.42	0.43	0.52
0.58	0.65	0.72	0.56	0.68	0.73	0.765	0.855	0.975	0.78	0.825	1.02	0.46	0.47	0.56	0.45	0.46	0.55
0.6	0.68	0.74	0.58	0.7	0.75	0.81	0.885	1.005	0.795	0.855	1.05	0.49	0.5	0.58	0.48	0.49	0.57

collision rate, increases agent movement frequency, and reduces task completion time. It is a more powerful choice for practical applications in multi-agent systems, where scalability and adaptability are critical. These findings emphasize the importance of integrating adaptive learning rate strategies into multi-agent systems. By dynamically optimizing agent behavior in real time, AADMA overcomes the limitations of fixed learning rate optimizers, improving the scalability and adaptability of key performance metrics. This integration provides a robust framework for multi-agent systems, ensuring high performance and efficiency in complex, dynamic environments.

4) COMPARISON OF SGD, SRSM, ADAM, AND AADMA

The performance comparison chart of SGD, SRSM, ADAM, and AADMA presented in Fig. 9 evaluates their effectiveness across S1, S4, and S5 based on key performance indicators: ACR, AMF, and TCT. These metrics provide valuable insights into the scalability, adaptability, and efficiency of the different optimization methods when applied to increasingly complex environments.

From the Agent Collision Rate (ACR) chart, it is clear that as the number of iterations increases, the collision rate increases for all optimization techniques across all scenarios. This trend suggests that collisions become more frequent as agents interact, although further fine-tuning or different environmental conditions may be needed. The SGD-ACR configuration (solid blue line) shows relatively higher and steeper increases in collision rates across the different scenarios, indicating suboptimal collision reduction compared to other methods. SRSM-ACR (dashed lines) shows a noticeable improvement over SGD. Specifically, SRSM stabilizes the increase in collision rates in the three scenarios, maintaining a more consistent performance than SGD. The ADAM-ACR configuration (solid cyan, orange, and purple lines) tends to better control the collision rate, with particularly improved performance in Scenario 4 and Scenario 5, where the collision rate is lower and the increase is more gradual compared to SGD. ADAMA-ACR (solid magenta, brown, and gray lines) also shows improved performance, though not as strong as ADAM in all cases. ADAMA may offer better performance in Scenario 5, where

the collision rate is minimized more effectively than in other configurations.

In all scenarios, the collision rate increases with the number of iterations. SGD shows the highest and fastest increase in ACR, indicating poor collision avoidance performance. SRSM performs better, stabilizing the increase in collisions. ADAM controls the collision rate well, with notable performance improvements in S4 and S5. AADMA outperforms all other methods in all scenarios, particularly in S5, where it achieves a 20 percent improvement over SGD.

The Agent Movement Frequency (AMF) chart illustrates how agent movement efficiency changes over iterations. For all configurations, there is a noticeable increase in movement frequency, indicating that agents become more active as iterations progress. The SGD-AMF results (solid blue, red, and green lines) show improved movement frequency across scenarios but with relatively poor stability and slower increases compared to other methods. This suggests suboptimal exploration and learning behavior under the SGD configuration. The SRSM-AMF (dashed lines) shows better consistency in agent movement and faster improvement in Scenario 4 and Scenario 5, indicating that SRSM helps agents achieve better exploration and movement balance, even as the number of agents and environmental complexity increase. The ADAM-AMF lines (solid cyan, orange, and purple) show the largest increase in movement frequency. ADAM adapts more quickly and effectively than SGD and SRSM, demonstrating a faster learning process that enhances agent movement across all scenarios. ADAMA-AMF shows competitive performance in the early iterations (solid magenta, brown, and gray lines), and appears to stabilize after the 10th iteration. This suggests that ADAMA may need further fine-tuning to outperform ADAM, though it still improves significantly compared to SGD.

Movement frequency improves in all methods, and SGD shows slower and less consistent progress. SRSM offers better movement efficiency, especially in more complex scenarios. ADAM shows the fastest increase in AMF, reflecting its superior adaptability. AADMA consistently outperforms all other methods, maintaining the highest and most stable movement frequency, particularly in dynamic scenarios like S5, achieving a 42 percent improvement over SGD.

The Task Completion Time (TCT) chart illustrates the efficiency of the multi-agent system in completing tasks. Across all optimization techniques, TCT decreases as iterations progress, indicating improved agent efficiency over time. SGD-TCT (solid blue, red, and green lines) consistently shows the highest TCT values across all scenarios, suggesting slower task completion and inefficient learning behavior in optimizing task-related actions. SRSM-TCT (dashed lines) significantly reduces TCT compared to SGD, demonstrating faster convergence and improved task completion efficiency. ADAM-TCT (solid cyan, orange, and purple lines) consistently achieves the lowest TCT values across all scenarios, particularly excelling in S4 and S5. This indicates that ADAM

optimizes task completion effectively, providing faster convergence and improved learning efficiency. AADMA-TCT (solid magenta, brown, and gray lines) reduces TCT further than SGD but does not surpass ADAM in all cases. However, it still enhances task completion efficiency, particularly in S5, where it outperforms SGD by 22 percent.

Across ACR, AMF, and TCT, AADMA consistently outperforms SGD, SRSM, and ADAM, especially in dynamic and complex environments such as S4 (relocating targets) and S5 (moving obstacles). AADMA's adaptive learning rate enables efficient navigation, better collision avoidance, and faster task completion, whereas SGD struggles due to its static learning rate, making it the least adaptive.

The results are shown in Fig. 9, where different optimization strategies are compared:

- MADDPG with SGD (Baseline): Serves as a reference, showing slower convergence and higher collision rates.
- MADDPG + SRSM: Improves ACR and TCT by enhancing scalability and adaptability.
- MADDPG with ADAM: This incorporates adaptive learning rate optimization, improving convergence speed, stability, and collision rates.
- MADDPG with AADMA: Further enhances performance, reducing collision rates and improving overall system stability.

These results indicate that while SRSM optimizes sample efficiency, the adaptive learning rate (ALR) stabilizes convergence, particularly in dynamic environments. Both methods benefit significantly when tested independently, improving MADDPG's overall performance. Their combined application enhances efficiency, scalability, and adaptability in complex multi-agent environments.

The loss function analysis graph presented in Fig. 10 compares the performance of SGD, SRSM, ADAM, and AADMA across ACR, AMF, and TCT over more than 20 iterations in S1, S4, and S5. In simpler environments like S1, all optimization methods show stable, low-loss values. SGD performs efficiently with low ACR, AMF, and TCT values, while SRSM shows slight fluctuations due to its model-based approach. ADAM performs slightly better than SGD in terms of ACR and AMF but lacks the adaptive learning rate of AADMA. AADMA consistently outperforms the other methods, maintaining low TCT, high AMF, and low ACR, highlighting the advantage of its adaptive learning rate.

As the complexity of the environment increases in S4 (reallocating target) and S5 (moving obstacles), the performance trends of the optimization methods diverge more clearly. SGD's loss values increase significantly, especially in S5, where dynamic obstacles introduce greater coordination challenges. With its fixed learning rate, SGD struggles to adapt to these dynamic changes, leading to a sharp rise in ACR, AMF, and TCT loss values. This issue is particularly evident in S5, where SGD fails to effectively adjust to the changing conditions. SRSM, despite being a model-based optimization method, shows similar performance to SGD in S4 and S5, with slight increases in ACR and TCT due to

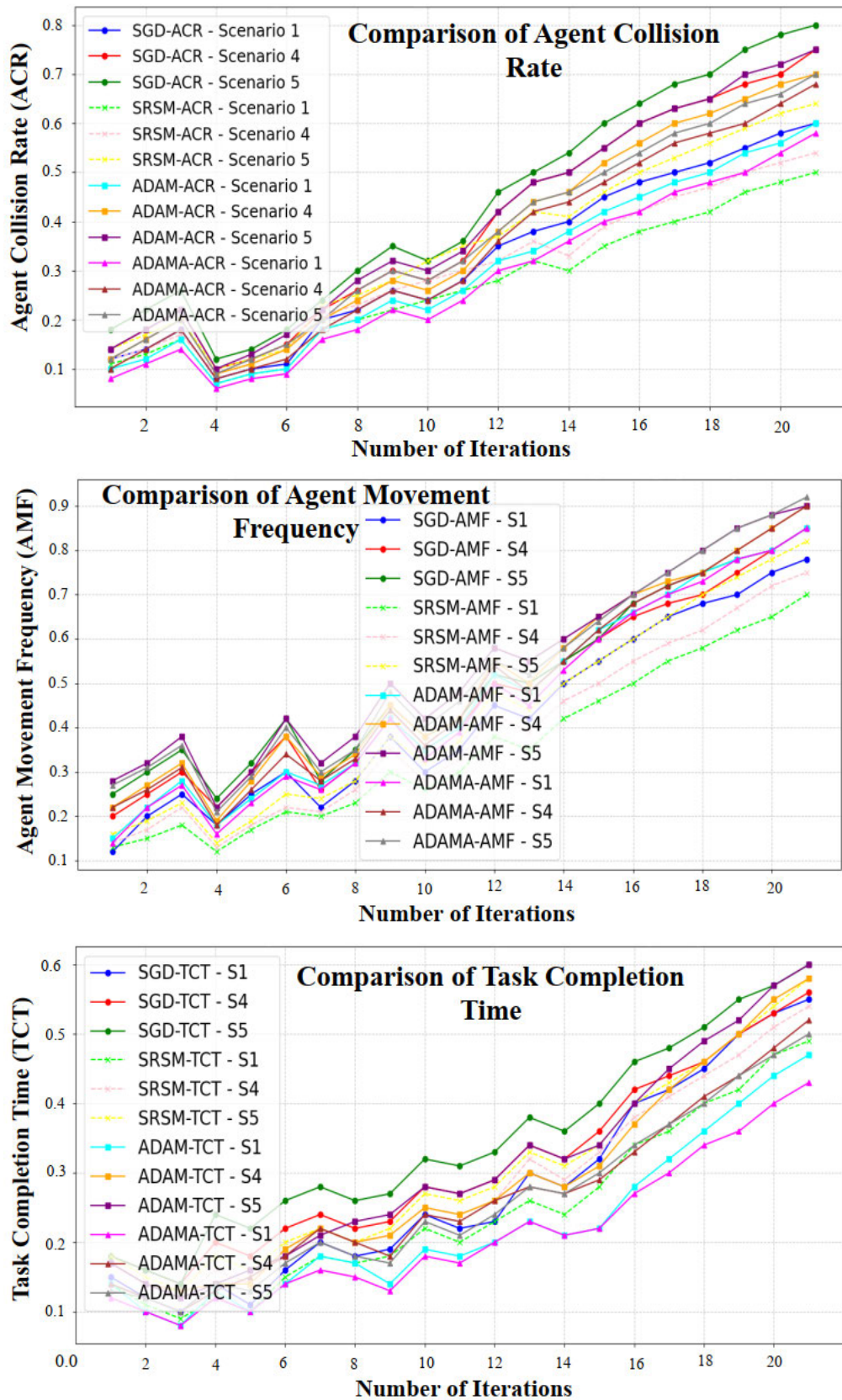


FIGURE 9. Performance comparison charts.

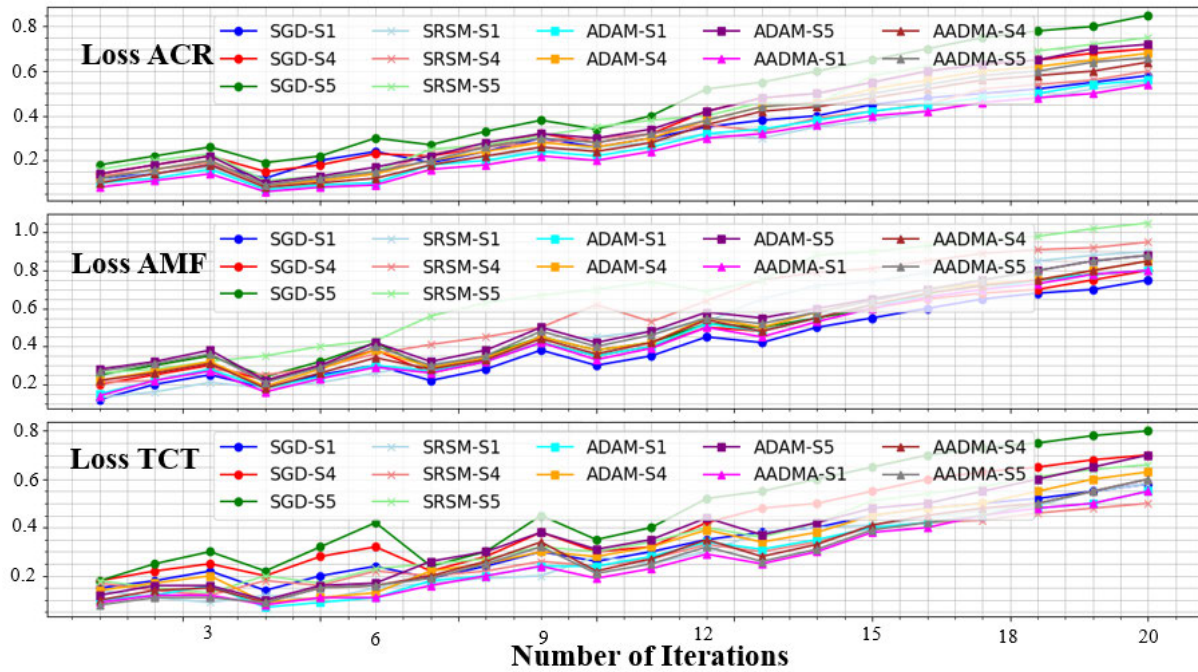


FIGURE 10. Loss function comparison of SGD, SRSM, ADAM, and AADMA.

the limitations of the agent model in dynamic environments. SRSM is typically more efficient in high-dimensional or computationally expensive problems, but it is less effective in environments with continuous, real-time changes.

By benefiting from its gradient-based adaptability, ADAM shows moderate improvements over SGD, especially in TCT and AMF. While it performs better than SGD, it still does not match the robustness and efficiency demonstrated by AADMA. In S4 and S5, ADAM shows some adaptability, but its performance is less consistent than AADMA. As environmental complexity increases, ADAM's performance becomes suboptimal as it cannot adjust its learning rate as effectively as AADMA. AADMA, with its adaptive learning rate mechanism, enables the optimization process to adapt to changes in S4 and S5 dynamically. As a result, AADMA achieves significantly lower loss values in ACR, AMF, and TCT compared to SGD and ADAM in these complex environments. Its ability to adapt to changing conditions minimizes the impact of dynamic obstacles and target relocation, ensuring more stable agent coordination and improving overall performance.

The comparative analysis of ACR, AMF, and TCT with SGD, ADAM, SRSM, and AADMA reveals a clear trend: Integrating an adaptive learning mechanism significantly improves system performance in dynamic and complex environments. While SGD and SRSM show limitations in handling the evolving characteristics of multi-agent systems, particularly in environments with high agent density and complex obstacles, AADMA stands out as the most robust and effective method. The analysis emphasizes that AADMA's adaptive learning rate enables it to handle

non-linear environmental complexities efficiently, achieving lower collision rates, sustained movement efficiency, and faster task completion times. This makes AADMA highly effective in real-world applications. The results underscore the critical role of adaptive optimization techniques in overcoming the challenges posed by dynamic and complex multi-agent environments, paving the way for solutions with greater scalability and real-time adaptability in multi-agent systems.

V. DISCUSSION

In this study, we extend our previous work [13] by introducing a powerful multi-agent system (MAS) framework that integrates agent modeling and adaptive learning mechanisms to address scalability and adaptability challenges in dynamic environments. This new framework builds on our prior research, combining agent-related variables with environment-related variables to ensure efficient system operation in complex real-world scenarios.

First, we evaluate the system's performance using MAD-DPG with an SGD optimizer, measuring metrics such as Agent Collision Rate (ACR), Agent Movement Frequency (AMF), and Task Completion Time (TCT). While SGD performs well in static environments, its fixed learning rate is inefficient in complex dynamic environments, leading to increased collision rates and longer task completion times. The second stage combines the response surface method (RSM) with central composite design to address this, effectively modeling the nonlinear agent-environment relationship. This alternative model reduces computational costs and highlights key trade-offs, such as the balance

between agent speed and ACR. In the final stage, we integrate MADDPG with an adaptive learning rate (ALR) optimizer, using ADAM and AADMA. Results show that AADMA outperforms ADAM and SGD, particularly in environments with dynamic obstacles and target repositioning, thus enhancing system adaptability and performance.

Despite these advances, some limitations remain. Integrating adaptive learning rates introduces computational overhead, especially in high-agent-density scenarios. Scalability tests with up to 120 agents show that our approach is well-suited for deployment in real-world systems such as autonomous vehicle coordination, robot swarm management, and smart city traffic control. Our framework can quickly adapt to dynamic environments while maintaining agent coordination, which is crucial for applications requiring real-time decision-making and adaptability.

However, our research mainly focuses on controlled simulation environments, and further testing is needed in more realistic environments. Future work will involve expanding this system to run with larger agent populations (potentially up to thousands of agents) and incorporating advanced technologies like parallel processing to manage computational overhead in high-density scenarios. In addition, we plan to test the system on real-world robotic platforms to assess its performance in practical tasks such as automated warehouse management and urban traffic control. The integration of these systems will allow us to better evaluate scalability, adaptability, and real-time performance under complex and dynamic conditions.

VI. CLOSING REMARKS

In this study, we highlight the critical role of surrogate modeling and adaptive learning mechanisms in enhancing multi-agent systems (MAS) for dynamic and complex environments. The proposed framework effectively addresses scalability and adaptability challenges by integrating MADDPG with adaptive learning rates (ALR) and surrogate response surface modeling (SRSR). The findings reveal that traditional methods like MADDPG with SGD struggle in dynamic scenarios, while AADMA consistently outperforms both SGD and ADAM by achieving superior scalability, adaptability, and robustness. AADMA's dynamic learning rate adjustments enable real-time adaptation, reducing collision rates, improving movement efficiency, and shortening task completion times, even in scenarios involving relocating targets and moving obstacles. SRSR complements these adaptive learning mechanisms by enabling efficient design space exploration, reducing computational costs, and delivering reliable performance predictions. Together, these methodologies enhance MAS scalability and adaptability, making them more suitable for real-world applications like swarm robotics, distributed systems, and collaborative task execution. Future work will address computational overhead in high agent-density settings, extend the framework's scalability to ultra-large systems, and broadening its applicability to competitive and resource-constrained environments. These

advancements will further strengthen MAS's performance, ensuring its effectiveness in solving real-world challenges in dynamic and unpredictable scenarios.

ACKNOWLEDGMENT

Raza Muhammad Kazim acknowledges the help with coding provided by Ph.D. students Mingfei Jiang and Amjad Muhammad from Beijing Institute of Technology.

REFERENCES

- [1] A. Dorri, S. S. Kanhere, and R. Jurdak, "Multi-agent systems: A survey," *IEEE Access*, vol. 6, pp. 28573–28593, 2018.
- [2] V. François-Lavet, P. Henderson, R. Islam, M. G. Bellemare, and J. Pineau, "An introduction to deep reinforcement learning," *Found. Trends Mach. Learn.*, vol. 11, nos. 3–4, pp. 219–354, Jan. 2018.
- [3] H. Yu, Z. Shen, C. Leung, C. Miao, and V. R. Lesser, "A survey of multi-agent trust management systems," *IEEE Access*, vol. 1, pp. 35–50, 2013.
- [4] L. Kaiser, M. Babaeizadeh, P. Milos, B. Osinski, R. H. Campbell, K. Czechowski, D. Erhan, C. Finn, P. Kozakowski, S. Levine, A. Mohiuddin, R. Sepassi, G. Tucker, and H. Michalewski, "Model-based reinforcement learning for Atari," 2019, *arXiv:1903.00374*.
- [5] L. Shi, Z. Ma, S. Yan, and Y. Zhou, "Cucker-smale flocking behavior for multiagent networks with cooperation interactions and communication delays," *IEEE Trans. Syst., Man, Cybern., Syst.*, vol. 54, no. 9, pp. 5824–5833, Sep. 2024.
- [6] Y. Shang, D. Li, Z. Peng, A. Heydari, K. Turkoglu, and J. Thunberg, "IEEE access special section editorial: Recent developments in consensus problems for complex networked systems," *IEEE Access*, vol. 6, pp. 60993–60995, 2018.
- [7] H. Ji and Y. Jin, "Impact of task constraint on agent team size of self-organizing systems measured by effective entropy," *J. Comput. Inf. Sci. Eng.*, vol. 24, no. 8, Aug. 2024.
- [8] H. Geng, H. Wu, J. Miao, S. Hou, and Z. Chen, "Consensus of heterogeneous multi-agent systems under directed topology," *IEEE Access*, vol. 10, pp. 5936–5943, 2022.
- [9] M. Nanjanath and M. Gini, "Repeated auctions for robust task execution by a robot team," *Robot. Auto. Syst.*, vol. 58, no. 7, pp. 900–909, Jul. 2010.
- [10] Y. Rizk, M. Awad, and E. W. Tunstel, "Decision making in multiagent systems: A survey," *IEEE Trans. Cognit. Develop. Syst.*, vol. 10, no. 3, pp. 514–529, Sep. 2018.
- [11] J. O. Berndt and O. Herzog, "Efficient multiagent coordination in dynamic environments," in *Proc. IEEE/WIC/ACM Int. Conferences Web Intell. Intell. Agent Technol.*, vol. 2, Aug. 2011, pp. 188–195.
- [12] M. H. Qasem, N. Obeid, A. Hudaib, M. A. Almaiah, A. Al-Zahrani, and A. Al-Khasawneh, "Multi-agent system combined with distributed data mining for mutual collaboration classification," *IEEE Access*, vol. 9, pp. 70531–70547, 2021.
- [13] M. Jiang, Z. Ming, C. Li, J. K. Allen, and F. Mistree, "Design of self-organizing systems using multi-agent reinforcement learning and the compromise decision support problem construct," *J. Mech. Design*, vol. 146, no. 5, May 2024, Art. no. 051711.
- [14] Y. Li and X. Chen, "Modeling and simulation of a swarm of robots for box-pushing task," in *Proc. 12th Medit. Conf. Control Autom.*, Kusadasi, Turkey, 2004, pp. 1–16.
- [15] Z. Lončarević and A. Gams, "Effects of increased entropy on robustness of reinforcement learning for robot box-pushing," in *Proc. Int. Conf. Robot. Alpe-Adria Danube Region*, Jan. 2024, pp. 97–105.
- [16] J. Zhou, X. Zhao, X. Zhang, D. Zhao, and H. Li, "Task allocation for multi-agent systems based on distributed many-objective evolutionary algorithm and greedy algorithm," *IEEE Access*, vol. 8, pp. 19306–19318, 2020.
- [17] H.-B. Choi, J.-B. Kim, Y.-H. Han, S.-W. Oh, and K. Kim, "MARL-based cooperative multi-AGV control in warehouse systems," *IEEE Access*, vol. 10, pp. 100478–100488, 2022.
- [18] A. Alharbi, G. Alshehri, and S. Elhag, "Reinforcement learning of emerging swarm technologies: A literature review," in *Proc. Future Technol. Conf.*, Jan. 2024, pp. 478–494.
- [19] M. F. G. Teixeira, "Autonomous vehicles coordination through decision-making mechanisms," *Fac. Eng., Univ. Porto, Porto, Portugal*, 2019.

- [20] K. Wan, D. Wu, B. Li, X. Gao, Z. Hu, and D. Chen, "ME-MADDPG: An efficient learning-based motion planning method for multiple agents in complex environments," *Int. J. Intell. Syst.*, vol. 37, no. 3, pp. 2393–2427, Mar. 2022.
- [21] Y. Wang, Z. Cheng, and M. Xiao, "UAVs' formation keeping control based on multi-agent system consensus," *IEEE Access*, vol. 8, pp. 49000–49012, 2020.
- [22] H. Ji and Y. Jin, "Knowledge acquisition of self-organizing systems with deep multiagent reinforcement learning," *J. Comput. Inf. Sci. Eng.*, vol. 22, no. 2, Apr. 2022, Art. no. 021010.
- [23] T. Rashid, M. Samvelyan, C. Schroeder de Witt, G. Farquhar, J. Foerster, and S. Whiteson, "QMIX: Monotonic value function factorisation for deep multi-agent reinforcement learning," 2018, *arXiv:1803.11485*.
- [24] A. Lixandru, "Proximal policy optimization with adaptive exploration," 2024, *arXiv:2405.04664*.
- [25] Y. Guo, J. Liu, R. Yu, P. Hang, and J. Sun, "MAPPO-PIS: A multi-agent proximal policy optimization method with prior intent sharing for CAVs' cooperative decision-making," 2024, *arXiv:2408.06656*.
- [26] J. Liu, Z. Lian, J. Shi, L. Dong, and C. Sun, "Intermittent fixed-time fuzzy consensus of nonlinear multiagent systems with unknown control directions and event-based communication," *IEEE Trans. Fuzzy Syst.*, vol. 32, no. 12, pp. 6917–6928, Dec. 2024.
- [27] S. T. Langlois, O. Akoroda, E. Carrillo, J. W. Herrmann, S. Azarm, H. Xu, and M. Otte, "Meta-reasoning structures, problems, and modes for multi-agent systems: A survey," *IEEE Access*, vol. 8, pp. 183080–183089, 2020.
- [28] Z. Wang, R. Wan, X. Gui, and G. Zhou, "Deep reinforcement learning of cooperative control with four robotic agents by MADDPG," in *Proc. Int. Conf. Comput. Eng. Intell. Control (ICCEIC)*, Nov. 2020, pp. 287–290.
- [29] H. Ji and Y. Jin, "Designing self-organizing systems with deep multi-agent reinforcement learning," in *Proc. Int. Design Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, Aug. 2019, p. V01AT02A011.
- [30] J. Cao, Z. Ming, J. K. Allen, and F. Mistree, "On how a self-organizing system produces collective behavior," in *Proc. Int. Des. Eng. Tech. Conf. Comput. Inf. Eng. Conf.*, Aug. 2023, p. V03BT03A062.
- [31] R. Kassab, A. Destounis, D. Tsilimantou, and M. Debbah, "Multi-agent deep stochastic policy gradient for event based dynamic spectrum access," in *Proc. IEEE 31st Annu. Int. Symp. Pers., Indoor Mobile Radio Commun.*, Aug. 2020, pp. 1–6.
- [32] W. Li et al., "Robust bipartite tracking consensus of multi-agent systems via neural network combined with extended high-gain observer," *ISA Trans.*, vol. 136, pp. 31–45, May 2023.
- [33] Q. Xu, E. Wehrle, and H. Baier, "Knowledge-based surrogate modeling in engineering design optimization," in *Surrogate-Based Modeling and Optimization Applications in Engineering*, Jan. 2013, pp. 313–336.
- [34] A. I. J. Forrester, A. S. Ester, and A. J. Ke, *Engineering Design Via Surrogate Modelling—A Practical Guide*. Hoboken, NY, USA: Wiley, 2008.
- [35] F. Li, X. Cai, L. Gao, and W. Shen, "A surrogate-assisted multiswarm optimization algorithm for high-dimensional computationally expensive problems," *IEEE Trans. Cybern.*, vol. 51, no. 3, pp. 1390–1402, Mar. 2021.
- [36] P. Hespanhol and A. Aswani, "Surrogate optimal control for strategic multi-agent systems," in *Proc. IEEE 58th Conf. Decis. Control (CDC)*, Dec. 2019, pp. 2802–2807.
- [37] G. G. Wang and S. Shan, "Review of metamodeling techniques in support of engineering design optimization," *J. Mech. Design*, vol. 129, no. 4, pp. 370–380, Apr. 2007.
- [38] U. Khurana, H. Samulowitz, and D. S. Turaga, "Feature engineering for predictive modeling using reinforcement learning," in *Proc. AAAI Conf. Artif. Intell.*, Jan. 2017, pp. 1–16.
- [39] S. Salamatin, W. Huleihel, A. Beirami, A. Cohen, and M. Médard, "Why botnets work: Distributed brute-force attacks need no synchronization," *IEEE Trans. Inf. Forensics Security*, vol. 14, no. 9, pp. 2288–2299, Sep. 2019.
- [40] F. Li, J. Qin, and Y. Kang, "Multi-agent system based distributed pattern search algorithm for non-convex economic load dispatch in smart grid," *IEEE Trans. Power Syst.*, vol. 34, no. 3, pp. 1993–2102, May 2019.
- [41] I. Pan and S. Das, "Kriging based surrogate modeling for fractional order control of microgrids," *IEEE Trans. Smart Grid*, vol. 6, no. 1, pp. 36–44, Jan. 2015.
- [42] C. Wang, A. Wang, S. Chen, G. Zhang, and B. Zhu, "Optimal operation of microgrids based on a radial basis function metamodel," *IEEE Syst. J.*, vol. 16, no. 3, pp. 4756–4767, Sep. 2022.
- [43] G. Hinton, N. Srivastava, and K. Swersky, "Neural networks for machine learning lecture 6a overview of mini-batch gradient descent," *Cited*, vol. 14, no. 8, p. 2, 2012.
- [44] S. Zhang and R. S. Sutton, "A deeper look at experience replay," 2017, *arXiv:1712.01275*.
- [45] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," 2014, *arXiv:1412.6980*.
- [46] G. K. Dziugaite, D. M. Roy, and Z. Ghahramani, "Training generative neural networks via maximum mean discrepancy optimization," in *Proc. 31st Conf. Uncertainty Artif. Intell.* Amsterdam, The Netherlands: AUAI Press, Jul. 2015, pp. 258–267.
- [47] J. Jiang and Z. Lu, "Adaptive learning rates for multi-agent reinforcement learning," Tech. Rep., 2023.
- [48] S. Omidshafiei, D. K. Kim, M. Liu, G. Tesauro, M. Riemer, C. Amato, M. Campbell, and J. P. How, "Learning to teach in cooperative multiagent reinforcement learning," in *Proc. 33rd AAAI Conf. Artif. Intell. 31st Innov. Appl. Artif. Intell. Conf.; 9th AAAI Symp. Educ. Adv. Artif. Intell.*, vol. 33, Honolulu, HI, USA, Jul. 2019, pp. 6128–6136.
- [49] D. Liu, F. Ren, J. Yan, G. Su, W. Gu, and S. Kato, "Scaling up multi-agent reinforcement learning: An extensive survey on scalability issues," *IEEE Access*, vol. 12, pp. 94610–94631, 2024.
- [50] J. Rane, Ö. Kaya, S. K. Mallick, and N. L. Rane, "Scalable and adaptive deep learning algorithms for large-scale machine learning systems," *Future Res. Opportunities Artif. Intell. Ind.*, vol. 5, pp. 2–40, Oct. 2024.
- [51] S. Gu, L. Yang, Y. Du, G. Chen, F. Walter, J. Wang, and A. Knoll, "A review of safe reinforcement learning: Methods, theory and applications," 2022, *arXiv:2205.10330*.
- [52] K. Li, S. Wu, and Y. Wang, "Social network evolution of multiagent groups in task allocation," *IEEE Trans. Computat. Social Syst.*, vol. 10, no. 5, pp. 2773–2785, Oct. 2023.
- [53] R. Lowe, Y. Wu, A. Tamar, J. Harb, P. Abbeel, and I. Mordatch, "Multi-agent actor-critic for mixed cooperative-competitive environments," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 30, Jan. 2017, pp. 1–16.
- [54] R. Zheng, S. Dou, S. Gao, Y. Hua, W. Shen, B. Wang, Y. Liu, S. Jin, Y. Zhou, L. Xiong, L. Chen, Z. Xi, N. Xu, W. Lai, M. Zhu, H. Huang, T. Gui, Q. Zhang, and X. Huang, "Delve into PPO: Implementation matters for stable RLHF," in *Proc. NeurIPS Workshop Instruct. Tuning Instruct. Following*, 2023, pp. 1–34.
- [55] G. Qu, C. Yu, S. Low, and A. Wierman, "Combining model-based and model-free methods for nonlinear control: A provably convergent policy gradient approach," 2020, *arXiv:2006.07476*.
- [56] M. Nauman and M. Cygan, "On all-action policy gradients," in *Proc. Deep Reinforcement Learn. Workshop NeurIPS*, 2022, pp. 1–16.
- [57] J. Yang, I. Borovikov, and H. Zha, "Hierarchical cooperative multi-agent reinforcement learning with skill discovery," in *Proc. 19th Int. Conf. Auto. Agents MultiAgent Syst.*, Auckland, New Zealand, Jan. 2019, pp. 1566–1574.
- [58] C. Yu, A. Velu, E. Vinitsky, Y. Wang, A. M. Bayen, and Y. Wu, "The surprising effectiveness of PPO in cooperative, multi-agent games," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2021, pp. 24611–24624.
- [59] J. Walker, E. Vértés, Y. Li, G. Dulac-Arnold, A. Anand, T. Weber, and J. B. Hamrick, "Investigating the role of model-based learning in exploration and transfer," in *Proc. Int. Conf. Mach. Learn.*, Jan. 2023, pp. 1–18.
- [60] G. Papoudakis, F. Christianos, L. Schäfer, and S. V. Albrecht, "Benchmarking multi-agent deep reinforcement learning algorithms in cooperative tasks," 2020, *arXiv:2006.07869*.
- [61] W. Huang, K. Li, K. Shao, T. Zhou, M. Taylor, J. Luo, D. Wang, H. Mao, J. Hao, J. Wang, and X. Deng, "Multiagent Q-learning with sub-team coordination," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 35, 2022, pp. 29427–29439.
- [62] Y. Guan, Y. Ren, S. E. Li, Q. Sun, L. Luo, and K. Li, "Centralized cooperation for connected and automated vehicles at intersections by proximal policy optimization," *IEEE Trans. Veh. Technol.*, vol. 69, no. 11, pp. 12597–12608, Nov. 2020.
- [63] S. Herremans, J. D. Hoog, S. Vanneste, D. Balemans, A. Anwar, S. Mercelis, and P. Hellinckx, "Autonomous navigation using model-based reinforcement learning," in *Proc. Int. Conf. P2P, Parallel, Grid, Cloud Internet Comput.* Cham, Switzerland: Springer, Oct. 2022, pp. 268–277.
- [64] L. Jia, B. Su, D. Xu, Y. Wang, J. Fang, and J. Wang, "Policy optimization algorithm with activation likelihood-ratio for multi-agent reinforcement learning," *Neural Process. Lett.*, vol. 56, no. 6, p. 247, Nov. 2024.

- [65] W. Guo, G. Liu, Z. Zhou, L. Wang, and J. Wang, "Enhancing the robustness of QMIX against state-adversarial attacks," *Neurocomputing*, vol. 572, Mar. 2024, Art. no. 127191.
- [66] H. Zhang, H. Chen, C. Xiao, B. Li, M. Liu, D. S. Boning, and C. Hsieh, "Robust deep reinforcement learning against adversarial perturbations on state observations," in *Proc. Adv. Neural Inf. Process. Syst.*, Jan. 2020, pp. 21024–21037.
- [67] Y. Pu, Y. Niu, Z. Yang, J. Ren, H. Li, and Y. Liu, "UniZero: Generalized and efficient planning with scalable latent world models," 2024, *arXiv:2406.10667*.
- [68] D. Heik, A. Bohm, F. Bahrpeyma, and D. Reichelt, "Application of inhomogeneous QMIX in various architectures to solve dynamic scheduling in manufacturing environments," in *Proc. IEEE 22nd Int. Conf. Ind. Informat. (INDIN)*, Aug. 2024, pp. 1–8.
- [69] A. D. Rio, D. Jimenez, and J. Serrano, "Comparative analysis of A3C and PPO algorithms in reinforcement learning: A survey on general environments," *IEEE Access*, vol. 12, pp. 146795–146806, 2024.
- [70] J. A. de Vries, K. S. Voskuil, T. M. Moerland, and A. Plaat, "Visualizing MuZero models," 2021, *arXiv:2102.12924*.
- [71] A. Jankovic, G. Chaudhary, and F. Goia, "Designing the design of experiments (DOE)—An investigation on the influence of different factorial designs on the characterization of complex systems," *Energy Buildings*, vol. 250, Nov. 2021, Art. no. 111298.
- [72] Y.-S. Ong, P. B. Nair, A. J. Keane, and K. W. Wong, "Surrogate-assisted evolutionary optimization frameworks for high-fidelity engineering design problems," *Stud. Fuzziness Soft Comput. Ser.*, vol. 167, pp. 307–331, Jan. 2004.
- [73] Z. Zhang and D. Wang, "Adaptive individual Q-learning—A multiagent reinforcement learning method for coordination optimization," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 36, no. 4, pp. 1–12, Jan. 2024.
- [74] Z. Wang and B. Liang, "Multi-agent spatial-temporal transformer for traffic signal control," in *Proc. 36th Chin. Control Decis. Conf. (CCDC)*, May 2024, pp. 856–862.
- [75] Y. Wang and F. Wu, "Multi-agent deep reinforcement learning with adaptive policies," 2019, *arXiv:1912.00949*.



RAZA MUHAMMAD KAZIM was born in Pakistan. He received the bachelor's degree in mechanical engineering from the University of Engineering and Technology, Taxila, in 2013, and the master's degree in mechanical design engineering from the University of Engineering and Technology, Lahore, in 2017. He is currently pursuing the Ph.D. degree with Beijing Institute of Technology (BIT), China, supported by the scholarship from China Scholarship Council. His research focuses on designing and developing multi-agent systems, with additional interests in addressing multiscale challenges in advanced manufacturing and production system engineering.



GUOXIN WANG received the B.S. and M.S. degrees from Lanzhou Jiaotong University, in 2001 and 2004, respectively, and the Ph.D. degree from Beijing Institute of Technology, China, in 2007. He was a Visiting Scholar with The University of Oklahoma, USA, from 2014 to 2015. He is currently an Associate Professor with Beijing Institute of Technology. His current research interests include reconfigurable manufacturing systems, intelligent design, and knowledge engineering.



ZHENJUN MING received the bachelor's degree in industrial engineering and the Ph.D. degree in mechanical engineering from Beijing Institute of Technology, in 2011 and 2017, respectively. He spent 18 months (from August 2014 to April 2016) as a Visiting Ph.D. Student with the Systems Realization Laboratory, The University of Oklahoma. He is currently an Associate Professor with the Institute for Industrial Engineering, Beijing Institute of Technology. He has published more than ten peer-reviewed research papers. His research interests include decision-based design, decision support systems, knowledge-based engineering, and knowledge-based systems.



JINHUI CAO is currently pursuing the bachelor's degree in industrial engineering with the School of Mechanical Engineering, Beijing Institute of Technology (BIT). He plans to continue his academic journey by undertaking a master's degree with BIT. His research focuses on adaptive architecture design and scalability optimization in multi-agent collaborative systems, with particular emphasis on frontier technologies in swarm intelligence and distributed decision-making. His academic interests include the intersection of autonomous systems engineering and emergent behaviors in complex adaptive systems.



JANET K. ALLEN (Member, IEEE) received the S.B. degree in life sciences from Massachusetts Institute of Technology and the Ph.D. degree in biophysics from the University of California. She has held the John and Mary Moore Chair. She has been a Professor of industrial and systems engineering with The University of Oklahoma, since August 2009. She joined the university after serving with Georgia Tech. Her research interests include the design of evolving cyber-physical systems, simulation-based design, managing uncertainty in design, sustainability, and design pedagogy.



FARROKH MISTREE received the Ph.D. degree in engineering from the University of California at Berkeley. He is a Professor and a L. A. Comp Chair with The University of Oklahoma. His research focuses on treating public policy as an evolving cyber-physical-social systems and designing as a service for small and medium-sized enterprises and rural development. He has made significant contributions to these fields and has published extensively in peer-reviewed journals. He is a fellow of ASME; an Associate Fellow of AIAA; a Life Member of the Honor Society of Phi Kappa Phi; and a member of ASEE, RINA, and SNAME.

...