Frontiers of Information Technology & Electronic Engineering
www.jzus.zju.edu.cn; engineering.cae.cn; www.springerlink.com
ISSN 2095-9184 (print); ISSN 2095-9230 (online)
E-mail: jzus@zju.edu.cn

Multi-agent reinforcement learning behavioral control for nonlinear second-order systems*#

Zhenyi ZHANG[1,2], Jie HUANG†‡[1,2], Congjie PAN[1,2]

[1]*College of Electrical Engineering and Automation, Fuzhou University, Fuzhou 350108, China*

[2]*5G+ Industrial Internet Institute, Fuzhou University, Fuzhou 350108, China*

†E-mail: jie.huang@fzu.edu.cn

Received June 1, 2023; Revision accepted Nov. 16, 2023; Crosschecked May 23, 2024

Abstract: Reinforcement learning behavioral control (RLBC) is limited to an individual agent without any swarm mission, because it models the behavior priority learning as a Markov decision process. In this paper, a novel multi-agent reinforcement learning behavioral control (MARLBC) method is proposed to overcome such limitations by implementing joint learning. Specifically, a multi-agent reinforcement learning mission supervisor (MARLMS) is designed for a group of nonlinear second-order systems to assign the behavior priorities at the decision layer. Through modeling behavior priority switching as a cooperative Markov game, the MARLMS learns an optimal joint behavior priority to reduce dependence on human intelligence and high-performance computing hardware. At the control layer, a group of second-order reinforcement learning controllers are designed to learn the optimal control policies to track position and velocity signals simultaneously. In particular, input saturation constraints are strictly implemented via designing a group of adaptive compensators. Numerical simulation results show that the proposed MARLBC has a lower switching frequency and control cost than finite-time and fixed-time behavioral control and RLBC methods.

Key words: Reinforcement learning; Behavioral control; Second-order systems; Mission supervisor
https://doi.org/10.1631/FITEE.2300394 **CLC number:** TP18

1 Introduction

Multi-agent systems have been widely used in civilian and military areas in the past decade, since they complete complex missions through swarm coordination (Cao YC et al., 2013; Liu Y et al., 2022). Second-order systems are usually used to characterize their dynamics, because most robot systems satisfy Newton's second theorem (Yao DY et al., 2020). Although the second-order systems generally exhibit better mission performance than individual agents, they sometimes encounter conflicts in multi-mission scenarios (Garattoni and Birattari, 2018). For example, second-order systems may form a rigid formation, but encounter obstacles along the path. Finding solutions to such conflicts is one of the key challenges in managing second-order systems.

Behavioral control was initially proposed by Brooks (1986), providing an effective solution by modeling and fusing behaviors. Originally, behavioral control is a layered and competitive architecture, in which lower-layer behaviors are allowed to execute only after all higher-layer behaviors have been completed (Brooks, 1991). However, since only the highest-layer behaviors are executed at each sampling time, mission execution efficiency is reduced, and system redundancy is wasted. To address this issue, a motor schema-based behavioral control system with a cooperative architecture was introduced

‡ Corresponding author

* Project supported by the National Natural Science Foundation of China (No. 92367109)

Electronic supplementary materials: The online version of this article (https://doi.org/10.1631/FITEE.2300394) contains supplementary materials, which are available to authorized users

ORCID: Zhenyi ZHANG, https://orcid.org/0000-0001-8581-879X; Jie HUANG, https://orcid.org/0000-0001-7346-5034

(Arkin, 1989; Balch and Arkin, 1998), where all behavior commands were summed to output a composite command. Nevertheless, it was challenging to select layers or weights due to each behavior having a unique physical meaning, and none of the behaviors was fully completed. To tackle the layer or weight selection problem, fuzzy behavioral control offered an effective solution by leveraging fuzzy logic technology, but it still struggled to balance complete execution and execution efficiency (Vadakkepat et al., 2004). For this purpose, Antonelli and Chiaverini (2006) proposed a novel null-space-based behavioral control (NSBC) framework, in which the highest-priority behavior must be completed in conjunction with simultaneous execution of one part of the lower-priority behaviors taking place on the null space of the highest-priority behavior. On one hand, behavior priorities are fixed or preset, which results in reduction in the dynamic performance of the mission. On the other hand, once the behavior priorities are switched, it is difficult to track the reference commands, which reduces the tracking control accuracy.

Since the NSBC method depends on a mission supervisor to assign behavior priorities, several mission supervisor design methods have been proposed to improve the dynamic performance of the mission, such as finite state automata mission supervisor (FSAMS) (Marino et al., 2013), fuzzy mission supervisor (FMS) (Marino et al., 2009), and model predictive control mission supervisor (MPCMS) (Chen YT et al., 2020). Nevertheless, both FSAMS and FMS rely on human intelligence to design state transition rules and fuzzy logic rules, respectively. The MPCMS relies on high-performance computing and storage units to meet online-computing and real-time requirements. In addition, the NSBC method requires a group of behavioral controllers to track reference commands in real time. In particular, the decision of behavior priorities and tracking of reference commands occur step by step, forming an integrated structure of decision and control. Most existing behavioral controllers primarily focus on improving the accuracy of reference command tracking (Schlanbusch et al., 2011; Ahmad et al., 2014; Ott et al., 2015; Huang et al., 2017; Santos et al., 2017; Wang WJ et al., 2021; Yao P et al., 2022). Specifically, some works proved that the tracking errors of reference commands can achieve finite-time or fixed-time convergence (Zhou et al., 2015, 2022; Chen J et al., 2016; Huang et al., 2019, 2022b; Zheng et al., 2023). Although these behavioral controllers have achieved excellent tracking control performance, control inputs and costs become excessively large when behavior priorities are switched. More recently, reinforcement learning behavioral control (RLBC) has been proposed to improve the mission supervisor and behavioral controller via trial-and-error learning (Zhang et al., 2022). Specifically, a reinforcement learning mission supervisor (RLMS) has been proposed to avoid artificial design rules and reduce online calculations. Moreover, a reinforcement learning controller (RLC) has been proposed to learn the optimal control policy and balance the control consumption and resource. Nevertheless, RLBC has the following shortcomings: (1) RLMS can be used for only single-agent applications, since it is modeled through Markov decision processes. As a result, any cooperative behavior cannot be implemented, which greatly limits the swarm intelligence. (2) Although RLBC guarantees the convergence of position error signals, it is not enough for second-order systems. Generally, second-order systems require both position and velocity error signals to converge. (3) RLC does not have input saturation constraints, and thus the control input may exceed the physical limit of the actuator. Particularly, if behavior priorities are switched, the problem of excessive control input may be aggravated.

Motivated by the concerns discussed above, in this paper we propose a novel multi-agent reinforcement learning behavioral control (MARLBC) method to overcome the three above-mentioned defects. The contributions of this paper are summarized as follows:

1. A multi-agent reinforcement learning mission supervisor (MARLMS) is proposed to learn the optimal joint behavior priority policy. The behavior priority switching problem is first modeled as a cooperative Markov game. The MARLMS allows cooperative behaviors to be added to training, learning, and execution, so that the individual limitation of the RLMS is broken. Through learning a joint behavior priority policy, the switching frequency of behavioral control is reduced significantly.

2. A group of second-order reinforcement learning controllers (SORLCs) with the identifier–actor–critic structure are developed to learn the optimal control policies. The Hamilton–Jacobi–Bellman

(HJB) equations of second-order systems are constructed by combining the position and velocity errors. During the mission execution, the control performance and consumption are always balanced. As a result, the control cost of behavioral control is reduced significantly.

3. Input saturation constraints are implemented to avoid second-order systems exceeding the physical limitations. A group of adaptive compensators are designed to maintain the optimal control performance and counteract the saturation effect in real time. The mission, tracking, and weight error signals are all proved to be semi-globally uniformly ultimately bounded (SGUUB).

2 Modeling and problem statement

2.1 Modeling

Consider a group of N ($N \geq 2$) nonlinear second-order systems, in which the dynamic model of the i^{th} ($i = 1, 2, \ldots, N$) agent is described as

$$\begin{cases} \dot{\boldsymbol{p}}_i = \boldsymbol{v}_i, \\ \dot{\boldsymbol{v}}_i = \boldsymbol{u}_i + \boldsymbol{f}_i(\boldsymbol{\chi}_i), \end{cases} \tag{1}$$

where $\boldsymbol{p}_i \in \mathbb{R}^n$ and $\boldsymbol{v}_i \in \mathbb{R}^n$ are the position and velocity respectively, $\boldsymbol{u}_i \in \mathbb{R}^n$ is the control input, $\boldsymbol{f}_i(\boldsymbol{\chi}_i) : \mathbb{R}^{2n} \to \mathbb{R}^n$ is the unknown nonlinear dynamics part with $\boldsymbol{f}_i(\boldsymbol{0}) = \boldsymbol{0}_n$, and $\boldsymbol{\chi}_i = [\boldsymbol{p}_i^{\mathrm{T}}, \boldsymbol{v}_i^{\mathrm{T}}]^{\mathrm{T}}$ is the generalized state.

The control input $\boldsymbol{u}_i = [u_{i,1}, u_{i,2}, \ldots, u_{i,n}]^{\mathrm{T}}$ is limited by the input saturation constraints as (Cao SJ et al., 2023)

$$u_{i,\iota} = \begin{cases} u_{\lim,i}, & u_{i,\iota} \geq u_{\lim,i}, \\ u_{i,\iota}, & -u_{\lim,i} < u_{i,\iota} < u_{\lim,i}, \\ -u_{\lim,i}, & u_{i,\iota} \leq -u_{\lim,i}, \end{cases} \tag{2}$$

where $\iota = 1, 2, \ldots, n$ and $u_{\lim,i} > 0$ is the known limitation.

Thereafter, we divide the control input as

$$\boldsymbol{u}_i = \boldsymbol{u}_{0,i} + \boldsymbol{u}_{\Delta,i}, \tag{3}$$

where $\boldsymbol{u}_{0,i} = [u_{0,i,1}, u_{0,i,2}, \ldots, u_{0,i,n}]^{\mathrm{T}} \in \mathbb{R}^n$ is the nominal part and $\boldsymbol{u}_{\Delta,i} = [u_{\Delta,i,1}, u_{\Delta,i,2}, \ldots, u_{\Delta,i,n}]^{\mathrm{T}} \in \mathbb{R}^n$ is the compensation part with

$$u_{\Delta,i,\iota} = \begin{cases} u_{\lim,i} - u_{0,i,\iota}, & u_{0,i,\iota} \geq u_{\lim,i}, \\ u_{0,i,\iota}, & -u_{\lim,i} < u_{0,i,\iota} < u_{\lim,i}, \\ -u_{\lim,i} - u_{0,i,\iota}, & u_{0,i,\iota} \leq -u_{\lim,i}. \end{cases} \tag{4}$$

Assumption 1 The unknown nonlinear dynamics part $\boldsymbol{f}_i(\boldsymbol{\chi}_i)$ is assumed Lipschitz continuous on the set containing the origin, and it is bounded with $|\boldsymbol{f}_i(\boldsymbol{\chi}_i)| \leq \epsilon_{\mathrm{f}}$, where ϵ_{f} is a known positive constant.

Assumption 2 The second-order systems work in a scenario in which all obstacles are static and fixed.

Assumption 3 The second-order systems are not always input-saturated, and the compensation part $\boldsymbol{u}_{\Delta,i}$ is bounded with $|\boldsymbol{u}_{\Delta,i}| \leq \varepsilon_{\Delta}$, where ε_{Δ} is a known positive constant.

Property 1 (Wen et al., 2021) Given two vectors $\boldsymbol{\eta}_1,\ \boldsymbol{\eta}_2 \in \mathbb{R}^n$, there is $\mathrm{tr}\{\boldsymbol{\eta}_1\boldsymbol{\eta}_2^{\mathrm{T}}\} = \boldsymbol{\eta}_1^{\mathrm{T}}\boldsymbol{\eta}_2 = \boldsymbol{\eta}_2^{\mathrm{T}}\boldsymbol{\eta}_1$, where tr denotes the trace.

Property 2 Given two vectors $\boldsymbol{\eta}_1,\ \boldsymbol{\eta}_2 \in \mathbb{R}^n$, there is $(\boldsymbol{\eta}_1^{\mathrm{T}}\boldsymbol{\eta}_2)^2 \leq \|\boldsymbol{\eta}_1\|^2\|\boldsymbol{\eta}_2\|^2$.

Property 3 Given two real numbers $\varsigma_1, \varsigma_2 \in \mathbb{R}$, there is $\varsigma_1\varsigma_2 \leq \frac{1}{2}\varsigma_1^2 + \frac{1}{2}\varsigma_2^2$.

Lemma 1 (Wen et al., 2018) Given a continuous function $V(t) \in \mathbb{R}$ satisfying $\dot{V}(t) \leq -\vartheta_1 V(t) + \vartheta_2$, where $\vartheta_1, \vartheta_2 \in \mathbb{R}$ are positive constants, we have the following result:

$$V(t) \leq \mathrm{e}^{-\vartheta_1 t}V(0) + \frac{\vartheta_2}{\vartheta_1}\left(1 - \mathrm{e}^{-\vartheta_1 t}\right). \tag{5}$$

2.2 Control objective

The control objective is to learn a joint behavior priority and a group of tracking control policies for the second-order systems (1) to complete some conflicting missions in a work scenario with Assumption 1, such that (1) agents form a desired formation while avoiding obstacles near the trajectories; (2) the behavior priorities of agents are dynamically and intelligently switched; (3) the control cost is minimized with the input saturation constraints, and all signals are bounded theoretically.

3 MARLBC design

In this section, a novel MARLBC method is proposed for a group of nonlinear second-order systems to switch the optimal behavior priorities and implement the optimal control inputs via trial-and-error learning. Fig. 1 portrays the block diagram of the proposed MARLBC method, which is a two-layer decision-control integration structure. In the upper layer, the NSBC framework is used to generate the reference commands, and it embeds an MARLMS to achieve intelligent switching of the

behavior priorities via the learning of a joint behavior priority policy. In the lower layer, a group of SORLCs are designed to implement the optimal tracking control with the input saturation constraints via minimizing the control cost and using the adaptive compensator. Particularly, bearing in mind the objective that agents must be enabled to interact with their environment at each time step, the priority-wise decision formulation and tracking control of the proposed MARLBC method are executed in a step-by-step manner.

3.1 Elementary and composite behavior design

First, a mission variable $\boldsymbol{\rho}_{i,j} \in \mathbb{R}^m$ $(m \leq n)$ is used to model the j^{th} elementary behavior as

$$\boldsymbol{\rho}_{i,j} = \boldsymbol{g}_{i,j}(\boldsymbol{p}_i), \tag{6}$$

where $\boldsymbol{g}_{i,j}(\cdot) : \mathbb{R}^n \to \mathbb{R}^m$ is the mission function and $j = 1, 2, \ldots, M$ with M being the number of behaviors.

Then, the differential of Eq. (6) is yielded as

$$\dot{\boldsymbol{\rho}}_{i,j} = \frac{\partial \boldsymbol{g}_{i,j}(\boldsymbol{p}_i)}{\partial \boldsymbol{p}_i}\dot{\boldsymbol{p}}_i = \boldsymbol{J}_{i,j}\boldsymbol{v}_i, \tag{7}$$

where $\boldsymbol{J}_{i,j} \in \mathbb{R}^{m\times n}$ is the mission Jacobian matrix.

Finally, the reference velocity command of the j^{th} elementary behavior of the i^{th} agent is yielded via using the closed-loop inverse kinematics algorithm as (Antonelli and Chiaverini, 2006)

$$\boldsymbol{v}_i = \boldsymbol{J}_{i,j}^{\dagger}(\dot{\boldsymbol{\rho}}_{\mathrm{d},i,j} + \boldsymbol{\Lambda}_{i,j}\tilde{\boldsymbol{\rho}}_{i,j}), \tag{8}$$

where $\boldsymbol{J}_{i,j}^{\dagger} = \boldsymbol{J}_{i,j}^{\mathrm{T}}(\boldsymbol{J}_{i,j}\boldsymbol{J}_{i,j}^{\mathrm{T}})^{-1} \in \mathbb{R}^{n\times m}$ is the right pseudo-inverse matrix of $\boldsymbol{J}_{i,j}$, $\boldsymbol{\rho}_{\mathrm{d},i,j} \in \mathbb{R}^m$ is the desired mission, $\boldsymbol{\Lambda}_{i,j} \in \mathbb{R}^{m\times m}$ is the mission gain, and $\tilde{\boldsymbol{\rho}}_{i,j} = \boldsymbol{\rho}_{\mathrm{d},i,j} - \boldsymbol{\rho}_{i,j} \in \mathbb{R}^m$ is the mission error.

Without loss of generality, a local behavior and two cooperative behaviors are given as follows.

3.1.1 Obstacle avoidance (OA) behavior

The OA behavior is a local behavior, which is designed to ensure that agents avoid obstacles near the path:

$$\begin{cases} \rho_{\mathrm{OA},i} = \min\{d_i^{\mathrm{o}}\}, \\ \rho_{\mathrm{OA,d},i} = d_{\mathrm{OA}}, \\ \boldsymbol{J}_{\mathrm{OA},i} = \boldsymbol{\Upsilon}_{\mathrm{OA},i}^{\mathrm{T}}, \end{cases} \tag{9}$$

where $\rho_{\mathrm{OA},i} \in \mathbb{R}$, $\rho_{\mathrm{OA,d},i} \in \mathbb{R}$, and $\boldsymbol{J}_{\mathrm{OA},i} \in \mathbb{R}^{1\times n}$ are the mission function, desired mission function, and mission Jacobian matrix of the OA behavior, respectively; $\min\{d_i^{\mathrm{o}}\} \in \mathbb{R}$ is the minimum distance between the i^{th} agent and obstacles; $d_{\mathrm{OA}} \in \mathbb{R}$ is the safe distance; and $\boldsymbol{\Upsilon}_{\mathrm{OA},i} = \dfrac{\boldsymbol{p}_{i,\min}^{\mathrm{o}}}{\min\{d_i^{\mathrm{o}}\}} \in \mathbb{R}^n$ with $\boldsymbol{p}_{i,\min}^{\mathrm{o}} \in \mathbb{R}^n$ being the position difference vector corresponding to the minimum distance between the i^{th} robot and the obstacles.

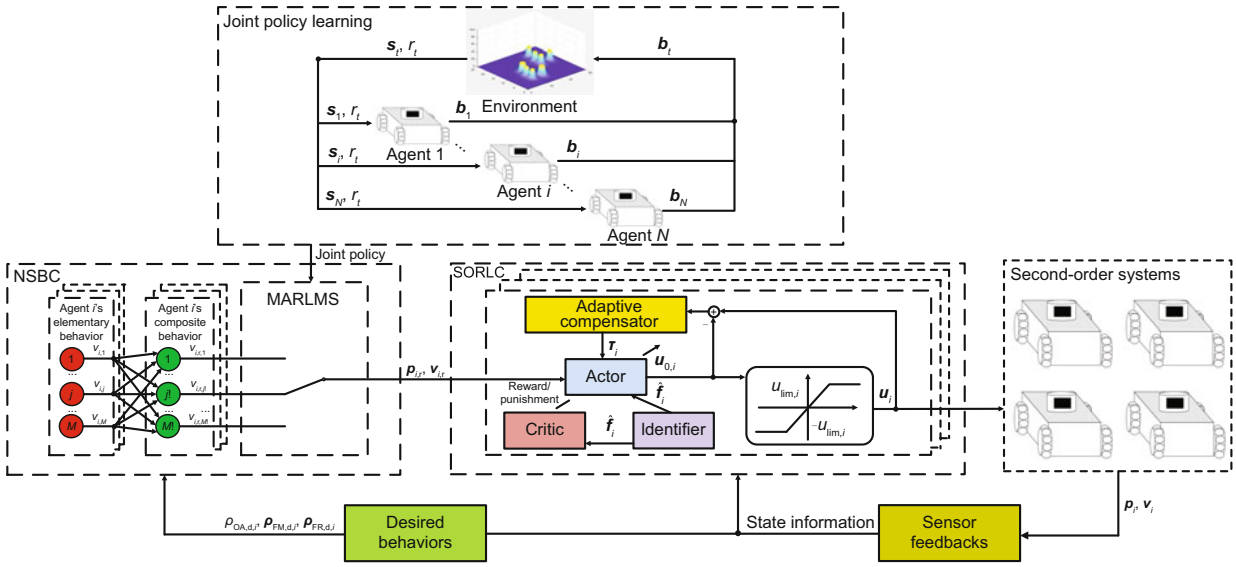

Fig. 1 Principle block diagram of the proposed MARLBC method

3.1.2 Formation maintenance (FM) behavior

The FM behavior is a cooperative behavior, which is designed to ensure that agents form and maintain a desired formation:

$$\begin{cases} \boldsymbol{\rho}_{\mathrm{FM},i} = \boldsymbol{p}_i - \boldsymbol{p}_{\mathrm{c}} - \boldsymbol{p}_i^{\mathrm{c}}, \\ \boldsymbol{\rho}_{\mathrm{FM,d},i} = \boldsymbol{p}_{\mathrm{c,d}} - \boldsymbol{p}_{\mathrm{c}}, \\ \boldsymbol{J}_{\mathrm{FM},i} = \boldsymbol{I}_n, \end{cases} \tag{10}$$

where $\boldsymbol{\rho}_{\mathrm{FM},i} \in \mathbb{R}^n$, $\boldsymbol{\rho}_{\mathrm{FM,d},i} \in \mathbb{R}^n$, and $\boldsymbol{J}_{\mathrm{FM},i} \in \mathbb{R}^{n\times n}$ are the mission function, desired mission function, and mission Jacobian matrix of the FM behavior, respectively; $\boldsymbol{p}_{\mathrm{c}} = \dfrac{1}{N}\sum_{i=1}^{N} \boldsymbol{p}_i$ is the formation centroid; $\boldsymbol{p}_{\mathrm{c,d}}$ is the desired centroid trajectory; $\boldsymbol{p}_i^{\mathrm{c}}$ is the relative position required for agent i and the formation centroid to form the formation; $\boldsymbol{I}_n$ is an identity matrix.

3.1.3 Formation reconstruction (FR) behavior

The FR behavior is a cooperative behavior, which is designed to drive agents to form and maintain a temporary formation:

$$\begin{cases} \boldsymbol{\rho}_{\mathrm{FR},i} = \boldsymbol{p}_i - \boldsymbol{p}_{\mathrm{c}} - \boldsymbol{\Gamma}_{\mathrm{FR},i}\, \boldsymbol{p}_i^{\mathrm{c}}, \\ \boldsymbol{\rho}_{\mathrm{FR,d},i} = \boldsymbol{p}_{\mathrm{c,d}} - \boldsymbol{p}_{\mathrm{c}}, \\ \boldsymbol{J}_{\mathrm{FR},i} = \boldsymbol{I}_n, \end{cases} \tag{11}$$

where $\boldsymbol{\rho}_{\mathrm{FR},i} \in \mathbb{R}^n$, $\boldsymbol{\rho}_{\mathrm{FR,d},i} \in \mathbb{R}^n$, and $\boldsymbol{J}_{\mathrm{FR},i} \in \mathbb{R}^{n\times n}$ are the mission function, desired mission function, and mission Jacobian matrix of the FR behavior, respectively, and $\boldsymbol{\Gamma}_{\mathrm{FR},i} \in \mathbb{R}^{n\times n}$ is the reconstruction matrix.

Although the designs of FM and FR behaviors are similar, their roles are inconsistent. The FM behavior aims to drive agents to form a desired formation. The FR behavior aims to drive agents to reconstruct a temporary formation when the desired formation cannot be achieved; e.g., agents encounter large-range obstacles. Thus, the design of the FR behavior includes a reconstruction matrix, which changes the desired formation to a temporary formation.

Composite behaviors are composed of elementary behaviors with different priorities. Behavior priority functions and hierarchy rules are the same as those adopted in Huang et al. (2022a). If the behavior priorities are assigned at time step t, then the reference commands of composite behaviors are expressed as

$$\begin{cases} \boldsymbol{v}_{i,\mathrm{r},t} = \boldsymbol{v}_{i,t}^1 + \sum_{\hat{j}=2}^{M} \boldsymbol{J}_{i,t}^{1,\hat{j}-1} \boldsymbol{v}_{i,t}^{\hat{j}}, \\ \boldsymbol{J}_{i,t}^{1,\hat{j}-1} = \boldsymbol{I}_3 - \left(\boldsymbol{J}_{i,t}^{1,\hat{j}}\right)^{\dagger} \boldsymbol{J}_{i,t}^{1,\hat{j}}, \\ \boldsymbol{J}_{i,t}^{1,\hat{j}} = \left[\left(\boldsymbol{J}_{i,t}^{1}\right)^{\mathrm{T}}, \left(\boldsymbol{J}_{i,t}^{2}\right)^{\mathrm{T}}, \ldots, \left(\boldsymbol{J}_{i,t}^{\hat{j}}\right)^{\mathrm{T}}\right]^{\mathrm{T}}, \end{cases} \tag{12}$$

where $\hat{j}$ is the value of the behavior priority, and $\boldsymbol{J}_{i,t}^{1,\hat{j}-1} \in \mathbb{R}^{n\times n}$ is the projection on the null space of the augmented Jacobian.

3.2 MARLMS design

We describe the behavior priority switching problem as a cooperative Markov game, where all agents share the same reward (Littman, 1994). We define the joint state set as $S = \{\boldsymbol{s}_t\}$, where $\boldsymbol{s}_t = (\boldsymbol{X}_t, \mathbf{Pr}_t, \mathfrak{G}_t) \in \mathbb{R}^{(2n+1)N+1}$ with $\boldsymbol{X}_t = [\boldsymbol{x}_i] \in \mathbb{R}^{2nN}$ being the joint generalized state, $\mathbf{Pr}_t \in \mathbb{R}^N$ the joint behavior priority identifier, and $\mathfrak{G}_t \in \mathbb{R}$ the formation identifier. We further define the joint behavior set as $B = \{\boldsymbol{b}_t\}$, where $\boldsymbol{b}_t = [v_{i,\mathrm{r},t}] \in \mathbb{R}^{nN}$, and define T_{e}, T_{s}, and $\mathfrak{D}$ as the total number of training episodes, total number of time steps, and experience replay buffer, respectively.

Then, the reward function is designed as

$$r_t = r_1 + r_2, \tag{13}$$

$$r_1 = \begin{cases} -10, & \mathfrak{G}_t = 0, \min\{d_i^{\mathrm{o}}\} \le d_{\mathrm{OA}}, \\ & \exists i = 1,2,\ldots,N, \\ 0, & \mathfrak{G}_t = 0, \min\{d_i^{\mathrm{o}}\} > d_{\mathrm{OA}}, \\ & \forall i = 1,2,\ldots,N, \\ -10, & \mathfrak{G}_t = 1, \min\{d_i^{\mathrm{o}}\} \le d_{\mathrm{OA}}, \\ & \exists i = 1,2,\ldots,N, \\ +5, & \mathfrak{G}_t = 1, \min\{d_i^{\mathrm{o}}\} > d_{\mathrm{OA}}, \\ & \forall i = 1,2,\ldots,N, \\ -10, & \mathfrak{G}_t = 2, \min\{d_i^{\mathrm{o}}\} \le d_{\mathrm{OA}}, \\ & \exists i = 1,2,\ldots,N, \\ +10, & \mathfrak{G}_t = 2, \min\{d_i^{\mathrm{o}}\} > d_{\mathrm{OA}}, \\ & \forall i = 1,2,\ldots,N, \end{cases} \tag{14}$$

$$r_2 = \begin{cases} 0, & \mathbf{Pr}_{t+1} = \mathbf{Pr}_t, \\ -3, & \mathbf{Pr}_{t+1} \ne \mathbf{Pr}_t. \end{cases} \tag{15}$$

Through comparing the relative positions between agents, the formation that has been taken up by the agents can be ascertained. If the relative positions meet the desired formation, then $\mathfrak{G}_t = 2$; if the relative positions meet the temporary formation, then $\mathfrak{G}_t = 1$; otherwise, $\mathfrak{G}_t = 0$. $\mathfrak{G}_t$ is used to define the formation state and receive the formation

reward. The reward function (13) is used to drive agents to learn a joint behavior priority policy that can meet the mission objective, which is composed of two parts; r_1 is designed to achieve the mission objective, and r_2 is designed to reduce the number of behavior priority switches. Specifically, the safety issue of agents is most important; so, as long as there is an agent entering the safe range of obstacles, the team will receive a reward of -10, regardless of whether the formation is formed or not. If agents do not violate safety constraints, then the focus can be on whether the agents form formations. If agents form a desired formation, then the team will receive a reward of $+10$; if agents form a temporary formation, then the team will receive a reward of $+5$; otherwise, the team will not receive any reward. The design of r_2 is simple: if the behavior priority of the previous sampling is different from that of the current sampling, the team will receive a reward of -3; otherwise, the team will not receive any reward.

Remark 1 For reinforcement learning algorithms, the reward function is the only evaluation indicator. In this study, the mission objective is that the second-order systems form a desired formation while avoiding obstacles near the path. Thus, the reward function needs to encourage the second-order systems to form a desired formation and avoid obstacles during the learning process. On one hand, if agents choose a certain joint behavior priority that violates the safety distance, then they will receive a negative reward, and tend to choose other joint behavior priorities in the same joint state. On the other hand, if agents choose a certain joint behavior priority to form a desired formation, then they receive positive rewards, and tend to select such joint behavior priority in the same joint state.

An MARLMS is developed based on the lenient deep Q-network (LDQN) algorithm (Wei and Luke, 2016). The pseudo code of the proposed MARLMS is shown in Algorithm S1 (supplementary materials). Before starting training, T_{e} and T_{s} should be inputted manually. At the initialization phase, all the learning-related components will be initialized, including Q-value $Q(\boldsymbol{s}_t, \boldsymbol{b}_t; \omega_{\mathrm{Q}}, \omega_{\mathrm{V}}, \omega_{\mathrm{B}})$, experience replay buffer $\mathfrak{D}$, $\bar{T}(\phi(\boldsymbol{s}_t))$-greedy exploration, and leniency $\mathfrak{L}(\boldsymbol{s}_t, \boldsymbol{b}_t)$. Specifically, the Q-value is estimated via using the dueling network to improve the estimation accuracy, in which $Q(\boldsymbol{s}_t, \boldsymbol{b}_t; \omega_{\mathrm{Q}}, \omega_{\mathrm{V}}, \omega_{\mathrm{B}}) = V(\boldsymbol{s}_t; \omega_{\mathrm{Q}}, \omega_{\mathrm{V}}) + B(\boldsymbol{s}_t, \boldsymbol{b}_t; \omega_{\mathrm{Q}}, \omega_{\mathrm{B}})$, where $V(\boldsymbol{s}_t; \omega_{\mathrm{Q}}, \omega_{\mathrm{V}})$ is the state-value network and $B(\boldsymbol{s}_t, \boldsymbol{b}_t; \omega_{\mathrm{Q}}, \omega_{\mathrm{B}})$ is the behavior advantage network with ω_{Q}, ω_{V}, and ω_{B} being the main network weights of the Q-value network, state-value network, and behavior advantage network, respectively (Wang ZY et al., 2016). At time step t, agents interact with the off-line training environment; they observe $\boldsymbol{s}_t$ and select $\boldsymbol{b}_t$ based on a $\bar{T}(\phi(\boldsymbol{s}_t))$-greedy policy, resulting in a team reward r_t and a transition to $\boldsymbol{s}_{t+1}$. The $\bar{T}(\phi(\boldsymbol{s}_t))$-greedy policy is that agents select a random $\boldsymbol{b}_t$ with probability $\bar{T}^{\zeta}(\phi(\boldsymbol{s}_t))$ and $\boldsymbol{b}_t = \arg\max_{\boldsymbol{b}} Q(\boldsymbol{s}_t, \boldsymbol{b}_t)$ with probability $1 - \bar{T}^{\zeta}(\phi(\boldsymbol{s}_t))$; ζ is an exponent; $\bar{T}(\phi(\boldsymbol{s}_t)) = \dfrac{1}{\bar{M}} \sum_{j=1}^{\bar{M}} T_t(\phi(\boldsymbol{s}_t), \boldsymbol{b}_j)$, where $T_t(\phi(\boldsymbol{s}_t), \boldsymbol{b}_t)$ is the decay temperature of state-behavior pair $(\boldsymbol{s}_t, \boldsymbol{b}_t)$ and $\bar{M}$ is the number of combinations of different behavior priorities, which usually satisfies $\bar{M} = M!$. Thereafter, the transition $(\boldsymbol{s}_t,\ \boldsymbol{b}_t,\ r_t,\ \boldsymbol{s}_{t+1})$ will be stored into $\mathfrak{D}$, and it is marked by a leniency as

$$\begin{cases} \mathfrak{L}(\boldsymbol{s}_t, \boldsymbol{b}_t) = 1 - \mathrm{e}^{-\kappa_{\mathfrak{L}} T_t(\phi(\boldsymbol{s}_t), \boldsymbol{b}_t)}, \\ T_{t+1}(\phi(\boldsymbol{s}_t), \boldsymbol{b}_t) = \gamma_{\mathfrak{L}} T_t(\phi(\boldsymbol{s}_t), \boldsymbol{b}_t), \\ \gamma_{\mathfrak{L}} = \mathrm{e}^{\rho_\gamma d_\gamma^t}, \end{cases} \tag{16}$$

where $\kappa_{\mathfrak{L}}$ is the leniency moderation factor, $\phi(\cdot)$ is the hash auto-encoding function, $\gamma_{\mathfrak{L}}$ is the discount factor, ρ_γ is the temperature exponent, and d_γ is the decay rate.

To further reduce the overestimate of Q-value, the averaged Q-value framework is introduced to update the Q-value (Anschel et al., 2017). As a result, the iterative form of the Q-value is shown in line 7 of Algorithm S1, where λ is the number of previously learned Q-values. The temporal-difference (TD) error can be calculated as $\delta_t = y_{\boldsymbol{s}_t, \boldsymbol{b}_t} - Q(\boldsymbol{s}_t, \boldsymbol{b}_t; \omega_{\mathrm{Q}}, \omega_{\mathrm{V}}, \omega_{\mathrm{B}})$, where $y_{\boldsymbol{s}_t, \boldsymbol{b}_t}$ is shown in line 8 of Algorithm S1, and ω_{Q}^-, ω_{V}^-, ω_{B}^- are the target network weights of the Q-value network, state-value network, and behavior advantage network, respectively, which are fixed within some time steps. Thereafter, the Q-value updating is determined based on the leniency $\mathfrak{L}(\boldsymbol{s}_t, \boldsymbol{b}_t)$ as

$$\begin{aligned} &Q(\boldsymbol{s}_t, \boldsymbol{b}_t; \omega_{\mathrm{Q}}, \omega_{\mathrm{V}}, \omega_{\mathrm{B}}) \\ &= \begin{cases} Q(\boldsymbol{s}_t, \boldsymbol{b}_t; \omega_{\mathrm{Q}}, \omega_{\mathrm{V}}, \omega_{\mathrm{B}}) + \alpha_t \delta_t, & \delta_t > 0 \text{ or } \vartheta > \mathfrak{L}_t, \\ Q(\boldsymbol{s}_t, \boldsymbol{b}_t; \omega_{\mathrm{Q}}, \omega_{\mathrm{V}}, \omega_{\mathrm{B}}), & \delta_t \le 0 \text{ and } \vartheta \le \mathfrak{L}_t, \end{cases} \end{aligned} \tag{17}$$

where $\alpha_t \in (0, 1)$ is the learning rate, $\vartheta \sim U(0, 1)$ represents a random variable, $\mathfrak{L}_t$ is short for $\mathfrak{L}(\boldsymbol{s}_t, \boldsymbol{b}_t)$,

and the updating of network weight is as shown in line 9 of Algorithm S1.

The off-line training stops until all episodes are accomplished. Finally, the learned joint policy guides agents to select the optimal joint behavior priority in real time. Once the behavior priorities are determined, the reference velocity $\boldsymbol{v}_{i,\mathrm{r}}$ and position $\boldsymbol{p}_{i,\mathrm{r}}$ can be calculated using Eq. (12).

Remark 2 In this study, we focus on the means for designing a mission supervisor for deciding the behavior priorities based on joint learning. Although the MARLMS uses the lenient DQN algorithm, other joint learning algorithms are also effective, e.g., hysteretic DQN. Since the joint behavior priority is a discrete policy, the value iterative algorithms usually work better.

3.3 SORLC design

We define the position and velocity tracking errors of second-order systems as

$$\boldsymbol{e}_{\mathrm{p},i}=\boldsymbol{p}_i-\boldsymbol{p}_{i,\mathrm{r}}, \tag{18}$$

$$\boldsymbol{e}_{\mathrm{v},i}=\boldsymbol{v}_i-\boldsymbol{v}_{i,\mathrm{r}}. \tag{19}$$

The differentials of Eqs. (18) and (19) are yielded as

$$\dot{\boldsymbol{e}}_{\mathrm{p},i}=\dot{\boldsymbol{p}}_i-\dot{\boldsymbol{p}}_{i,\mathrm{r}}=\boldsymbol{e}_{\mathrm{v},i}, \tag{20}$$

$$\dot{\boldsymbol{e}}_{\mathrm{v},i}=\boldsymbol{u}_i+\boldsymbol{f}_i(\boldsymbol{\chi}_i)-\dot{\boldsymbol{v}}_{i,\mathrm{r}}, \tag{21}$$

where $\dot{\boldsymbol{v}}_{i,\mathrm{r}}$ is the differential of $\boldsymbol{v}_{i,\mathrm{r}}$.

We define the integrated tracking error as

$$\boldsymbol{e}_i=\left[\boldsymbol{e}_{\mathrm{p},i}^{\mathrm{T}},\ \boldsymbol{e}_{\mathrm{v},i}^{\mathrm{T}}\right]^{\mathrm{T}}, \tag{22}$$

$$\dot{\boldsymbol{e}}_i=\left[\dot{\boldsymbol{e}}_{\mathrm{p},i}^{\mathrm{T}},\ \dot{\boldsymbol{e}}_{\mathrm{v},i}^{\mathrm{T}}\right]^{\mathrm{T}}. \tag{23}$$

The value function is defined as

$$V_i(\boldsymbol{e}_i)=\int_t^{\infty}L_i\big(\boldsymbol{e}_i(\mu),\boldsymbol{u}_i(\boldsymbol{e}_i)\big)\mathrm{d}\mu, \tag{24}$$

where $L_i\big(\boldsymbol{e}_i(\mu),\boldsymbol{u}_i(\boldsymbol{e}_i)\big)=\alpha_{\mathrm{V}}\boldsymbol{e}_i^{\mathrm{T}}\boldsymbol{e}_i+\beta_{\mathrm{V}}\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{u}_i\in\mathbb{R}$ is the cost function with α_{V} and β_{V} being the adjustable cost parameters. In most cases, if there are no special requirements, it is common and convenient to select $\alpha_{\mathrm{V}}=\beta_{\mathrm{V}}=1$ (Liu DR et al., 2021).

We define $\boldsymbol{u}_i^*$ as the optimal tracking control policy. As a result, the optimal value function is expressed as

$$\begin{aligned}V_i^*(\boldsymbol{e}_i)&=\min_{\boldsymbol{u}_i\in\Theta(\Omega)}\left(\int_t^{\infty}L_i\big(\boldsymbol{e}_i(\mu),\boldsymbol{u}_i(\boldsymbol{e}_i)\big)\mathrm{d}\mu\right)\\&=\int_t^{\infty}L_i\big(\boldsymbol{e}_i(\mu),\boldsymbol{u}_i^*(\boldsymbol{e}_i)\big)\mathrm{d}\mu,\end{aligned} \tag{25}$$

where $\boldsymbol{u}_i\in\Theta(\Omega)$ is the admissible control policy.

Definition 1 (Admissible control policy (Wen et al., 2021)) A control policy $\boldsymbol{u}_i$ is said to be admissible in association with Eq. (24) on a set Ω, denoted by $\boldsymbol{u}_i\in\Theta(\Omega)$, if $\boldsymbol{u}_i$ is continuous, $\boldsymbol{u}_i(0)=\boldsymbol{0}$, $\boldsymbol{u}_i$ stabilizes Eq. (1), and $V_i(\boldsymbol{e}_i)$ is finite.

The HJB equation of a second-order system is yielded by integrating Eqs. (18)–(23) and (25) as

$$\begin{aligned}&H_i\left(\boldsymbol{e}_i,\boldsymbol{u}_i^*,\frac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_i}\right)\\=&L_i(\boldsymbol{e}_i,\boldsymbol{u}_i)+\frac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_i^{\mathrm{T}}}\dot{\boldsymbol{e}}_i\\=&\alpha_{\mathrm{V}}\left\|\boldsymbol{e}_i\right\|^2+\beta_{\mathrm{V}}\left\|\boldsymbol{u}_i^*\right\|^2\\&+\frac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_i^{\mathrm{T}}}\begin{bmatrix}\boldsymbol{v}_i-\boldsymbol{v}_{i,\mathrm{r}}\\ \boldsymbol{u}_i^*+\boldsymbol{f}_i(\boldsymbol{\chi}_i)-\dot{\boldsymbol{v}}_{i,\mathrm{r}}\end{bmatrix}\\=&0,\end{aligned} \tag{26}$$

where $\dfrac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_i}=\left[\dfrac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{p},i}^{\mathrm{T}}},\dfrac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}^{\mathrm{T}}}\right]^{\mathrm{T}}\in\mathbb{R}^{2n}$ is the gradient of V_i^* with respect to $\boldsymbol{e}_i$, and $\dfrac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{p},i}}\in\mathbb{R}^n$ and $\dfrac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}}\in\mathbb{R}^n$ are the gradients of V_i^* with respect to $\boldsymbol{e}_{\mathrm{p},i}$ and $\boldsymbol{e}_{\mathrm{v},i}$, respectively.

By solving $\dfrac{\mathrm{d}H_i\left(\boldsymbol{e}_i,\boldsymbol{u}_i^*,\dfrac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_i}\right)}{\mathrm{d}\boldsymbol{u}_i^*}=0$, the optimal control policy $\boldsymbol{u}_i^*$ is yielded as

$$\boldsymbol{u}_i^*=-\frac{1}{2\beta_{\mathrm{V}}}\frac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}}. \tag{27}$$

Furthermore, the following equation is obtained by substituting Eq. (27) into Eq. (26):

$$\begin{aligned}&H_i\left(\boldsymbol{e}_i,\boldsymbol{u}_i^*,\frac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_i}\right)\\=&\alpha_{\mathrm{V}}\left\|\boldsymbol{e}_i\right\|^2+\frac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{p},i}^{\mathrm{T}}}\boldsymbol{e}_{\mathrm{v},i}+\frac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}^{\mathrm{T}}}\left(\boldsymbol{f}_i(\boldsymbol{\chi}_i)-\dot{\boldsymbol{v}}_{i,\mathrm{r}}\right)\\&-\frac{1}{4\beta_{\mathrm{V}}}\frac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}^{\mathrm{T}}}\frac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}}.\end{aligned} \tag{28}$$

Eq. (27) is not the complete form of an analytical solution of $\boldsymbol{u}_i^*$, since $\dfrac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}}$ is unknown. To implement $\boldsymbol{u}_i^*$, it is required to obtain the term $\dfrac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}}$ by solving Eq. (28). Nevertheless, the analytical solution of $\dfrac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}}$ is difficult to obtain, since

the dynamic models of second-order systems are nonlinear and imprecise. Therefore, an identifier–actor–critic reinforcement learning algorithm is introduced for learning the optimal control policy instead of calculating the analytical solution.

Thus, we divide the optimal value function gradient as

$$\begin{aligned}\frac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}}=&2\beta_{\mathrm{V}}\xi_{\mathrm{p},i}\boldsymbol{e}_{\mathrm{p},i}+2\beta_{\mathrm{V}}\xi_{\mathrm{v},i}\boldsymbol{e}_{\mathrm{v},i}\\&+2\beta_{\mathrm{V}}\boldsymbol{f}_i(\boldsymbol{\chi}_i)+2\beta_{\mathrm{V}}\boldsymbol{\tau}_i+\boldsymbol{V}_i^{\mathrm{o}},\end{aligned}\tag{29}$$

where $\boldsymbol{V}_i^{\mathrm{o}}=-2\beta_{\mathrm{V}}\xi_{\mathrm{p},i}\boldsymbol{e}_{\mathrm{p},i}-2\beta_{\mathrm{V}}\xi_{\mathrm{v},i}\boldsymbol{e}_{\mathrm{v},i}-2\beta_{\mathrm{V}}\boldsymbol{f}_i-2\beta_{\mathrm{V}}\boldsymbol{\tau}_i+\dfrac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}}\in\mathbb{R}^2$, $\xi_{\mathrm{p},i}$ and $\xi_{\mathrm{v},i}$ are positive constants, and $\boldsymbol{\tau}_i\in\mathbb{R}^n$ is the adaptive compensation with the updating law rendered as

$$\dot{\boldsymbol{\tau}}_i=-\xi_{\tau,i}\boldsymbol{\tau}_i+\boldsymbol{u}_{\Delta,i},\tag{30}$$

where $\xi_{\tau,i}>0$ is the designed compensation parameter.

Substituting Eq. (29) into Eq. (27), we obtain

$$\begin{aligned}\boldsymbol{u}_i^*=&-\xi_{\mathrm{p},i}\boldsymbol{e}_{\mathrm{p},i}-\xi_{\mathrm{v},i}\boldsymbol{e}_{\mathrm{v},i}-\boldsymbol{f}_i(\boldsymbol{\chi}_i)\\&-\boldsymbol{\tau}_i-\frac{1}{2\beta_{\mathrm{V}}}\boldsymbol{V}_i^{\mathrm{o}}.\end{aligned}\tag{31}$$

Neural networks (NNs) are known to have excellent approximation capabilities. As a result, given the compact set $\Omega_1\in\mathbb{R}^n$ and $\Omega_2\in\mathbb{R}^{2n}$, for $\forall\boldsymbol{\tau}_i\in\Omega_1$ and $\forall\boldsymbol{\chi}_i\in\Omega_2$, $\boldsymbol{f}_i(\boldsymbol{\chi}_i)$ and $\boldsymbol{V}_i^{\mathrm{o}}$ can be approximated by NNs as

$$\boldsymbol{f}_i(\boldsymbol{\chi}_i)=\left(\boldsymbol{\omega}_{\mathrm{f},i}^*\right)^{\mathrm{T}}\boldsymbol{\Psi}_{\mathrm{f},i}(\boldsymbol{\chi}_i)+\boldsymbol{\sigma}_{\mathrm{f},i}(\boldsymbol{\chi}_i),\tag{32}$$

$$\boldsymbol{V}_i^{\mathrm{o}}=\left(\boldsymbol{\omega}_{\mathrm{V},i}^*\right)^{\mathrm{T}}\boldsymbol{\Psi}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i)+\boldsymbol{\sigma}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i),\tag{33}$$

where $\boldsymbol{\omega}_{\mathrm{f},i}^*\in\mathbb{R}^{w_{\mathrm{f}}\times n}$ and $\boldsymbol{\omega}_{\mathrm{V},i}^*\in\mathbb{R}^{w_{\mathrm{V}}\times n}$ are the ideal weight matrices with w_{f} and w_{V} being the neuron numbers, $\boldsymbol{\Psi}_{\mathrm{f},i}(\boldsymbol{\chi}_i)\in\mathbb{R}^{w_{\mathrm{f}}}$ and $\boldsymbol{\Psi}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i)\in\mathbb{R}^{w_{\mathrm{V}}}$ are the basis function vectors, and $\boldsymbol{\sigma}_{\mathrm{f},i}(\boldsymbol{\chi}_i)\in\mathbb{R}^n$ and $\boldsymbol{\sigma}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i)\in\mathbb{R}^n$ are the approximation errors bounded as $\|\boldsymbol{\sigma}_{\mathrm{f},i}(\boldsymbol{\chi}_i)\|\leq\varepsilon_{\mathrm{f}}$ and $\|\boldsymbol{\sigma}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i)\|\leq\varepsilon_{\mathrm{V}}$ respectively with ε_{f} and ε_{V} being positive constants.

Then, the following equations are obtained by substituting Eqs. (32) and (33) into Eqs. (29) and (31):

$$\begin{aligned}\frac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}}=&2\beta_{\mathrm{V}}\xi_{\mathrm{p},i}\boldsymbol{e}_{\mathrm{p},i}+2\beta_{\mathrm{V}}\xi_{\mathrm{v},i}\boldsymbol{e}_{\mathrm{v},i}+2\beta_{\mathrm{V}}\boldsymbol{\tau}_i\\&+2\beta_{\mathrm{V}}\left(\boldsymbol{\omega}_{\mathrm{f},i}^*\right)^{\mathrm{T}}\boldsymbol{\Psi}_{\mathrm{f},i}(\boldsymbol{\chi}_i)\\&+\left(\boldsymbol{\omega}_{\mathrm{V},i}^*\right)^{\mathrm{T}}\boldsymbol{\Psi}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i)\\&+2\beta_{\mathrm{V}}\boldsymbol{\sigma}_{\mathrm{f},i}(\boldsymbol{\chi}_i)+\boldsymbol{\sigma}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i),\end{aligned}\tag{34}$$

$$\begin{aligned}\boldsymbol{u}_i^*=&-\xi_{\mathrm{p},i}\boldsymbol{e}_{\mathrm{p},i}-\xi_{\mathrm{v},i}\boldsymbol{e}_{\mathrm{v},i}-\boldsymbol{\tau}_i\\&-\left(\boldsymbol{\omega}_{\mathrm{f},i}^*\right)^{\mathrm{T}}\boldsymbol{\Psi}_{\mathrm{f},i}(\boldsymbol{\chi}_i)-\frac{1}{2\beta_{\mathrm{V}}}\left(\boldsymbol{\omega}_{\mathrm{V},i}^*\right)^{\mathrm{T}}\boldsymbol{\Psi}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i)\\&-\boldsymbol{\sigma}_{\mathrm{f},i}(\boldsymbol{\chi}_i)-\frac{1}{2\beta_{\mathrm{V}}}\boldsymbol{\sigma}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i).\end{aligned}\tag{35}$$

Nevertheless, $\boldsymbol{u}_i^*$ cannot be implemented, since the optimal weights $\boldsymbol{\omega}_{\mathrm{f},i}^*$ and $\boldsymbol{\omega}_{\mathrm{V},i}^*$ are unknown. Therefore, an identifier–actor–critic reinforcement learning algorithm is used to learn the optimal control policies, in which $\boldsymbol{\omega}_{\mathrm{f},i}^*$ is estimated by the identifier NN, and $\boldsymbol{\omega}_{\mathrm{V},i}^*$ is estimated by both actor and critic NNs. Specifically, the critic NN estimates $\boldsymbol{\omega}_{\mathrm{V},i}^*$ in $\dfrac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}}$, while the actor NN estimates $\boldsymbol{\omega}_{\mathrm{V},i}^*$ in $\boldsymbol{u}_i^*$. The idea underlying such a network structure stems from the actor–critic reinforcement learning algorithm, which variously uses two different networks to implement and evaluate the control policy.

Specifically, the identifier NN is designed to estimate the unknown nonlinear dynamics part as

$$\hat{\boldsymbol{f}}_i(\boldsymbol{\chi}_i)=\hat{\boldsymbol{\omega}}_{\mathrm{f},i}^{\mathrm{T}}\boldsymbol{\Psi}_{\mathrm{f},i}(\boldsymbol{\chi}_i),\tag{36}$$

where $\hat{\boldsymbol{f}}_i(\boldsymbol{\chi}_i)\in\mathbb{R}^n$ is the estimate of $\boldsymbol{f}_i(\boldsymbol{\chi}_i)$, and $\hat{\boldsymbol{\omega}}_{\mathrm{f},i}\in\mathbb{R}^{w_{\mathrm{f}}\times n}$ is the weight matrix of the identifier NN. The updating law of identifier NN is designed as

$$\dot{\hat{\boldsymbol{\omega}}}_{\mathrm{f},i}=\boldsymbol{\Gamma}_{\mathrm{f},i}\left[\boldsymbol{\Psi}_{\mathrm{f},i}(\boldsymbol{\chi}_i)\left(\boldsymbol{e}_{\mathrm{p},i}^{\mathrm{T}}+\boldsymbol{e}_{\mathrm{v},i}^{\mathrm{T}}\right)-\nu_{\mathrm{f},i}\hat{\boldsymbol{\omega}}_{\mathrm{f},i}\right],\tag{37}$$

where $\boldsymbol{\Gamma}_{\mathrm{f},i}\in\mathbb{R}^{w_{\mathrm{f}}\times w_{\mathrm{f}}}$ is a positive-definite matrix and $\nu_{\mathrm{f},i}\in\mathbb{R}$ is the designed identifier parameter.

Then, the critic NN is designed to evaluate the control performance as

$$\begin{aligned}\frac{\mathrm{d}\hat{V}_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}}=&2\beta_{\mathrm{V}}\xi_{\mathrm{p},i}\boldsymbol{e}_{\mathrm{p},i}+2\beta_{\mathrm{V}}\xi_{\mathrm{v},i}\boldsymbol{e}_{\mathrm{v},i}\\&+2\beta_{\mathrm{V}}\hat{\boldsymbol{\omega}}_{\mathrm{f},i}^{\mathrm{T}}\boldsymbol{\Psi}_{\mathrm{f},i}(\boldsymbol{\chi}_i)+2\beta_{\mathrm{V}}\boldsymbol{\tau}_i\\&+\hat{\boldsymbol{\omega}}_{\mathrm{c},i}^{\mathrm{T}}\boldsymbol{\Psi}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i),\end{aligned}\tag{38}$$

where $\dfrac{\mathrm{d}\hat{V}_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}}\in\mathbb{R}^n$ is the estimate of $\dfrac{\mathrm{d}V_i^*(\boldsymbol{e}_i)}{\mathrm{d}\boldsymbol{e}_{\mathrm{v},i}}$, $\hat{\boldsymbol{\omega}}_{\mathrm{c},i}\in\mathbb{R}^{w_{\mathrm{V}}\times n}$ is the weight matrix of the critic NN, and $\hat{\boldsymbol{V}}_i^{\mathrm{o}}=\hat{\boldsymbol{\omega}}_{\mathrm{c},i}^{\mathrm{T}}\boldsymbol{\Psi}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i)$ in the critic NN. The updating law of the critic NN is designed as

$$\dot{\hat{\boldsymbol{\omega}}}_{\mathrm{c},i}=-\gamma_{\mathrm{c},i}\boldsymbol{\Psi}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i)\boldsymbol{\Psi}_{\mathrm{V},i}^{\mathrm{T}}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i)\hat{\boldsymbol{\omega}}_{\mathrm{c},i},\tag{39}$$

where $\gamma_{\mathrm{c},i}>0$ is the critic learning factor.

Finally, the actor NN is designed to implement the control input as

$$\begin{aligned}\boldsymbol{u}_i =& -\xi_{\mathrm{p},i}\boldsymbol{e}_{\mathrm{p},i}-\xi_{\mathrm{v},i}\boldsymbol{e}_{\mathrm{v},i}-\hat{\boldsymbol{\omega}}_{\mathrm{f},i}^{\mathrm{T}}\boldsymbol{\Psi}_{\mathrm{f},i}(\boldsymbol{\chi}_i)\\&-\boldsymbol{\tau}_i-\frac{1}{2\beta_{\mathrm{V}}}\hat{\boldsymbol{\omega}}_{\mathrm{a},i}^{\mathrm{T}}\boldsymbol{\Psi}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i),\end{aligned}\tag{40}$$

where $\hat{\boldsymbol{\omega}}_{\mathrm{a},i}\in\mathbb{R}^{w_{\mathrm{V}}\times n}$ is the weight matrix of the actor NN, and $\hat{\boldsymbol{V}}_i^{\mathrm{o}}=\hat{\boldsymbol{\omega}}_{\mathrm{a},i}^{\mathrm{T}}\boldsymbol{\Psi}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i)$ in the actor NN. The updating law of the actor NN is designed as

$$\begin{aligned}\dot{\hat{\boldsymbol{\omega}}}_{\mathrm{a},i}=&-\boldsymbol{\Psi}_{\mathrm{V},i}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i)\boldsymbol{\Psi}_{\mathrm{V},i}^{\mathrm{T}}(\boldsymbol{\chi}_i,\boldsymbol{\tau}_i)[\gamma_{\mathrm{a},i}(\hat{\boldsymbol{\omega}}_{\mathrm{a},i}-\hat{\boldsymbol{\omega}}_{\mathrm{c},i})\\&+\gamma_{\mathrm{c},i}\hat{\boldsymbol{\omega}}_{\mathrm{c},i}],\end{aligned}\tag{41}$$

where $\gamma_{\mathrm{a},i}>0$ is the actor learning factor.

Theorem 1 Using the SORLCs (36), (38), and (40) with updating laws (37), (39), and (41) for the second-order systems (1) with bounded initial conditions, and ascertaining that the design parameters are satisfied with the following conditions:

$$\begin{cases}\xi_{\mathrm{e},i}>\xi_{\mathrm{p},i}+\xi_{\mathrm{v},i}-\sqrt{4(\xi_{\mathrm{p},i}-2)(\xi_{\mathrm{v},i}-3)},\\ \xi_{\mathrm{p},i}>2,\ \xi_{\mathrm{v},i}>3,\ \xi_{\tau,i}>\dfrac{3}{2},\ \gamma_{\mathrm{a},i}>\dfrac{1}{2\beta_{\mathrm{V}}^2},\\ \gamma_{\mathrm{a},i}>\gamma_{\mathrm{c},i}>\dfrac{\gamma_{\mathrm{a},i}}{2},\end{cases}\tag{42}$$

we infer that (1) the tracking error $\boldsymbol{e}_i$, adaptive compensation $\boldsymbol{\tau}_i$, weight error of identifier $\tilde{\boldsymbol{\omega}}_{\mathrm{f},i}$, weight error of actor $\tilde{\boldsymbol{\omega}}_{\mathrm{a},i}$, and weight error of critic $\tilde{\boldsymbol{\omega}}_{\mathrm{c},i}$ are SGUUB and (2) $\boldsymbol{e}_i$ will converge to a desired accuracy pursuant to adjusting $\xi_{\mathrm{e},i}$ in such a way that it becomes large enough.

Definition 2 (SGUUB (Wen et al., 2021)) An error $\boldsymbol{e}_i(t)$ is said to be semi-globally uniformly ultimately bounded for any compact set Ω_{e} and for all $\boldsymbol{e}_i(t_0)\in\Omega_{\mathrm{e}}$, if there exist $\varkappa_i>0$ and $T(\varkappa_i,\boldsymbol{e}_i(t_0))$ such that $\|\boldsymbol{e}_i(t)\|\le\varkappa_i$, $\forall t\ge t_0+T$.

The proofs of Theorem 1, mission stability, and boundedness are provided in the supplementary materials.

4 Simulations

In this section, numerical simulations are considered under which four second-order systems form desired formations while avoiding obstacles via executing the designed OA, FM, and FR behaviors.

4.1 Case study

In the simulation case, the state dimension of second-order systems is set to $n=2$. The desired mission of the FM behavior is $\boldsymbol{p}_{\mathrm{c,d}}=[-2.5+t;0]$. The initial positions are $\boldsymbol{p}_{1,0}=[-2;8]$, $\boldsymbol{p}_{2,0}=[-2;-8]$, $\boldsymbol{p}_{3,0}=[-7;14]$, and $\boldsymbol{p}_{4,0}=[-7;-14]$. The target positions are $\boldsymbol{p}_{1,\mathrm{g}}=[100;6]$, $\boldsymbol{p}_{2,\mathrm{g}}=[100;-6]$, $\boldsymbol{p}_{3,\mathrm{g}}=[95;12]$, and $\boldsymbol{p}_{4,\mathrm{g}}=[95;-12]$. The initial velocities all are $[0;0]$. The relative positions required for agents and the formation centroid to form the desired formation are $\boldsymbol{p}_1^{\mathrm{c}}=[2.5;6]$, $\boldsymbol{p}_2^{\mathrm{c}}=[2.5;-6]$, $\boldsymbol{p}_3^{\mathrm{c}}=[-2.5;12]$, and $\boldsymbol{p}_4^{\mathrm{c}}=[-2.5;-12]$. The reconstruction matrices are $\boldsymbol{\Gamma}_{\mathrm{FR},1}=[9/5,0;0,0]$, $\boldsymbol{\Gamma}_{\mathrm{FR},2}=[3/5,0;0,0]$, $\boldsymbol{\Gamma}_{\mathrm{FR},3}=[-3/5,0;0,0]$, and $\boldsymbol{\Gamma}_{\mathrm{FR},4}=[-9/5,0;0,0]$. The initial weights of the identifier are $\boldsymbol{\omega}_{\mathrm{f},1}=\mathbf{1}_{12\times2}$, $\boldsymbol{\omega}_{\mathrm{f},2}=\mathbf{0.98}_{12\times2}$, $\boldsymbol{\omega}_{\mathrm{f},3}=\mathbf{0.96}_{12\times2}$, and $\boldsymbol{\omega}_{\mathrm{f},4}=\mathbf{0.94}_{12\times2}$. The initial weights of the actor are $\boldsymbol{\omega}_{\mathrm{a},1}=\mathbf{0.5}_{12\times2}$, $\boldsymbol{\omega}_{\mathrm{a},2}=\mathbf{0.48}_{12\times2}$, $\boldsymbol{\omega}_{\mathrm{a},3}=\mathbf{0.46}_{12\times2}$, and $\boldsymbol{\omega}_{\mathrm{a},4}=\mathbf{0.44}_{12\times2}$. The initial weights of the critic are $\boldsymbol{\omega}_{\mathrm{c},1}=\mathbf{1}_{12\times2}$, $\boldsymbol{\omega}_{\mathrm{c},2}=\mathbf{0.96}_{12\times2}$, $\boldsymbol{\omega}_{\mathrm{c},3}=\mathbf{0.92}_{12\times2}$, and $\boldsymbol{\omega}_{\mathrm{c},4}=\mathbf{0.88}_{12\times2}$. The NNs are designed with $w_{\mathrm{f}}=12$, $w_{\mathrm{V}}=12$, with centers evenly spaced in the range $[-6,6]$ and the width being 2. The dynamic models of agents are set as

$$\begin{cases}\dot{\boldsymbol{p}}_i=\boldsymbol{v}_i,\\ \dot{\boldsymbol{v}}_i=\boldsymbol{u}_i+\underbrace{\begin{bmatrix}v_i(1)\sin^2(p_i(2))+v_i(2)\cos^2(p_i(1))\\ v_i(1)\sin^2(p_i(2))-v_i(2)\cos^2(p_i(1))\end{bmatrix}}_{\boldsymbol{f}_i(\boldsymbol{\chi}_i)}.\end{cases}\tag{43}$$

Remark 3 Eq. (43) is a numerical simulation case of dynamic models (1), which does not correspond to the practical system. However, the dynamic models (1) can characterize many practical systems, such as unmanned mobile vehicles (Huang et al., 2019), unmanned aerial vehicles (Dong et al., 2017), and robotic manipulators (Wen et al., 2017). In fact, any practical system that can be modeled or simplified into the form of the dynamic models (1) would be capable of implementing the proposed MARLBC method.

Other simulation parameters of the environment and MARLBC are listed in Tables 1 and 2, respectively.

4.2 Simulation results

4.2.1 Comparison between the proposed MARLMS and other existing mission supervisors

The mission performance of the proposed MARLMS is compared with those of the FSAMS (Marino et al., 2013), MPCMS (Chen YT et al.,

2020), and RLMS (Zhang et al., 2022). The switching rules of the MARLMS for the i^{th} agent are designed as follows: if $d_{\text{OA}} < \min\{d_i^{\text{o}}\} \leq 2d_{\text{OA}}$, then the FR behavior is switched as the highest priority; if $\min\{d_i^{\text{o}}\} \leq d_{\text{OA}}$, then the OA behavior is switched as the highest priority; otherwise, the FM behavior is switched as the highest priority. For the MPCMS, the cost function is set as the weighted sum of formation and reconfiguration errors; meanwhile, the OA behavior is set as the constraints. Since the RLMS cannot be applied to the cooperative behaviors, both FM and FR behaviors are regarded as the motion behavior. The results of different mission supervisors are shown in Figs. 2 and 3. Because the FSAMS has the switching effect near the state transition threshold, the trajectories have strong oscillations and the safety constraints are violated sometimes. The MPCMS achieves perfect mission performance, but its iteration time is much longer than that of other mission supervisors. This phenomenon is explained by the fact that the MPCMS has to perform online calculation of the optimal behavior priority at each sampling time. Because of the cooperative behaviors of agents being incapable of implementation, the RLMS ignores the swarm intelligence. The proposed MARLMS maintains not only the good mission performance but also the low iteration time. To better demonstrate the advantages of MARLMS, some numerical results are shown in Tables 3 and 4. Compared to the FSAMS, the numbers of behavior priority switches of the MARLMS are significantly reduced, by 96.6%, 97.7%, 97.7%, and 96.6%, for the 1st, 2nd, 3rd, and 4th agents, respectively. Compared to the MPCMS, the online iteration time of the MARLMS is significantly reduced, by 99.8%, for all the four agents. All results prove the superiority of the proposed MARLMS.

Table 1 Simulation parameters of the environment

Parameter	Symbol	Value
Obstacle 1	$\boldsymbol{p}_{\text{O}1}$	$[20; 5]$
Obstacle 2	$\boldsymbol{p}_{\text{O}2}$	$[40; -11]$
Obstacle 3	$\boldsymbol{p}_{\text{O}3}$	$(x-60)^2+(y-9)^2=5^2$
Obstacle 4	$\boldsymbol{p}_{\text{O}4}$	$(x-70)^2+(y-9)^2=5^2$
Obstacle 5	$\boldsymbol{p}_{\text{O}5}$	$(x-60)^2+(y+9)^2=5^2$
Obstacle 6	$\boldsymbol{p}_{\text{O}6}$	$(x-70)^2+(y+9)^2=5^2$

Table 2 Simulation parameters of the MARLBC method

Parameter	Symbol	Value
Mission gains	Λ_{OA}, $\boldsymbol{\Lambda}_{\text{FM}}$, $\boldsymbol{\Lambda}_{\text{FR}}$	20, $8\boldsymbol{I}_n$, $8\boldsymbol{I}_n$
Safe distance	d_{OA}	2
Positive-definite matrix of the identifier	$\boldsymbol{\Gamma}_{\text{f},i}$	$0.4\boldsymbol{I}_{12}$
Designed parameter of the identifier	$\nu_{\text{f},i}$	0.6
Positive constants with respect to tracking errors	$\xi_{\text{p},i}$, $\xi_{\text{v},i}$	100, 100
Compensation	$\xi_{\tau,i}$, $\boldsymbol{\tau}_i(0)$	50, [0; 0] N·m
Input constraint	$u_{\text{lim},i}$	200 N·m
Sampling time	Δt	0.05 s
Learning factors	$\gamma_{\text{a},i}$, $\gamma_{\text{c},i}$	6, 4
Leniency factor	$\kappa_{\mathfrak{L}}$	2
Temperatures	ρ_γ, d_γ	−0.01, 0.95
Learning rate	α_t	0.0001
Exponent	ζ	0.999
Replay buffer	$\mathfrak{D}$	50 000
Total number of training episodes	T_{e}	100 000
Total number of time steps	T_{s}	2000
Weights	α_{V}, β_{V}	1, 1

4.2.2 Comparison between the proposed SORLC and the traditional RLC

The control performances of the proposed SORLC and the traditional RLC (Zhang et al., 2022) are compared, and the results are shown in Figs. 4–7. Fig. 4 shows that the network weights of the identifier, actor, and critic are all converged and bounded. Fig. 5 shows that the weight errors of the identifier, actor, and critic are all converged to near zero. Based on the results of Figs. 4 and 5, the various optimal weights are obtained; thus, the optimal weight norms of identifiers of the four agents are all 0.1; the optimal weight norms of actors of the 1st, 2nd, 3rd,

Table 3 Number of behavior priority switches of different mission supervisors

Agent	Number of switches			
	MARLMS	FSAMS	MPCMS	RLMS
1	4	118	4	6
2	2	86	2	4
3	2	86	2	4
4	4	118	4	6

Table 4 Online iteration time of different mission supervisors

Agent	Time (ms)			
	MARLMS	FSAMS	MPCMS	RLMS
1	1.4	1.0	597.2	1.0
2	1.1	0.8	530.8	0.7
3	1.3	0.9	663.5	0.8
4	1.5	1.3	796.2	1.1

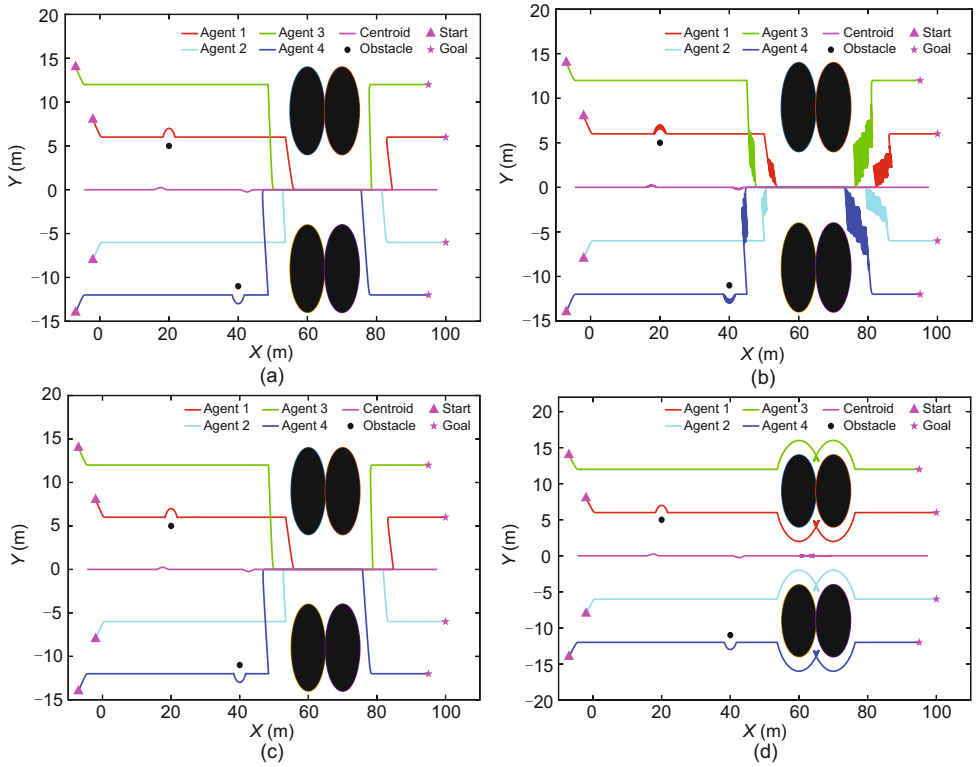

Fig. 2 Trajectories of agents with different mission supervisors: (a) MARLMS; (b) FSAMS; (c) MPCMS; (d) RLMS (References to color refer to the online version of this figure)

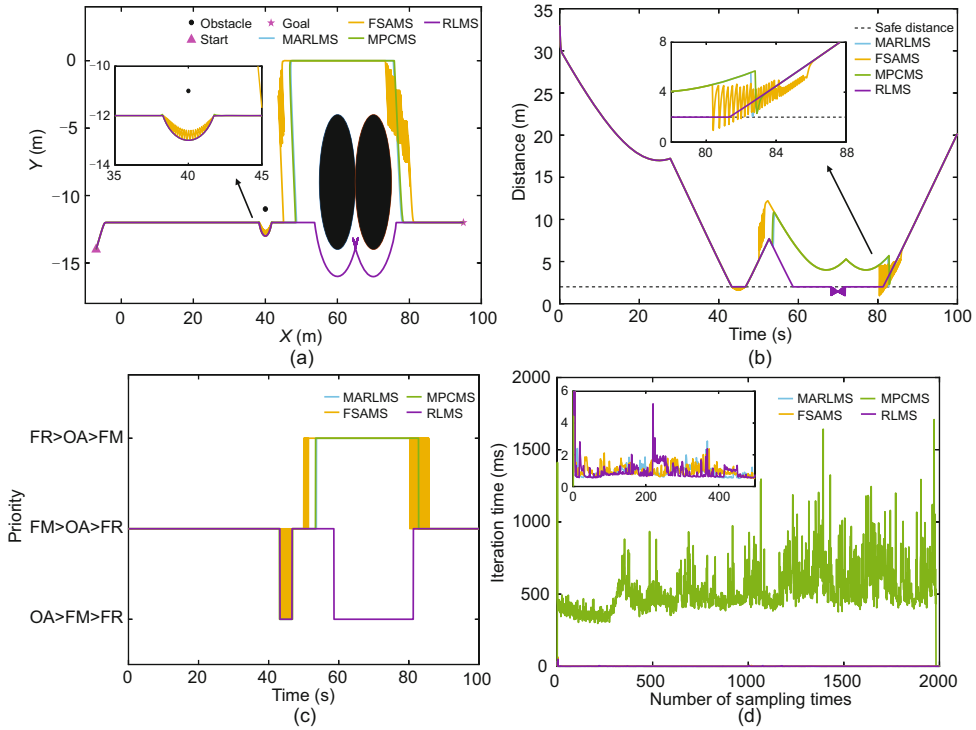

Fig. 3 Results of the 4$^{\text{th}}$ agent with different mission supervisors: (a) trajectories; (b) distances between the agent and obstacles; (c) behavior priorities; (d) iteration time (References to color refer to the online version of this figure)

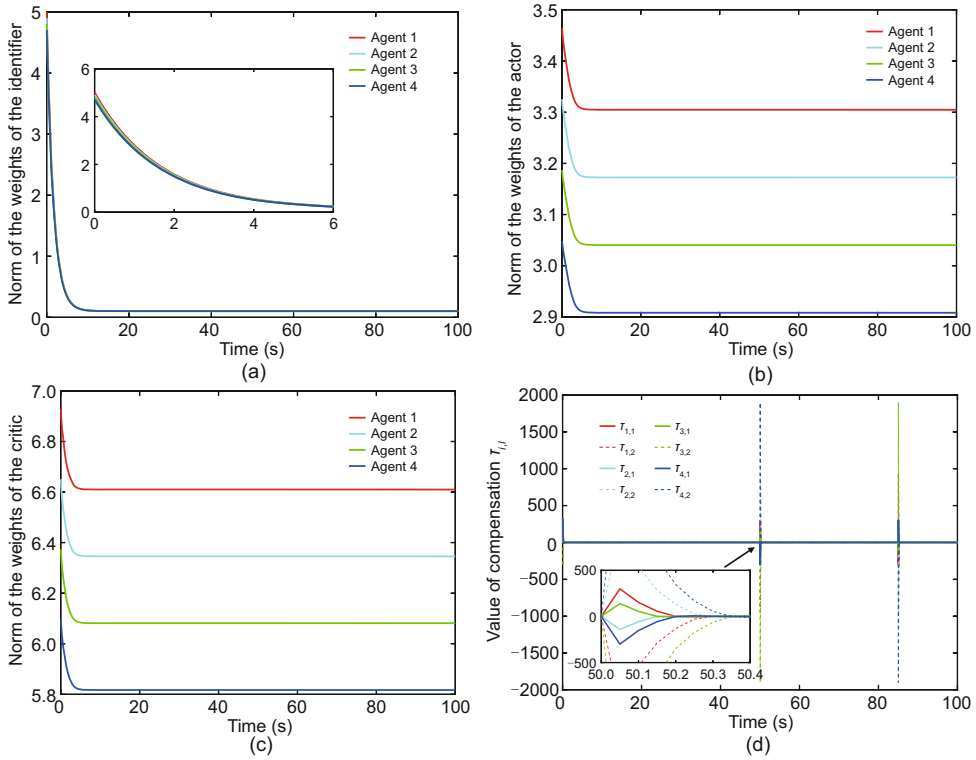

Fig. 4 Results of the proposed SORLC: (a) weights of the identifier; (b) weights of the actor; (c) weights of the critic; (d) adaptive compensators (References to color refer to the online version of this figure)

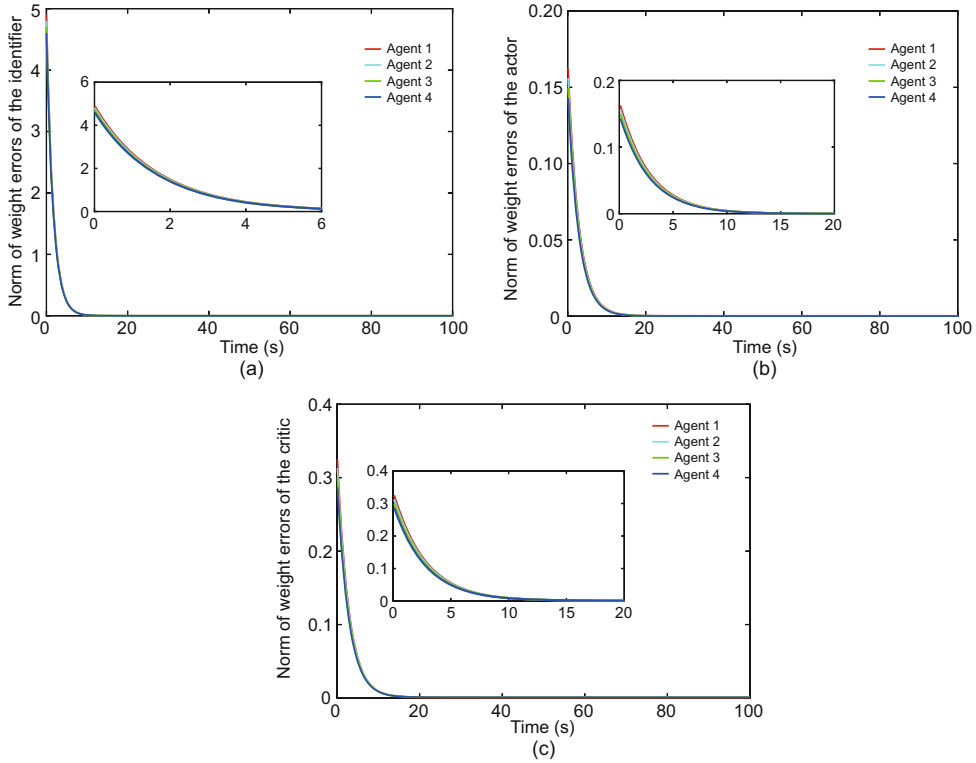

Fig. 5 Results of the proposed SORLC: (a) weight errors of the identifier; (b) weight errors of the actor; (c) weight errors of the critic (References to color refer to the online version of this figure)

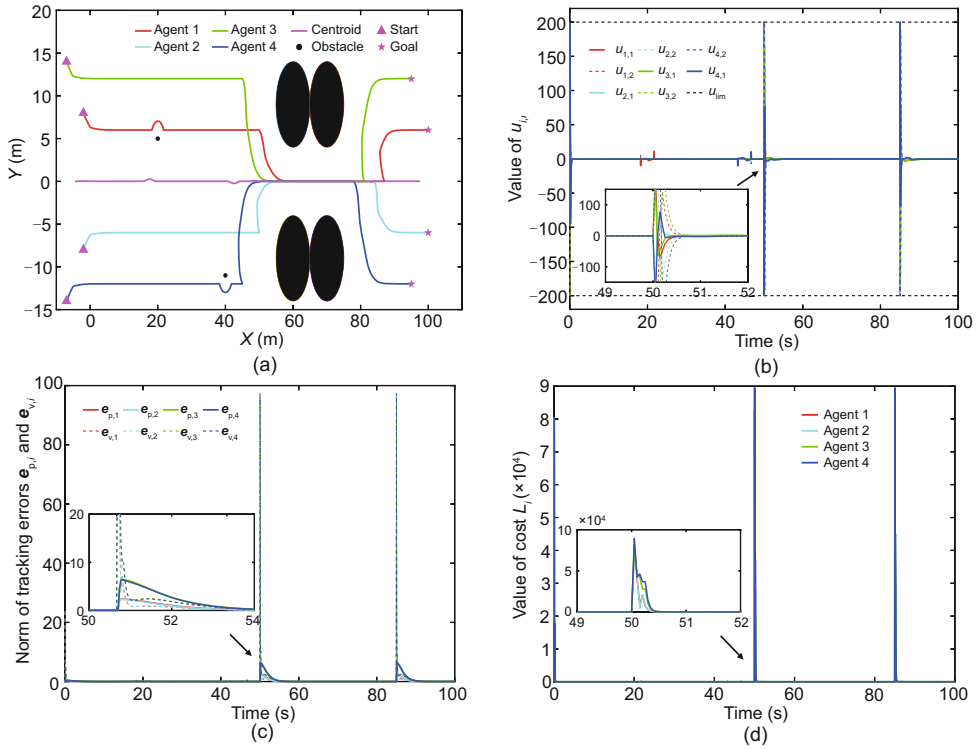

Fig. 6 Control performance of the proposed SORLC: (a) trajectories; (b) control inputs; (c) tracking errors; (d) costs (References to color refer to the online version of this figure)

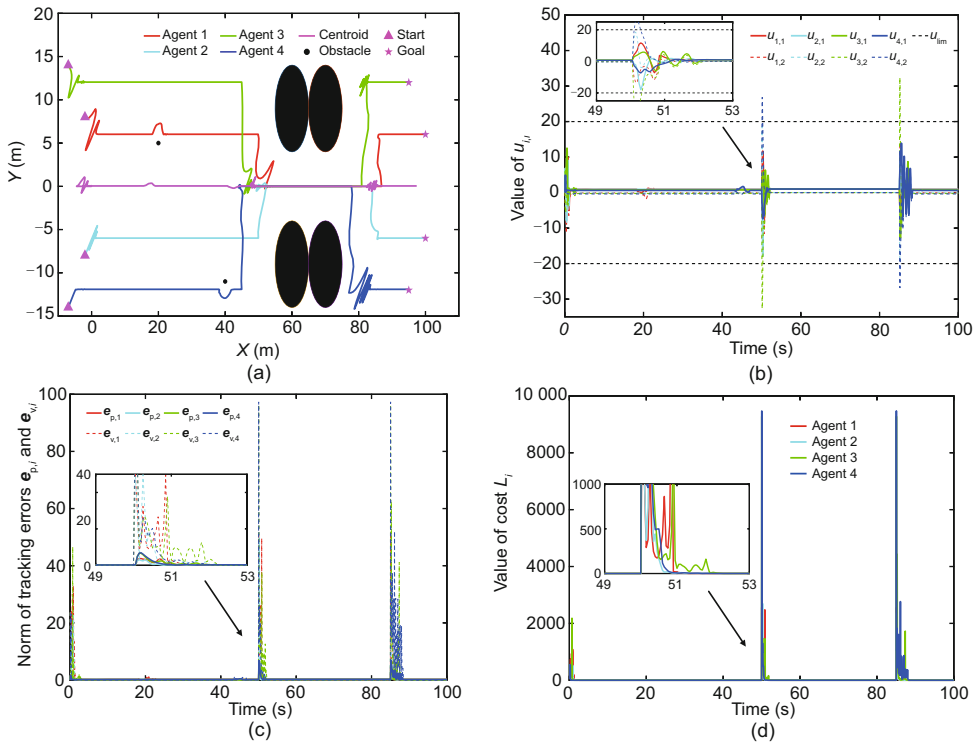

Fig. 7 Control performance of the traditional RLC: (a) trajectories; (b) control inputs; (c) tracking errors; (d) costs (References to color refer to the online version of this figure)

and 4^{th} agents are 3.305, 3.173, 3.040, and 2.908, respectively; the corresponding optimal weight norms of critics are 6.610, 6.346, 6.081, and 5.817. Because the control inputs of agents are not always in a saturated state, the adaptive compensators are bounded. Figs. 6 and 7 show the control performances of the SORLC and RLC, respectively. It is easy to see from Fig. 7 that once the behavior priorities are switched, the tracking accuracy of RLC will become very poor, and the trajectories of agents will become unstable. Moreover, it is easy to see from Figs. 6c and 7c that the velocity error of SORLC is converged within 0.7 s at the initial phase, but the RLC needs 3.4 s. This phenomenon is explained by the fact that the RLC cannot effectively track the reference velocities or ensure the convergence of velocity errors. As a result, when the behavior priorities are switched and reference velocities are suddenly changed, the RLC is unable to track the velocity signals, and its control accuracy is greatly reduced. Since the SORLC guarantees the convergence of both position and velocity errors, the trajectories of agents are smooth. For the second-order systems, it is necessary to track position and velocity simultaneously to maintain control accuracy. In addition, the SORLC always maintains the control input value below the input limit by implementing the input saturation constraints. Although the control accuracy will be slightly reduced, the actuator can be prevented from exceeding the physical limit when the behavior priorities are switched. All results prove the superiority of the proposed SORLC.

4.2.3 Comparison between the proposed MARLBC and other existing NSBC methods

The control performance of the proposed MARLBC is compared with those of the finite-time NSBC (Huang et al., 2019), fixed-time NSBC (Zhou et al., 2022), and RLBC (Zhang et al., 2022) methods. The results of different NSBC methods are shown in Figs. 8 and 9, where Fig. 8 shows the trajectories of agents and Fig. 9 shows the control performance of the 4^{th} agent. The finite-time NSBC method has satisfactory control performance, but its tracking control has obvious overshoot when the behavior priorities are switched. The fixed-time NSBC method has the highest error convergence speed and the smallest tracking error, but its control cost is the largest. Specifically, the control inputs and costs of both the finite-time and fixed-time NSBC methods will rise to unacceptable values, once the behavior priorities are switched. This phenomenon is explained by the fact that both finite-time and fixed-time NSBC methods will greatly increase the control input to achieve the desired control accuracy. It is not difficult to find from Fig. 9 that the RLBC method has many undesirable control results, since it ignores swarm intelligence in the decision layer and cannot effectively track the reference velocity signals in the control layer. MARLBC not only strictly satisfies the control saturation constraints, but also reduces the control cost when the behavior priorities are switched. Note that when the control limit is not implemented and the behavior priorities are switched, the control input will often increase to a value that may exceed the physical limit of the actuator. MARLBC is the only second-order behavioral control method that can maintain the control accuracy and satisfy the control limit simultaneously. To better demonstrate the advantages of MARLBC, some numerical results are shown in Tables 5 and 6, where Max-ai and Tot-ai ($i=1,2,3,4$) represent the maximum (corresponding to some performance indicator) and total (cumulative value corresponding to some performance indicator) values of the i^{th} agent, respectively (here, the performance indicator is either control cost or control input). The maximum and total values of control cost are calculated by $\max_t\{\boldsymbol{e}_i^{\mathrm{T}}\boldsymbol{e}_i+\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{u}_i\}$ and $\int_0^{\infty}\left(\boldsymbol{e}_i^{\mathrm{T}}\boldsymbol{e}_i+\boldsymbol{u}_i^{\mathrm{T}}\boldsymbol{u}_i\right)\mathrm{d}\mu$, respectively. The maximum and total values of control input are calculated by $\max_t\{\|\boldsymbol{u}_i\|\}$ and $\int_0^{\infty}\|\boldsymbol{u}_i\|\,\mathrm{d}\mu$, respectively. Compared to the finite-time NSBC framework, both the maximum and total values of control cost of MARLBC are reduced, where the maximum values of the 1^{st}, 2^{nd}, 3^{rd}, and 4^{th} agents are reduced by 47.1%, 47.1%, 85%, and 85%, respectively, and the corresponding total values are reduced by 12.2%, 9.9%, 58.8%, and 56.2%. In addition, both the maximum and total values of control input of MARLBC are reduced compared with those of the finite-time NSBC framework, where the maximum values of the 1^{st}, 2^{nd}, 3^{rd}, and 4^{th} agent are reduced by 47.9%, 47.9%, 74.0%, and 74.0%, respectively, and the corresponding total values are reduced by 19.8%, 18.8%, 21.9%, and 21.6%. Compared to the fixed-time NSBC framework, both the maximum and total values of control cost of MARLBC are also

Table 5 Maximum and total control cost values of different NSBC frameworks

Index	Maximum control cost value ($\times 10^4$)			Index	Total control cost value ($\times 10^4$)		
	MARLBC	Finite-time	Fixed-time		MARLBC	Finite-time	Fixed-time
Max-a1	9	17	103	Tot-a1	382	435	518
Max-a2	9	17	115	Tot-a2	364	404	790
Max-a3	9	60	372	Tot-a3	570	1384	1813
Max-a4	9	60	372	Tot-a4	609	1391	1868

Table 6 Maximum and total control input values of different NSBC frameworks

Index	Maximum control input value			Index	Total control input value		
	MARLBC	Finite-time	Fixed-time		MARLBC	Finite-time	Fixed-time
Max-a1	200	384	960	Tot-a1	2210	2756	9127
Max-a2	200	384	1061	Tot-a2	2341	2883	13 370
Max-a3	200	768	1920	Tot-a3	4118	5274	16 112
Max-a4	200	768	1920	Tot-a4	4285	5466	16 646

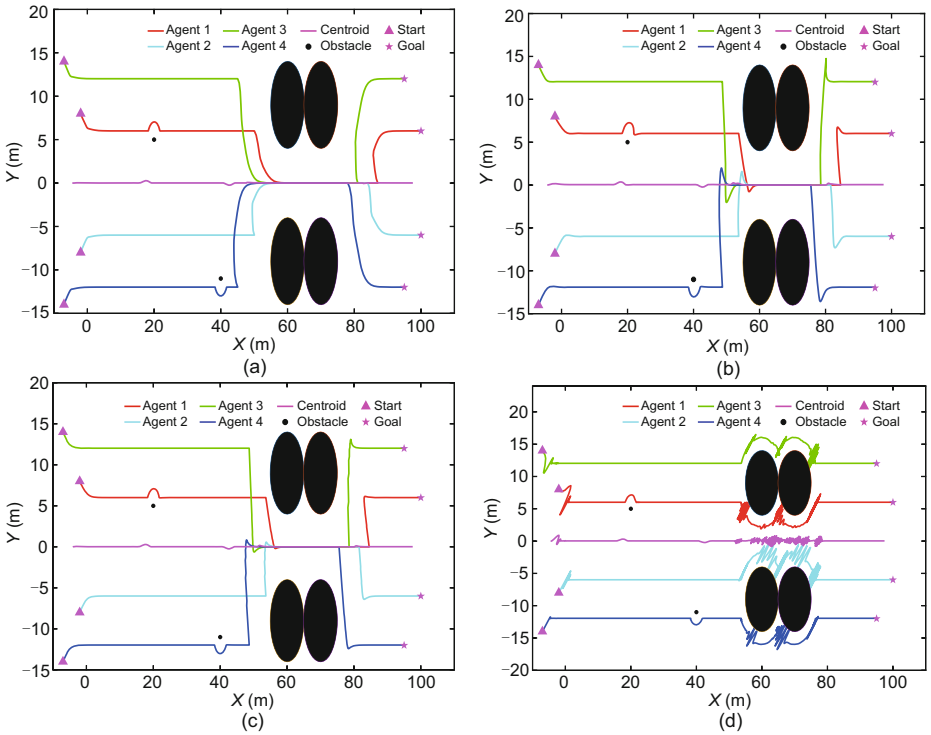

Fig. 8 Trajectories of agents with different NSBC methods: (a) MARLBC; (b) finite-time NSBC; (c) fixed-time NSBC; (d) RLBC (References to color refer to the online version of this figure)

reduced, where the maximum values of the 1st, 2nd, 3rd, and 4th agents are reduced by 91.3%, 92.2%, 97.6%, and 97.6%, respectively, and the corresponding total values are reduced by 26.3%, 53.9%, 68.6%, and 67.4%. Moreover, compared to the fixed-time NSBC framework, both the maximum and total values of control input of MARLBC are reduced, where the maximum values of the 1st, 2nd, 3rd, and 4th agents are reduced by 79.2%, 81.1%, 89.6%, and 89.6%, respectively, and the corresponding total values are reduced by 75.8%, 82.5%, 74.4%, and 74.3%. All results prove the superiority of the proposed MARLBC method.

5 Conclusions

In this paper, a novel MARLBC method is proposed for nonlinear second-order systems to achieve

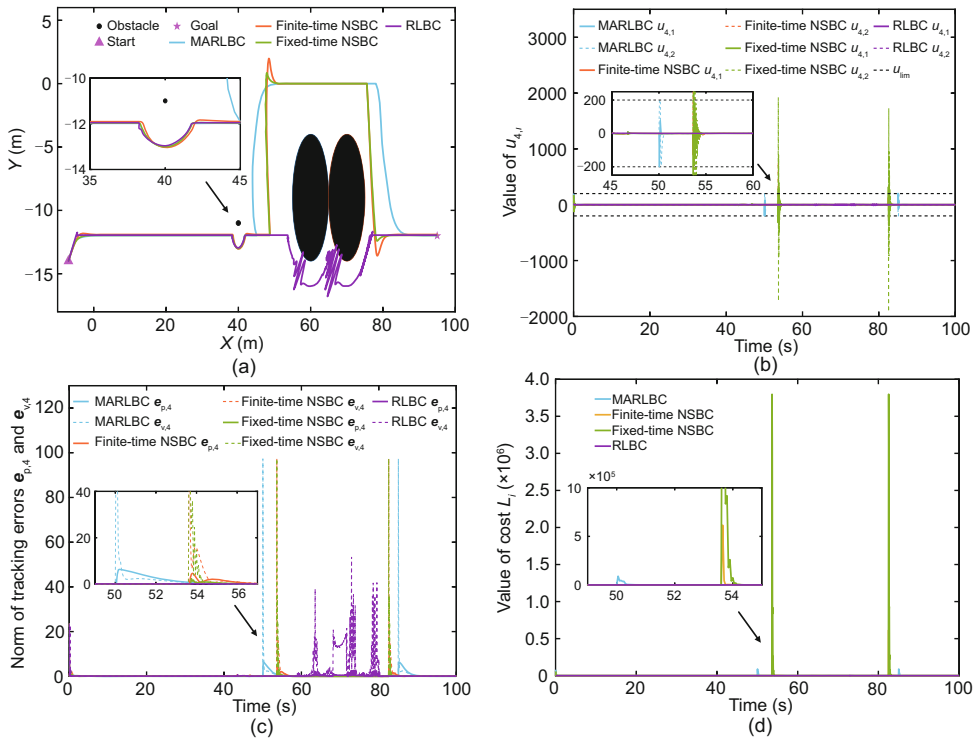

Fig. 9 Control performances of the 4th agent with different NSBC methods: (a) trajectories; (b) control inputs; (c) tracking errors; (d) costs (References to color refer to the online version of this figure)

swarm intelligence by performing multiple conflicting missions. The proposed MARLBC is a decision-control integrated structure including the MARLMS and SORLC. At the decision layer, the MARLMS learns an optimal joint behavior priority policy to assign behavior priorities dynamically and intelligently. At the control layer, the SORLC learns a group of optimal control policies to track both the position and velocity signals, and implement the input saturation constraints. Simulation results show that MARLBC has smaller control input and cost values than the existing NSBC methods of second-order systems when the behavior priorities are switched. In addition, compared with the traditional RLBC, the proposed MARLBC allows the implementation of cooperative behavior and ensures the convergence of velocity signals. Since the NSBC frame relies on a centralized mission supervisor to assign behavior priorities, the scalability of MARLBC is poor. Our future research would involve the overcoming of defects of the centralized mission supervisor via designing a distributed behavioral control structure.

Contributors

Jie HUANG and Zhenyi ZHANG designed the research. Zhenyi ZHANG and Congjie PAN processed the data and drafted the paper. Jie HUANG revised and finalized the paper.

Conflict of interest

All the authors declare that they have no conflict of interest.

Data availability

The data that support the findings of this study are available from the corresponding author upon reasonable request.

References

Ahmad S, Feng Z, Hu GQ, 2014. Multi-robot formation control using distributed null space behavioral approach. IEEE Int Conf on Robotics and Automation, p.3607-3612. https://doi.org/10.1109/icra.2014.6907380

Anschel O, Baram N, Shimkin N, 2017. Averaged-DQN: variance reduction and stabilization for deep reinforcement learning. Proc 34th Int Conf on Machine Learning, p.176-185.

Antonelli G, Chiaverini S, 2006. Kinematic control of platoons of autonomous vehicles. *IEEE Trans Robot*, 22(6):1285-1292. https://doi.org/10.1109/TRO.2006.886272

Arkin RC, 1989. Motor schema-based mobile robot navigation. *Int J Robot Res*, 8(4):92-112. https://doi.org/10.1177/027836498900800406

Balch T, Arkin RC, 1998. Behavior-based formation control for multirobot teams. *IEEE Trans Robot Autom*, 14(6):926-939. https://doi.org/10.1109/70.736776

Brooks RA, 1986. A robust layered control system for a mobile robot. *IEEE J Robot Autom*, 2(1):14-23. https://doi.org/10.1109/JRA.1986.1087032

Brooks RA, 1991. New approaches to robotics. *Science*, 253(5025):1227-1232. https://doi.org/10.1126/science.253.5025.1227

Cao SJ, Sun L, Jiang JJ, et al., 2023. Reinforcement learning-based fixed-time trajectory tracking control for uncertain robotic manipulators with input saturation. *IEEE Trans Neur Netw Learn Syst*, 34(8):4584-4595. https://doi.org/10.1109/TNNLS.2021.3116713

Cao YC, Yu WW, Ren W, et al., 2013. An overview of recent progress in the study of distributed multi-agent coordination. *IEEE Trans Ind Inform*, 9(1):427-438. https://doi.org/10.1109/TII.2012.2219061

Chen J, Gan MG, Huang J, et al., 2016. Formation control of multiple Euler–Lagrange systems via null-space-based behavioral control. *Sci China Inform Sci*, 59(1):1-11. https://doi.org/10.1007/s11432-015-5504-6

Chen YT, Zhang ZY, Huang J, 2020. Dynamic task priority planning for null-space behavioral control of multi-agent systems. *IEEE Access*, 8:149643-149651. https://doi.org/10.1109/ACCESS.2020.3016347

Dong XW, Zhou Y, Ren Z, et al., 2017. Time-varying formation tracking for second-order multi-agent systems subjected to switching topologies with application to quadrotor formation flying. *IEEE Trans Ind Electron*, 64(6):5014-5024. https://doi.org/10.1109/TIE.2016.2593656

Garattoni L, Birattari M, 2018. Autonomous task sequencing in a robot swarm. *Sci Robot*, 3(20):eaat0430. https://doi.org/10.1126/scirobotics.aat0430

Huang J, Cao M, Zhou N, et al., 2017. Distributed behavioral control for second-order nonlinear multi-agent systems. *IFAC-PapersOnLine*, 50(1):2445-2450. https://doi.org/10.1016/j.ifacol.2017.08.407

Huang J, Zhou N, Cao M, 2019. Adaptive fuzzy behavioral control of second-order autonomous agents with prioritized missions: theory and experiments. *IEEE Trans Ind Electron*, 66(12):9612-9622. https://doi.org/10.1109/TIE.2019.2892669

Huang J, Mo ZB, Zhang ZY, et al., 2022a. Behavioral control task supervisor with memory based on reinforcement learning for human–multi-robot coordination systems. *Front Inform Technol Electron Eng*, 23(8):1174-1188. https://doi.org/10.1631/FITEE.2100280

Huang J, Wu WH, Zhang ZY, et al., 2022b. Human decision-making modeling and cooperative controller design for human–agent interaction systems. *IEEE Trans Human-Mach Syst*, 52(6):1122-1134. https://doi.org/10.1109/THMS.2022.3185333

Littman ML, 1994. Markov games as a framework for multi-agent reinforcement learning. Proc 11$^{\text{th}}$ Int Conf on Machine Learning, p.157-163. https://doi.org/10.1016/b978-1-55860-335-6.50027-1

Liu DR, Xue S, Zhao B, et al., 2021. Adaptive dynamic programming for control: a survey and recent advances. *IEEE Trans Syst Man Cybern Syst*, 51(1):142-160. https://doi.org/10.1109/TSMC.2020.3042876

Liu Y, Li HY, Lu RQ, et al., 2022. An overview of finite/fixed-time control and its application in engineering systems. *IEEE/CAA J Autom Sin*, 9(12):2106-2120. https://doi.org/10.1109/JAS.2022.105413

Marino A, Caccavale F, Parker LE, et al., 2009. Fuzzy behavioral control for multi-robot border patrol. Proc 17$^{\text{th}}$ Mediterranean Conf on Control and Automation, p.246-251. https://doi.org/10.1109/med.2009.5164547

Marino A, Parker LE, Antonelli G, et al., 2013. A decentralized architecture for multi-robot systems based on the null-space-behavioral control with application to multi-robot border patrolling. *J Intell Robot Syst*, 71(3):423-444. https://doi.org/10.1007/s10846-012-9783-5

Ott C, Dietrich A, Albu-Schäffer A, 2015. Prioritized multi-task compliance control of redundant manipulators. *Automatica*, 53:416-423. https://doi.org/10.1016/j.automatica.2015.01.015

Santos MCP, Rosales CD, Sarcinelli-Filho M, et al., 2017. A novel null-space-based UAV trajectory tracking controller with collision avoidance. *IEEE/ASME Trans Mech*, 22(6):2543-2553. https://doi.org/10.1109/tmech.2017.2752302

Schlanbusch R, Kristiansen R, Nicklasson PJ, 2011. Spacecraft formation reconfiguration with collision avoidance. *Automatica*, 47(7):1443-1449. https://doi.org/10.1016/j.automatica.2011.02.014

Vadakkepat P, Miin OC, Peng X, et al., 2004. Fuzzy behavior-based control of mobile robots. *IEEE Trans Fuzzy Syst*, 12(4):559-565. https://doi.org/10.1109/TFUZZ.2004.832536

Wang WJ, Li CJ, Guo YN, 2021. Relative position coordinated control for spacecraft formation flying with obstacle/collision avoidance. *Nonl Dyn*, 104(2):1329-1342. https://doi.org/10.1007/s11071-021-06348-9

Wang ZY, Schaul T, Hessel M, et al., 2016. Dueling network architectures for deep reinforcement learning. Proc 33$^{\text{rd}}$ Int Conf on Machine Learning, p.1995-2003.

Wei EM, Luke S, 2016. Lenient learning in independent-learner stochastic cooperative games. *J Mach Learn Res*, 17(1):2914-2955.

Wen GX, Chen CLP, Liu YJ, et al., 2017. Neural network-based adaptive leader-following consensus control for a class of nonlinear multiagent state-delay systems. *IEEE Trans Cybern*, 47(8):2151-2160. https://doi.org/10.1109/TCYB.2016.2608499

Wen GX, Chen CLP, Feng J, et al., 2018. Optimized multi-agent formation control based on an identifier–actor–critic reinforcement learning algorithm. *IEEE Trans Fuzzy Syst*, 26(5):2719-2731. https://doi.org/10.1109/TFUZZ.2017.2787561

Wen GX, Chen CLP, Ge SS, 2021. Simplified optimized backstepping control for a class of nonlinear strict-feedback systems with unknown dynamic functions. *IEEE Trans Cybern*, 51(9):4567-4580. https://doi.org/10.1109/TCYB.2020.3002108

Yao DY, Li HY, Lu RQ, et al., 2020. Distributed sliding-mode tracking control of second-order nonlinear multi-agent systems: an event-triggered approach. *IEEE Trans Cybern*, 50(9):3892-3902. https://doi.org/10.1109/TCYB.2019.2963087

Yao P, Wei YX, Zhao ZY, 2022. Null-space-based modulated reference trajectory generator for multi-robots formation in obstacle environment. *ISA Trans*, 123:168-178. https://doi.org/10.1016/j.isatra.2021.05.033

Zhang ZY, Mo ZB, Chen YT, et al., 2022. Reinforcement learning behavioral control for nonlinear autonomous system. *IEEE/CAA J Autom Sin*, 9(9):1561-1573. https://doi.org/10.1109/JAS.2022.105797

Zheng CB, Pang ZH, Wang JX, et al., 2023. Null-space-based time-varying formation control of uncertain nonlinear second-order multiagent systems with collision avoidance. *IEEE Trans Ind Electron*, 70(10):10476-10485. https://doi.org/10.1109/TIE.2022.3217585

Zhou N, Xia YQ, Wang ML, et al., 2015. Finite-time attitude control of multiple rigid spacecraft using terminal sliding mode. *Int J Robust Nonl Contr*, 25(12):1862-1876. https://doi.org/10.1002/rnc.3182

Zhou N, Cheng XD, Sun ZQ, et al., 2022. Fixed-time cooperative behavioral control for networked autonomous agents with second-order nonlinear dynamics. *IEEE Trans Cybern*, 52(9):9504-9518. https://doi.org/10.1109/TCYB.2021.3057219

List of supplementary materials