TYPE Original Research
PUBLISHED 28 March 2023
DOI 10.3389/fcteg.2023.989232

OPEN ACCESS

EDITED BY
Maria Guinaldo,
National University of Distance Education (UNED), Spain

REVIEWED BY
Kecai Cao,
Nantong University, China
Ernesto Aranda-Escolastico,
National University of Distance Education (UNED), Spain

*CORRESPONDENCE
Aiyi Li,
li-aiyi@ist.osaka-u.ac.jp

SPECIALTY SECTION
This article was submitted to Networked Control, a section of the journal Frontiers in Control Engineering

RECEIVED 08 July 2022
ACCEPTED 14 February 2023
PUBLISHED 28 March 2023

CITATION
Li A, Ogura M and Wakamiya N (2023), Communication-free shepherding navigation with multiple steering agents. *Front. Control. Eng.* 4:989232.
doi: 10.3389/fcteg.2023.989232

# Communication-free shepherding navigation with multiple steering agents

Aiyi Li*, Masaki Ogura and Naoki Wakamiya

Graduate School of Information Science and Technology, Osaka University, Suita, Japan

Flocking guidance addresses a challenging problem considering the navigation and control of a group of passive agents. To solve this problem, shepherding offers a bio-inspired technique for navigating such a group of agents using external steering agents with appropriately designed movement law. Although most shepherding research is mainly based on the availability of centralized instructions, these assumptions are not realistic enough to solve some emerging application problems. Therefore, this paper presents a decentralized shepherding method where each steering agent makes movements based on its own observation without any inter-agent communication. Our numerical simulations confirm the effectiveness of the proposed method by showing its high success rate and low costs in various placement patterns. These advantages particularly improve with the increase in the number of steering agents. We also confirm the robustness and resilience properties of the proposed method *via* numerical simulations.

KEYWORDS

**herding, shepherding, decentralization, multi-agent system (MAS), non-linear dynamics**

## 1 Introduction

Shepherding problem (Long et al., 2020) refers to the problem of designing the movement law of steering agents (called shepherds) to navigate another set of agents (called sheep) driven by the repulsive force from steering agents and has been attracting emerging attention by its applicability in robotics (Chung et al., 2018), group dynamics (Vemula et al., 2018), and nanochemistry (Mou et al., 2020). Against the development of leader-following control (Consolini et al., 2008; Qu et al., 2021), several works of shepherding research have been published toward providing effective solutions to the shepherding problem within various scientific fields including the systems and control theory (Bacon and Olgac, 2012; Pierson and Schwager, 2018), robotics (Zhi and Lien, 2021), and the complexity science (El-Fiqi et al., 2020).

One of the major challenges in the shepherding problem is clarifying how to coordinate *multiple* steering agents for effective guidance. In this direction, various methodologies and algorithms have been proposed in the literature. For example, Lien et al. (2005) have illustrated the effectiveness of the navigation by steering agents taking a prescribed formation. Extending this work, Pierson and Schwager (2018) have presented a 3-D herding algorithm based on the dimension reduction of the whole multi-agent system. Similar works can be found in (Song et al., 2021) and (Chipade and Panagou, 2020), where caging-based algorithms for guiding a flock of agents are proposed. El-Fiqi et al. (2020) have presented a centralized shepherding algorithm that assigns a path to each steering agent. Bacon and Olgac (2012) have presented a quasi-decentralized control law for guiding agents with multiple steering agents based on sliding mode control.

Most of the existing shepherding algorithms with multiple steering agents assume the existence of a central coordinator (Lien et al., 2005; Bacon and Olgac, 2012; Pierson and Schwager, 2018; Chipade and Panagou, 2020; El-Fiqi et al., 2020; Song et al., 2021). This assumption requires the coordinator's ability to observe the whole system and the steering agents' ability of communication. However, these requirements can severely limit the practical feasibility of the algorithms. Although we can find in the literature a few decentralized shepherding algorithms with multiple steering agents, these works still implicitly assume the communication among steering agents. For example, the shepherding algorithm proposed by (Lee and Kim, 2017) requires that a shepherd can know the intention of another shepherd, which is hard to realize without communication between shepherds. Also, in the shepherding algorithm developed by (Hu et al., 2020), the shepherds initially need to perform multiple rounds of communications for executing a distributed clustering algorithm to reach a consensus on which sub-flock is shepherded by which shepherd.

The objective of this paper is to propose an algorithm for communication-free shepherding navigation with multiple steering agents. Our approach is to start from an existing single-shepherd algorithm called Farthest-Agent Targeting algorithm (Tsunoda et al., 2018). Leveraging on the simplicity of the algorithm, we then construct an algorithm for the shepherding by multiple steering agents under the assumption that each shepherd knows its relative position to the goal, and the relative position of other agents within the shepherd's recognition range. Within the proposed algorithm, although each shepherd attempts to guide the whole flock by chasing its own target sheep independently and without inter-shepherd communication, cooperative behavior emerges as a consequence of the spatial distribution of shepherds induced by the inter-shepherd repulsion built into the algorithm. A shepherd's target sheep is determined as the sheep maximizing the weighted difference between the sheep's distance from the goal and the one from the shepherd. The effectiveness of the proposed algorithm is illustrated with extensive numerical simulations.

This paper is organized as follows. In Section 2, we state the problem studied in this paper. In Section 3, we describe our communication-free shepherding algorithm. We present the numerical simulations in Section 4, compare the performance with centralized shepherding algorithms in Section 5, and discuss the robustness and resilience of the proposed algorithm in Section 6.

## 2 Problem statement

We consider the situation where there exist $N$ agents to be navigated, called sheep, and $M$ steering agents, called shepherds. Throughout this paper, we use the notations $[N] = \{1, 2, \dots, N\}$ and $[M] = \{1, 2, \dots, M\}$. The sheep and shepherd are assumed to dynamically move on the two-dimensional space $\mathbb{R}^2$ in the discrete-time. For any $i \in [N]$, $k \in [M]$, and $t = 0, 1, 2, \dots$, we let $p_i(t)$ ($q_k(t)$) denote the position of the $i$th sheep ($k$th shepherd, respectively) at time $t$. We suppose that each sheep has a limited range $r > 0$ of recognizing other agents. Therefore, the indices of the sheep and the shepherd agents that can be recognized by the $i$th sheep at time $t$ are given by

$$\mathcal{N}_i(t) = \left\{ j \in [N] \mid 0 < \|p_i(t) - p_j(t)\| < r \right\}$$

and

$$\mathcal{M}_i(t) = \{\ell \in [M] \mid 0 < \|p_i(t) - q_\ell(t)\| < r\}.$$

In this paper, we adopt one of the standard models (Lee and Kim, 2017; Tsunoda et al., 2018) of the dynamics of the sheep. Within the model, the movement of the $i$th sheep is described as

$$p_i(t+1) = p_i(t) + u_i(t)$$

where $u_i(t) \in \mathbb{R}^2$ represents the movement vector of the $i$th sheep and is supposed to be of the form

$$u_i(t) = c_1 u_{i1}(t) + c_2 u_{i2}(t) + c_3 u_{i3}(t) + c_4 u_{i4}(t) + c_5 u_{i5}(t) \tag{1}$$

as illustrated in Figure 1. In this equation, the vectors $u_{i1}(t)$, $u_{i2}(t)$, and $u_{i3}(t) \in \mathbb{R}^2$ are the forces of separation, alignment, and cohesion given by

$$\begin{aligned} u_{i1}(t) &= -|\mathcal{N}_i(t)|^{-1} \sum_{j \in \mathcal{N}_i(t)} \psi\big(p_j(t) - p_i(t)\big), \\ u_{i2}(t) &= |\mathcal{N}_i(t)|^{-1} \sum_{j \in \mathcal{N}_i(t)} \phi\big(u_j(t-1)\big), \\ u_{i3}(t) &= |\mathcal{N}_i(t)|^{-1} \sum_{j \in \mathcal{N}_i(t)} \phi\big(p_j(t) - p_i(t)\big) \end{aligned} \tag{2}$$

where $\phi(x) = x/\|x\|$ is a normalization operator and

$$\psi(x) = x/\|x\|^3 \tag{3}$$

is a potential-like function. We remark that these three forces motivated by the seminal work by Reynolds (1987) are not implemented in its original manner. For example, although the original definition of the cohesion force steers an agent to move toward the average position of local neighbors, the current implementation takes average after performing normalization by $\phi$. We further remark that, to avoid ill-posedness of the model, we extend the domain of these mappings to the whole space $\mathbb{R}^2$ by letting $\phi(0) = 0$ and $\psi(0) = 0$. The reason for using the normalization $\psi$ only for the separation force is to ensure that a sheep agent does not collide with other sheep. Because $\psi(x)$ diverges as $x$ tends to zero, the separation force becomes dominant as two agents become close and, hence, the inter-agent distance tends to zero. On the other hand, when the inter-agent distance is relatively large, the separation term becomes negligible and the cohesion term becomes dominant. This approach has been taken in various swarm models in the literature (Parrish et al., 2002; Gazi and Passino, 2003).

Also, the vector $u_{i4}(t)$ is the force of repulsion from shepherd agents given by

$$u_{i4}(t) = -|\mathcal{M}_i(t)|^{-1} \sum_{\ell \in \mathcal{M}_i(t)} \psi(q_\ell(t) - p_i(t)),$$

in which we use the normalization $\psi$ to avoid a collision with a shepherd. We remark that, when the set $\mathcal{N}_i(t)$ is empty, we regard the vectors in Eq. 2 to be the zero vectors. Likewise, if the set $\mathcal{M}_i(t)$ is empty, then we set $u_{i4}(t) = 0$. This rule applies to other similar equations appearing later in this paper. Finally, $u_{i5}(t)$ is a random vector accounting for disturbances; its norm and angle are independently drawn from a uniform distribution on the intervals $[0, 1]$ and $[0, 2\pi]$, respectively.

As for the shepherd agents, we place the following restriction on the information available for navigation. The recognition range of a shepherd is assumed to be finite and is set to be $r' > 0$; therefore, the set of indices of the sheep and the shepherd agents that can be recognized by the $k$th shepherd at time $t$ are given by

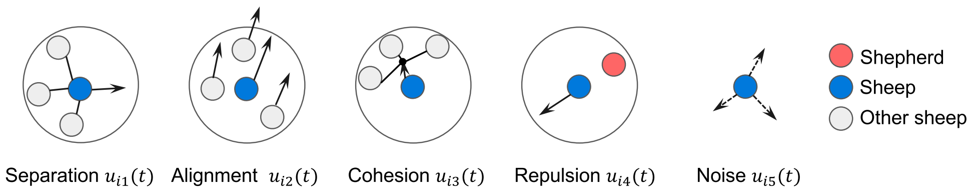


FIGURE 1
Schematic of the sheep model described in Eq. 1. A sheep receives separation, alignment, cohesion from other sheep and repulsion from shepherds within the sensing range. The sheep is also exposed to noise.

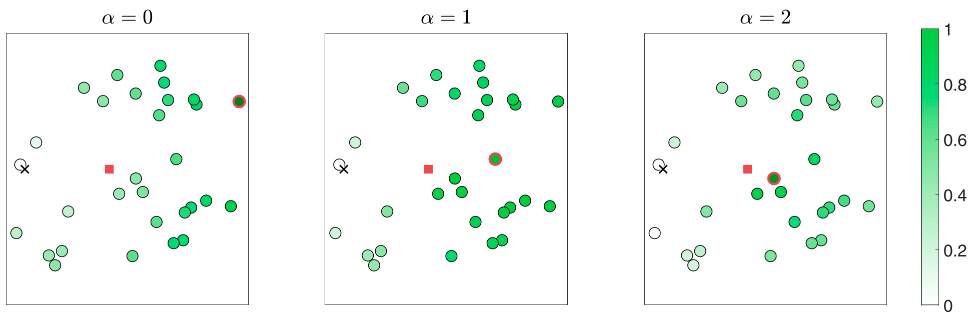


FIGURE 2
Target selection by the proposed algorithm. Goal: X; Sheep: circles; Shepherd: square. For each value of $\alpha$, we color each sheep $i$ using the [0,1]-normalized value of the objective function $\|p_i - x_g\| - \alpha\|p_i - q_k(t)\|$. When $\alpha = 0$, the shepherd targets the sheep farthest from the goal. On the other hand, for larger $\alpha$, the shepherd targets a sheep far from the goal and close to the shepherd.

$$\mathcal{N}_k'(t) = \left\{j \in [N] \mid 0 < \|q_k(t) - p_j(t)\| < r'\right\}$$

and

$$\mathcal{M}_k'(t) = \{\ell \in [M] \mid 0 < \|q_k(t) - q_\ell(t)\| < r'\}.$$

We further assume that a shepherd can detect only its relative position to the goal and the relative position of other agents within the shepherd's recognition range. Hence, the $k$th shepherd needs to determine its own movement at time $t$ based only on the following types of vectors.

- $p_j(t) - q_k(t)$ $(j \in \mathcal{N}_k'(t))$;
- $q_\ell(t) - q_k(t)$ $(\ell \in \mathcal{M}_k'(t))$;
- $x_g - q_k(t)$.

We remark that, the availability of the first two types of information is widely assumed in the literature, and could be realized with proximity sensors. Also, the sensing of the third type of information can be realized by recent sensing methodologies such as single beacon-based positioning systems (Hou et al., 2016).

Finally, the objective of the navigation by shepherd agents is to herd all the sheep into a goal region $G \subset \mathbb{R}^2$. In this paper, we suppose that the goal region $G$ is the closed disk with center $x^g \in \mathbb{R}^2$ and radius $R^g > 0$.

**Remark 2.1.** In this paper, we do not aim to analytically establish the effectiveness of the shepherding algorithm to be proposed in the next section. The major reason for this choice is its intrinsic difficulty arising from the non-linearity of the Boid model. In fact, several existing works (see, e.g., Lien et al., 2005; Bacon and Olgac, 2012; Pierson and Schwager, 2018; Chipade and Panagou, 2020; El-Fiqi et al., 2020; Song et al., 2021; Lee and Kim, 2017) on the shepherding problem do not provide a mathematical proof for the performance of the proposed control methodologies. This tendency is a common practice in the field of swarm guidance, as the non-linearity of the swarm model often makes it challenging to perform a meaningful mathematical analysis.

## 3 Proposed algorithm

In this section, we describe the algorithm that we propose for the movement of the shepherd agents. We start by recalling the Farthest-Agent Targeting (FAT) algorithm (Tsunoda et al., 2018) designed for the case of a single shepherd (i.e., $M = 1$). In the algorithm, the movement of the (first) shepherd is specified as $q_1(t+1) = q_1(t) + v_1(t)$, where $v_1(t) \in \mathbb{R}^2$ represents the movement vector of the shepherd. Let us denote the position of the sheep agent farthest from the goal by $\xi_1(t)$; i.e., define

$$\xi_1(t) = \underset{p \in \{p_j(t)\}_{j \in [N]}}{\arg\max} \|p - x_g\|.$$

Then, in the FAT algorithm, the movement vector $v_1(t)$ is specified as the weighted sum of the following three vectors:

$$\phi(\xi_1(t) - q_1(t)),\ -\psi(\xi_1(t) - q_1(t)),\ -\phi\big(x_g - q_1(t)\big), \tag{4}$$

which are, respectively, to realize the movement of the shepherd for chasing the farthest agent, taking an appropriate distance with the farthest agent, and pushing the farthest agent toward the goal region. As for the second term, the term allows us to realize an appropriate, non-vanishing distance for the same reason that the normalization $\psi$ in the sheep model allows a sheep to avoid a collision. Despite being simple, the FAT algorithm is known for its effectiveness in performing the shepherding navigation with a single shepherd (Tsunoda et al., 2018). However, the algorithm requires knowledge of the positions of all sheep. Furthermore, when generalized to the situation of multiple shepherds, the formula would result in all the shepherds targeting the same sheep, which is presumably inefficient.

Based on these observations, in this paper, we propose an extended version of the FAT algorithm to let each shepherd choose, as its target, a sheep both close to itself and far from the goal. Specifically, we propose that the sheep agent targeted by the $k$th shepherd is determined by the formula

$$\xi_k(t) = \underset{p \in \{p_j(t)\}_{j \in \mathcal{N}'k(t)}}{\arg\max} \big(\|p - x_g\| - \alpha\|p - q_k(t)\|\big), \tag{5}$$

where $\alpha \geq 0$ is the parameter determining the behavior of shepherds within the proposed algorithm. For example, when $\alpha = 0$, only the first term $\|p - x_g\|$ remains in Eq. 5 and, therefore, all shepherds target the sheep farthest from the goal; i.e., the proposed algorithm reduces to the FAT algorithm. On the other hand, when $\alpha$ is sufficiently large, each shepherd chooses the closest sheep as its target, which specifically prevents the scattering phenomenon caused by the FAT algorithm, as illustrated in Figure 2. Hence, we can expect that choosing a moderate value of $\alpha$ would result in a control strategy that is as effective as the FAT algorithm and is less suffered from the scattering phenomenon. We here emphasize that $\xi_k(t)$ is decidable by the $k$th shepherd because the sheep's relative position to the goal is computable as $p_j(t) - x_g = (p_j(t) - q_k(t)) + (x_g - q_k(t))$.

We can now state the proposed movement algorithm of the shepherds. As in the FAT algorithm, we let

$$q_k(t+1) = q_k(t) + v_k(t),$$

where $v_k(t)$ denotes the movement vector of the $k$th shepherd. This vector is to be constructed as the weighted sum of the following four vectors. First, we define

$$v_{k1}(t) = \phi(\xi_k(t) - q_k(t))$$

for the $k$th shepherd to chase the target sheep. Secondly, in order to take an appropriate distance between the shepherd and sheep agents, we define

$$v_{k2}(t) = -|\mathcal{N}'k(t)|^{-1} \sum_{j \in \mathcal{N}'_{k(t)}} \psi\big(p_j(t) - q_k(t)\big) \tag{6}$$

so that the $k$th shepherd receives repulsion force from all the neighboring sheep agents. Thirdly, to achieve guidance toward the goal region, we define the vector

$$v_{k3}(t) = -\phi\big(x_g - q_k(t)\big)$$

by adopting Eq. 4. Finally, in order to avoid competition among shepherd agents for efficient guidance, we introduce the vector

$$v_{k4}(t) = -\|x_g - q_k(t)\|\,|\mathcal{M}'_k(t)|^{-1} \sum_{\ell \in \mathcal{M}'_k(t)} \psi\big(q_\ell(t) - q_k(t)\big), \tag{7}$$

which represents repulsion between shepherd agents. Because shepherds need to be relatively closer to each other at the final stage of the shepherding navigation, we introduce the weight term $\|x_g - q_k(t)\|$. Now, based on the four vectors introduced above, we define the movement vector of the $k$th shepherd as

$$v_k(t) = d_1 v_{k1}(t) + d_2 v_{k2}(t) + d_3 v_{k3}(t) + d_4 v_{k4}(t) \tag{8}$$

for positive constants $d_1$, $d_2$, $d_3$, and $d_4$.

**Remark 3.1.** Let us discuss the communication costs of the proposed algorithm and existing distributed shepherding algorithms with multiple shepherds (Lee and Kim, 2017; Hu et al., 2020). The proposed algorithm does not require communication between shepherds in the sense that each shepherd requires only its relative position with other shepherds, which can be accomplished with its own sensing devices. On the other hand, as discussed in Section 1, the existing algorithms require communication between shepherd agents because each agent needs to understand the intention of other shepherds. Specifically, the algorithm by (Lee and Kim, 2017) can require $O(n_f M^2)$ times of communications between shepherd agents at each time step, where $n_f$ denotes the number of sub-flocks of sheep. Also, within the algorithm presented by (Hu et al., 2020), in order to execute a clustering algorithm for determining which sub-flock is chased by which shepherd, $O(n_f M)$ times of communications needs to be periodically performed between shepherds.

## 4 Numerical simulation

In this section, we present numerical simulations to illustrate the effectiveness of the proposed algorithm. Throughout the simulations, all the parameters and variables are assumed to be dimensionless.

### 4.1 Configuration

We assume that there exists $N$ sheep to be guided in two-dimension space. We suppose that, at the initial time, $N$ sheep are placed uniformly and randomly on the disc centered at the origin and having an initial radius $R_s$ (0). We design the pattern of the initial distribution as 1) a small flock: $N = 20$, $R_s(0) = 40$, 2) a large flock: $N = 50$, $R_s(0) = 60$, and 3) two separate flocks: $N = 20$, $R_s(0) = 40$ for one flock and $N = 30$, $R_s(0) = 50$ for another flock. The parameters of the sheep model are set as

$$c_1 = 200,\ c_2 = 0.02,\ c_3 = 0.2,\ c_4 = 400,\ c_5 = 0.1 \tag{9}$$

and $r = 50$. We call this parameter set or scenario as *baseline*. The goal $G$ is supposed to have the center $x^g = [150,150]^\top$ and radius $R^g = 80$. For the comprehensiveness of our experiment, we prepare the following three different placement patterns of the shepherd agents; shepherd agents are initially 1) placed around at the bottom-left of the sheep agents (*bottom-left*), 2) placed around at the top-right of the sheep agents (*top-right*), and 3)

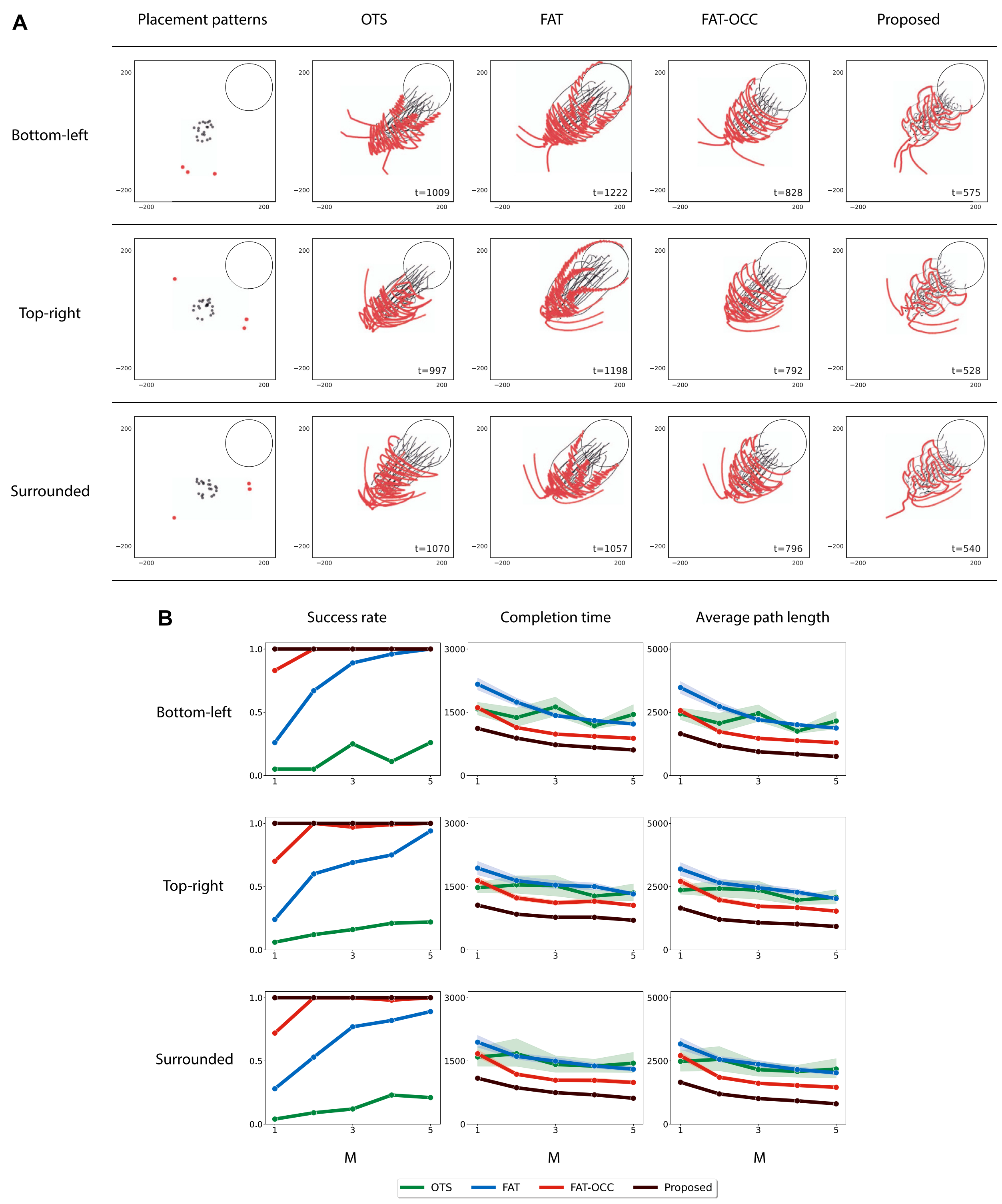


**FIGURE 3**
Herding a small flock in the baseline sheep model. **(A)** Initial configurations and corresponding trajectories. The number of shepherds is set to $M = 3$. First column: Samples of initial configurations. Second to fifth columns: Trajectories of the quickest navigations among those performed for randomly generated 100 initial configurations. Circle: goal region. Red dots: shepherds. Gray dots: sheep. The numbers at the bottom-right indicate the time at which the shepherding navigation is completed. It is remarked that the initial configurations in each row are not necessarily the same. A movie can be found at https://youtu.be/AlrRmSfvYcs. **(B)** Performances of the four algorithms. Horizontal axes represent the number of shepherds $M$ from 1 to 5. First column: the rate of successful navigation. Second column: success time. Third column: average traversal distance of shepherds. In the second and third columns, a solid line draws an estimate of the mean value and shaded areas describe the confidence interval for that estimate.

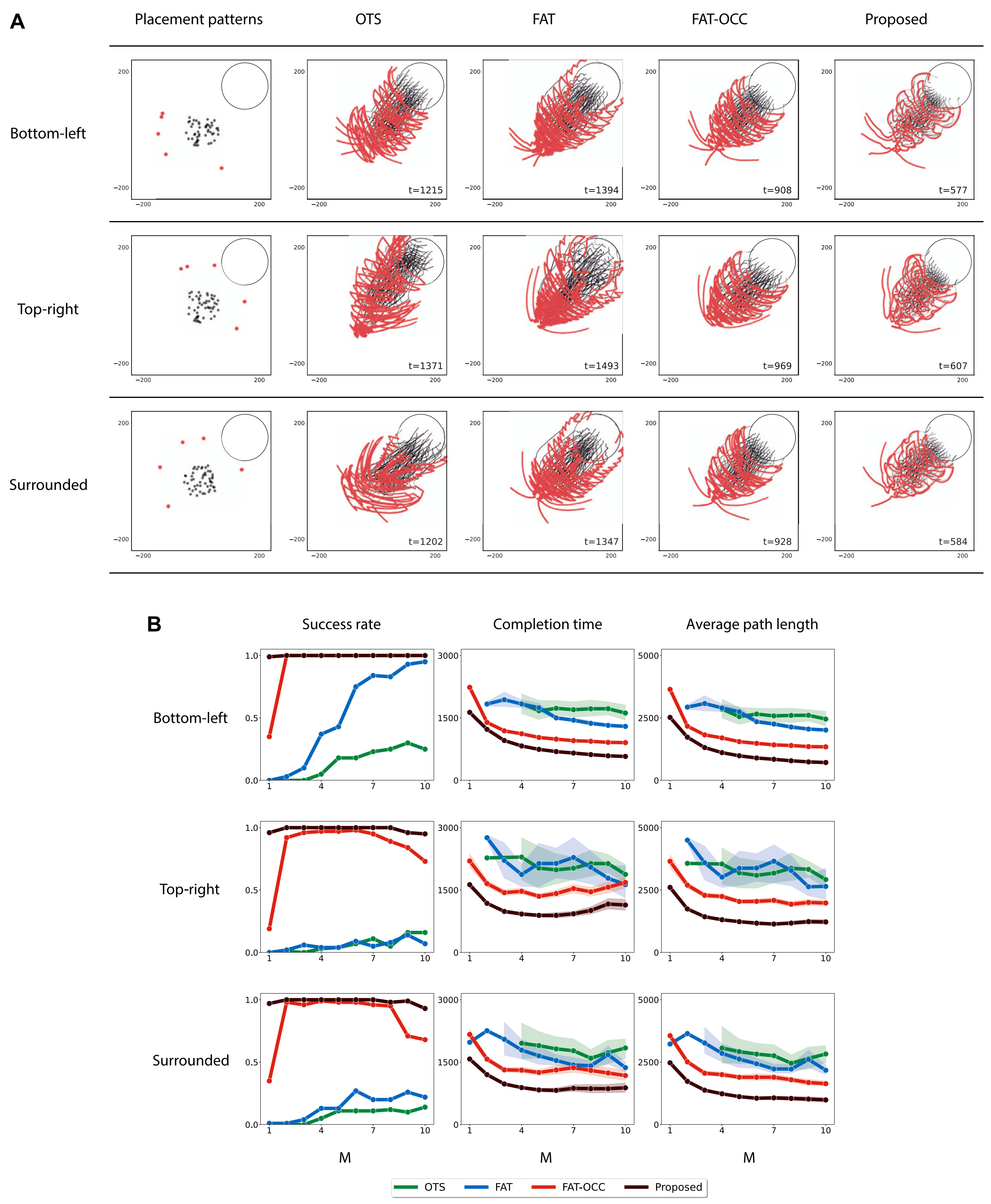


FIGURE 4
Herding a large sheep flock in the baseline sheep model. **(A)** Initial configurations and corresponding trajectories. The number of shepherds is set to $M = 5$. A movie can be found at https://youtu.be/AlrRmSfvYcs. **(B)** Performances of the four algorithms. Horizontal axes represent the number of shepherds $M$ from 1 to 10.

surrounding the sheep agents (*surrounding*). For each of the placement patterns, we randomly generate 100 different initial configurations of agents. Samples of the initial configurations are shown in the first column of Figure 3A. Furthermore, in the numerical simulations, we modify the operator $\psi$ in the original model as

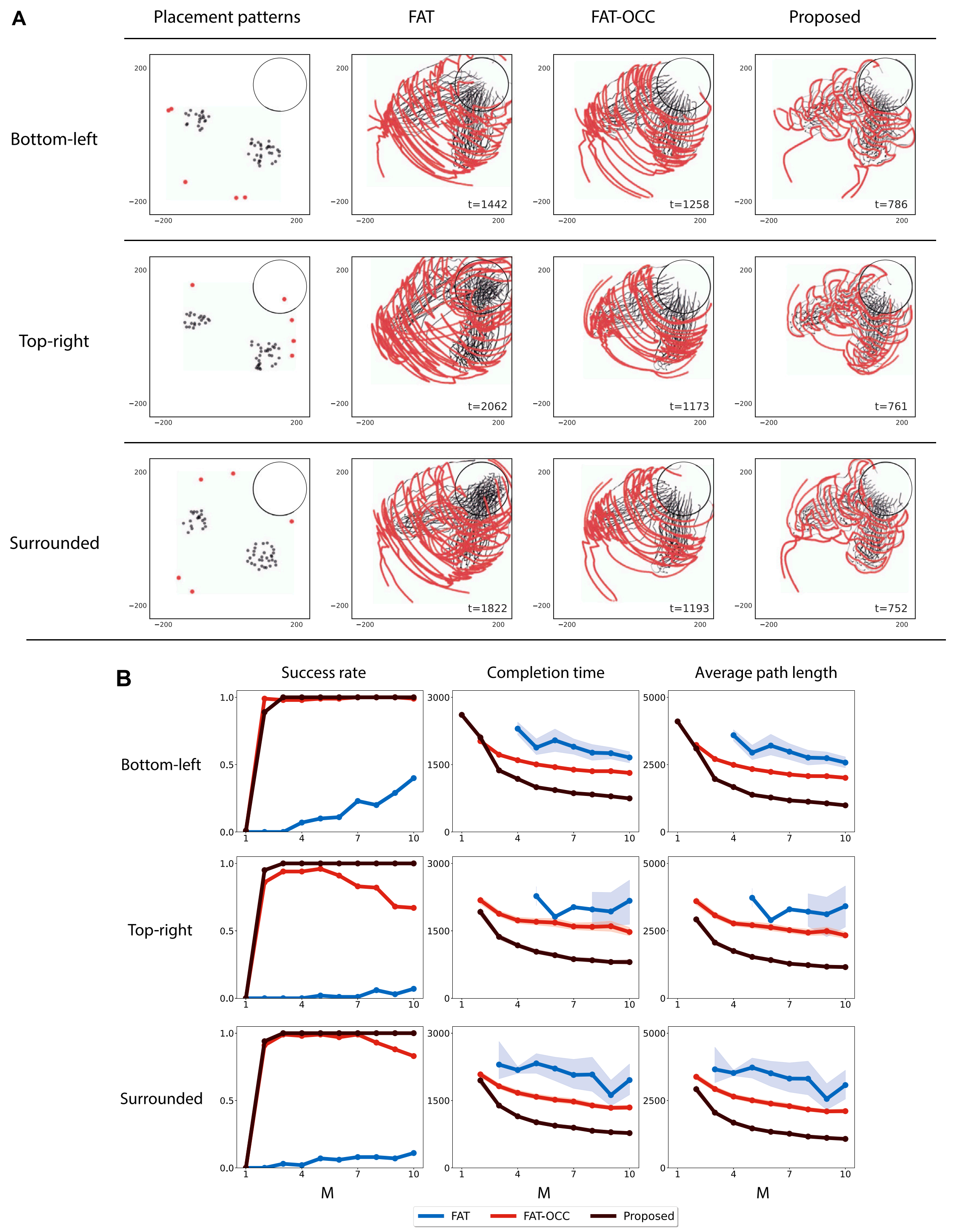


**FIGURE 5**
Herding two separate flocks in the baseline sheep model. **(A)** Initial configurations and corresponding trajectories. The number of shepherds is set to $M = 5$. A movie can be found at https://youtu.be/AIrRmSfvYcs. **(B)** Performances of the three algorithms. Horizontal axes represent the number of shepherds $M$ from 1 to 10.

$$\psi(x) = \begin{cases} x/\|x\|^3, & \text{if } \|x\| \geq r, \\ x/\left(\|x\|r^2\right), & \text{if } 0 < \|x\| < r, \\ 0, & \text{otherwise} \end{cases}$$

to uniformly bind the speed of the agents by avoiding numerical instability of the operator $\psi$ in its original definition Eq. 3. In our numerical simulations, we use $r = 3$. This value allows us to avoid

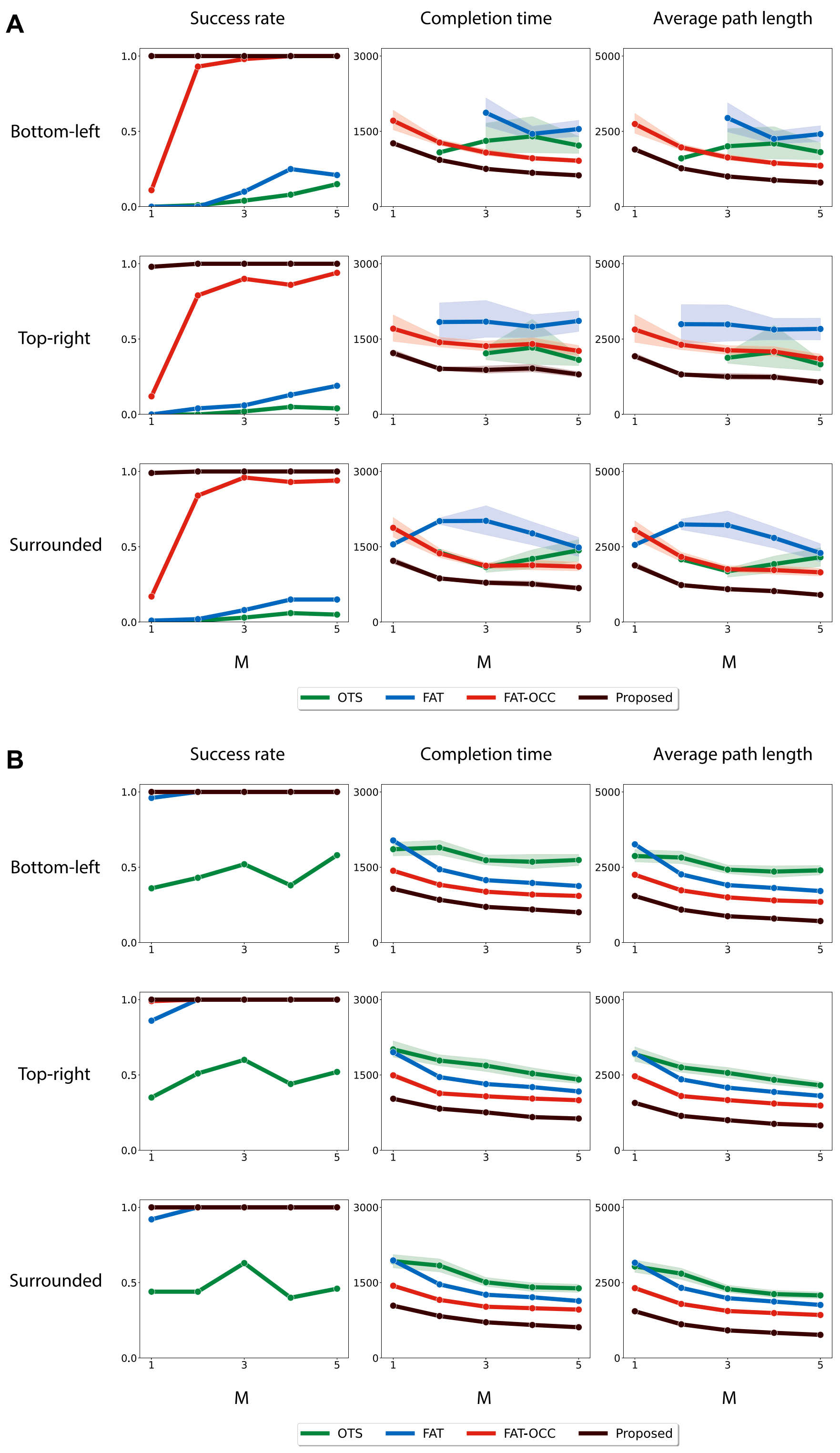


**FIGURE 6**
Herding a small flock in the sensitive and insensitive sheep model separately. **(A)** Performances of the four algorithms in the sensitive sheep model. Horizontal axes represent the number of shepherds *M* from 1 to 5. **(B)** Performances of the four algorithms in the insensitive sheep model. Horizontal axes represent the number of shepherds *M* from 1 to 5.

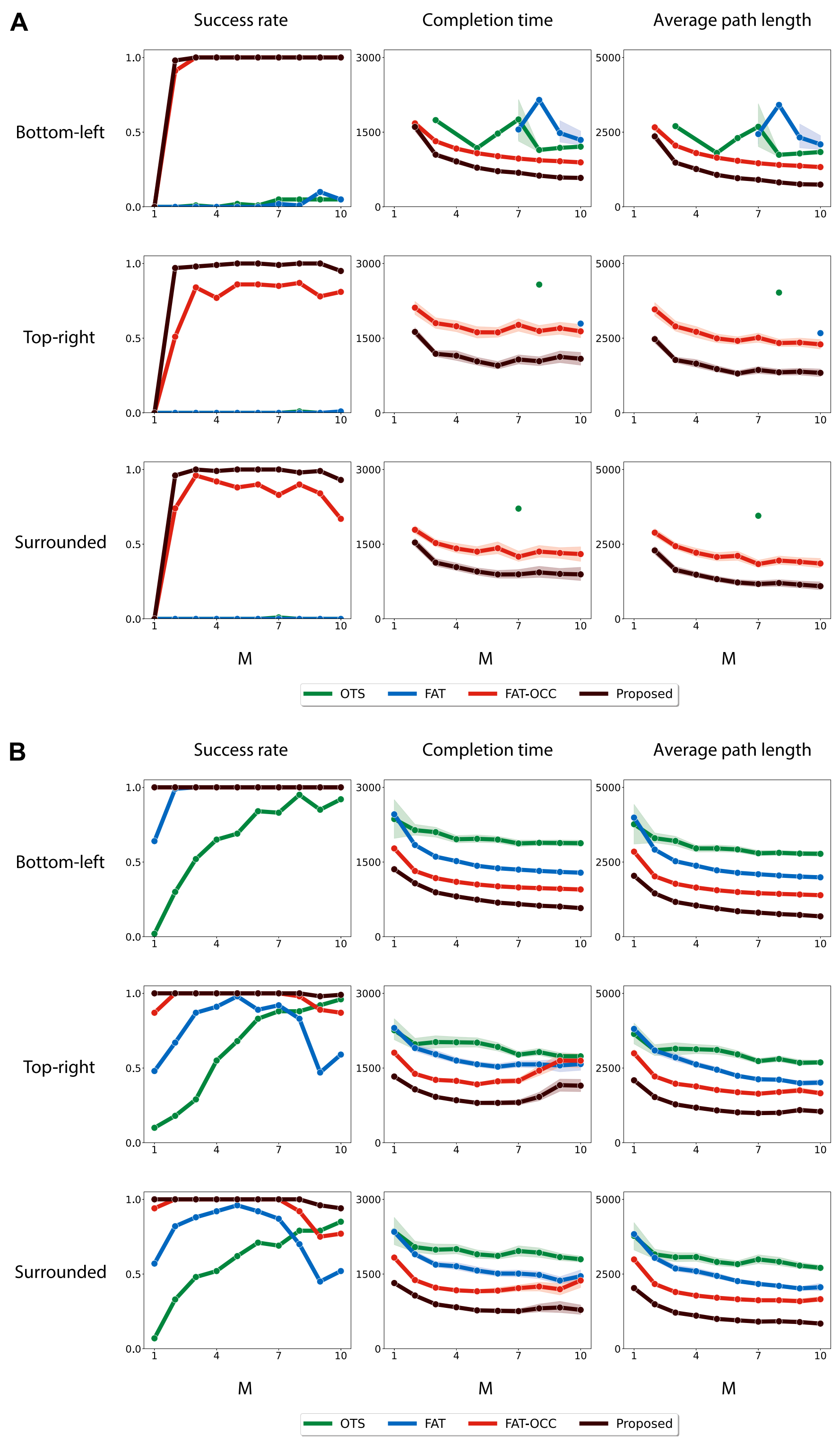


**FIGURE 7**
Herding a large flock in the sensitive and insensitive sheep model separately. **(A)** Performances of the three algorithms in the sensitive sheep model. Horizontal axes represent the number of shepherds $M$ from 1 to 10. **(B)** Performances of the three algorithms in the insensitive sheep model. Horizontal axes represent the number of shepherds $M$ from 1 to 10.

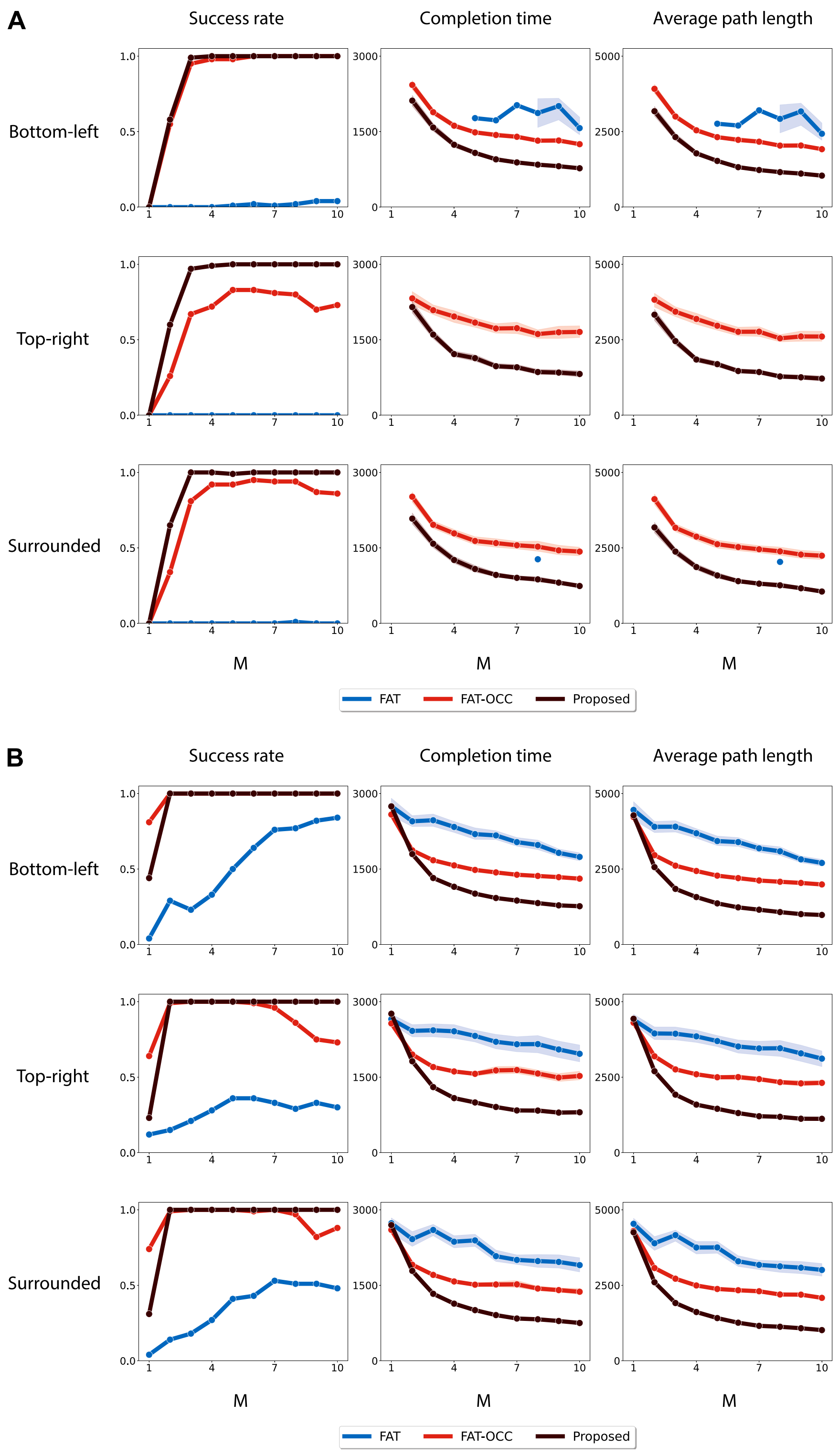


FIGURE 8
Herding two separate flocks in the sensitive and insensitive sheep model separately. **(A)** Performances of the four algorithms in the sensitive sheep model. Horizontal axes represent the number of shepherds *M* from 1 to 10. **(B)** Performances of the four algorithms in the insensitive sheep model. Horizontal axes represent the number of shepherds *M* from 1 to 10.

numerical instability while ensuring that the separation force is still dominant between agents that are close enough to each other.

## 4.2 Proposed and baseline algorithms

We describe the algorithms to be compared in our numerical simulations. For each of the initial configurations, all of the algorithms are to be terminated when all the sheep agents belong to the goal $G$, or after 3,000 steps regardless of the result of navigation. In the former case, we label the trial for the initial configuration as success.

### 4.2.1 Proposed algorithm

We use the parameters $d_1 = 2$, $d_2 = 100$, $d_3 = 1$, $d_4 = 2$, and $r' = 300$. The parameter $\alpha$ in Equation 5 is set as $\alpha = 1$.

### 4.2.2 Farthest-agent targeting

When $\alpha = 0$, Eq. 5 lets each shepherd target the sheep farthest from the goal among all the sheep in the shepherd's recognition range. For this reason, we call the algorithm with $\alpha = 0$ the Farthest-Agent Targeting (FAT) algorithm. We remark that, for the purpose of a fair comparison, the recognition range $r'$ of the algorithm is set to be equal to the one in the proposed algorithm (and, hence, to be finite), while the original Farthest-Agent Targeting algorithm (Tsunoda et al., 2018) assumes the infinite recognition range of a shepherd agent.

### 4.2.3 Farthest-agent targeting with occlusion

The farthest-agent targeting algorithm with occlusion (FAT-OCC) (Tsunoda et al., 2018) is also considered. This algorithm is identical to the FAT algorithm except that $\mathcal{N}'_{k,\mathrm{occ}(t)}$ in Eq. 5 and Eq. 6 in which $\mathcal{N}'_{k,\mathrm{occ}}(t)$ represents the set of sheep agents recognizable under occlusion and is constructed as follows. For each $t$, we first initialize $\mathcal{N}'_{k,\mathrm{occ}}(t) = \emptyset$. We then order the set $\mathcal{N}'_k(t)$ as $(i_1, \ldots, i_{|\mathcal{N}'_k(t)|})$ in such a way that $\|p_{i_1}(t)\| \le \|p_{i_2}(t)\| \le \cdots \le \|p_{i_{|\mathcal{N}'_k(t)|}}(t)\|$. For each $\iota = 1, \ldots, |\mathcal{N}'_k(t)|$, we sequentially join the index $i_\iota$ to the set $\mathcal{N}'_{k,\mathrm{occ}}(t)$ if and only if $|\angle(p_\iota - q_k) - \angle(p_\phi - q_k)| > \theta$ for all $\phi \in \mathcal{N}'_{k,\mathrm{occ}}(t)$. We use the parameter $\theta = \pi/36$.

### 4.2.4 Online-target switching

The Online-Target Switching (OTS) algorithm proposed by Strömbom et al. (2014) is applied by judging the flock separation. We implement this algorithm by replacing $\xi_k(t)$ in Eq. 5 with $\xi_k^{\mathrm{ots}}(t)$ defined by

$$\xi_k^{\mathrm{ots}}(t) = \begin{cases} \bar{p}_k(t) + d^{\mathrm{ots}}\,\phi\big(\bar{p}_k(t) - x_g\big), & \text{if } \big\|p_k^{\#}(t) - \bar{p}_k(t)\big\| \le R^{\mathrm{ots}}, \\ p_k^{\#}(t) + d^{\mathrm{ots}}\,\phi\big(p_k^{\#}(t) - \bar{p}_k(t)\big), & \text{otherwise} \end{cases} \tag{10}$$

where

$$\bar{p}_k(t) = |\mathcal{N}'_k(t)|^{-1} \sum\nolimits_{j \in \mathcal{N}'_k(t)} p_j(t) \tag{11}$$

is the mass center of the sheep flock that the shepherd $k$ can observe,

$$p_k^{\#}(t) = \underset{p \in \{p_j(t)\}_{j \in \mathcal{N}'_k(t)}}{\arg\max} \|\bar{p}_k(t) - p\| \tag{12}$$

represents the position of the sheep farthest from the mass center, and $R^{\mathrm{ots}} = r^{\mathrm{ots}}\sqrt{\mathcal{N}'_k(t)}$ determines the size of the radius based on the number of the sheep agents, which is the same setup as the original algorithm. We choose $r^{\mathrm{ots}} = 10$ and $d^{\mathrm{ots}} = 25$ so that there is an appropriate distance of $R^{\mathrm{ots}}$ and $d^{\mathrm{ots}}$ between the shepherd and the flock. In this way, the shepherd can maintain the flock shape when changing the target position $\xi_k^{\mathrm{ots}}(t)$ in Eq. 10.

## 4.3 Performance comparison

We perform simulations to illustrate the effectiveness of the proposed algorithm. Within the simulations, we perform shepherding of a flock following one of the three initial distributions of sheep and three placements of shepherds using one of the four algorithms. We illustrate the performance of the algorithms using the trajectories of agents. Toward this end, for each pair of the four algorithms and three placement patterns, we pick the quickest trial among 100 initial configurations. The trajectories and their corresponding completion time are shown in Figures 3A, 4A, and 5A. We can observe that the trajectories of the shepherds in the proposed algorithm are smoother than those of the three baseline algorithms, confirming the effectiveness of the decentralized mechanism of the proposed algorithm. For herding the two separate flocks, we found through numerical simulations that the switching algorithm is not capable of performing the shepherding task, so we only compare the three remaining algorithms in Figure 5A. We also observe the FAT algorithm can perform poorly due to the cases of initial configurations. After examining the simulation data, we identify the following problems with the FAT algorithm. One problem is that herding a large flock can consume excessive time due to long traversal distances. Another problem is that a flock tends to be scattered when the shepherd agent chases the sheep agent on the opposite side of the flock.

For further evaluation and comparison of the proposed and the baseline algorithms, we introduce the following three performance measures. First, the success rate of an algorithm for a placement pattern is defined as the rate of successful trials among randomly generated 100 initial configurations. Second, we define the completion time as the execution time of the algorithm in its successful trials. Finally, the average path length is defined as the average of the mean traveling distance $M^{-1}\sum_{k=1}^{M}\sum_t \|v_k(t)\|$ of shepherds in successful trials.

Figures 3B, 4B, and 5B show how these three performance measures depend on the number of shepherds for each of the algorithms. We observe that the proposed algorithm achieves almost 100% success rate regardless of the number of shepherds and placement patterns, which confirms the effectiveness and resilience of the proposed algorithm. We can also observe that the proposed algorithm outperforms the baseline algorithms in completion time and average path length. Furthermore, the average completion time and average path length steadily decrease with respect to the number of shepherds. These trends suggest that the proposed algorithm allows stable and synergistic coordination of shepherds for the navigation of sheep agents.

For a more thorough comparison, let us consider two other scenarios in which the parameters of the sheep are different from the ones used in previous simulations. In the first scenario, we consider the parameters

$$c_1 = 250,\ c_2 = 0.025,\ c_3 = 0.2,\ c_4 = 500,\ c_5 = 0.1.$$

These values are all no less than the corresponding ones of the baseline scenario in Eq. 9. For this reason, we expect that the sheep flock with these parameters are more sensitive to the movement of the shepherds. Let us call this scenario *sensitive*. On the other hand, we prepare the other additional scenario to perform comparisons for the sheep flock that is harder to navigate. For this reason, in the second scenario, we use the parameters

$$c_1 = 150,\ c_2 = 0.015,\ c_3 = 0.2,\ c_4 = 300,\ c_5 = 0.1.$$

Because these values are all no greater than the corresponding ones of baseline, we call this scenario *insensitive*. Now, under these two additional scenarios, we perform the same set of simulations that we did for the baseline scenario Eq. 9. The results of the simulations in the scenarios sensitive and insensitive are illustrated in Figures 6A, 7A, and 8A and Figures 6B, 7B, and 8B, respectively. We can confirm that the proposed algorithm always shows higher success rates as well as lower completion time and shorter path lengths. In Figures 7A, 7B, we also observe a slight decrease in the success rate of the proposed algorithm. After investigating the failure cases, we find the following reasons. One reason is the interference behaviors among multiple shepherd agents when shepherd agents may coincidentally choose the same target and drive the target further away from the goal without returning. Another reason is due to the behaviors of the sheep agents; when the parameters of the sheep model are changed to be more sensitive, it can be difficult for the shepherd agents to include all the sheep agents inside the goal region simultaneously.

## 5 Centralized shepherding algorithms

In this section, we compare the proposed algorithm with centralized multi-agent shepherding algorithms in terms of cooperation and shepherding performance. Our aim is to help make the decision to select the appropriate algorithm according to the balance of performance compared to communication requirements. We first describe the centralized algorithms that we use for comparison. We then present the comparison of the proposed algorithm with the centralized algorithms.

### 5.1 Point-offset circling control

Pierson and Schwager (2018) proposes a shepherding algorithm in which shepherds form an arc formation to steer the flock. Within the algorithm, at each time a central controller computes the center of the mass $\bar{p}(t)$ and the position $p^{\#}(t)$ of the sheep farthest from the center similar to Eq. 11. Then, the controller regards the flock as a circle with center $\bar{p}(t)$ and radius $R_s(t) = \|p^{\#}(t) - \bar{p}(t)\|$. The controller then generates a circle having the same center but with a larger radius, and directs each shepherd to move toward a point on the larger circle. Each shepherd is placed evenly on the larger circle.

Specifically, within the algorithm, the controller computes the following quantities at each time $t$:

$$\begin{aligned} R^{\text{circle}}(t) &= \alpha^{\text{circle}} R_s(t), \\ \Delta_k^{\text{circle}} &= \Delta \frac{(2k - M - 1)}{(2M - 2)}, \\ \theta_{\bar{p}}(t) &= \angle\left(\bar{p}(t) - x_g\right) \end{aligned}$$

where $R^{\text{circle}}(t)$ represents the radius of the larger circle whose radius is controlled by parameter $\alpha^{\text{circle}} \geq 1$, $\Delta_k^{\text{circle}}$ represents the degree of $k$th shepherd for determining its placement on the circle, and $\theta_{\bar{p}}(t)$ represents the angle of the center of the sheep flock in the counterclockwise direction with respect to the positive direction of $x$-axis. Each shepherd needs to know its index $k$ within the total number $M$ of shepherds. The controller then directs each shepherd to move toward its target position defined by

$$\xi_k^{\text{circle}}(t) = \bar{p}(t) + R^{\text{circle}}(t) \begin{bmatrix} \cos \Delta_k^{\text{circle}} + \theta_{\bar{p}}(t) \\ \sin \Delta_k^{\text{circle}} + \theta_{\bar{p}}(t) \end{bmatrix}.$$

Within our simulation, we use $\alpha^{\text{circle}} = 1.5$ and $\Delta = 2\pi/3$.

### 5.2 Potential-based caging

The shepherding algorithm proposed by Song et al. (2021) employs a caging formalism in robotic manipulation and guides a group of sheep agents to the goal region safely and with provable guarantees. The cage is constructed by a regular $M$-sided polygon and has the shepherds as its vertices. The distance between the center of the flock and the vertex is set as $R^{\text{cage}}$ determined by

$$\begin{aligned} R^{\text{cage}}(t) - R_s(t) - d_{\text{CSM}} &= R^{\text{cage}}(t)\sin(\pi/M), \\ \Delta_k^{\text{cage}} &= \frac{2k}{M}\pi, \end{aligned}$$

where $d_{CSM}$ is the minimum required distance between the sheep and the point. Then, the target position for $k$th shepherd is set as

$$\xi_k^{\text{cage}}(t) = \bar{p}(t) + R^{\text{cage}}(t) \begin{bmatrix} \cos \Delta_k^{\text{cage}} \\ \sin \Delta_k^{\text{cage}} \end{bmatrix} + \alpha^{\text{cage}} \phi(x_g - \bar{p}(t)).$$

We remark that we are introducing the term parameter $\alpha^{\text{cage}} \phi(x_g - \bar{p}(t))$ so that the algorithm can achieve guidance of the flock into the goal region. In the caging process, each shepherd moves to a vertex close to itself as its target position while making sure that no vertex is shared with multiple shepherds. We use $d_{CSM} = 0.05 R_s(0)$ and $\alpha^{\text{cage}} = 8$.

### 5.3 Performance comparison

We perform simulations to herd the sheep flock based on the sheep model presented in Equation 1. For both centralized shepherding algorithms, the dynamics of the shepherd presented in Equation 8 is set as $d_2 = 0$, $d_3 = 0$, and $d_4 = 0$. From our preliminary simulations, we found that the algorithms presented above do not perform well in some situations. Therefore, in our simulations, to make the cooperation of multiple shepherds stable, we modify the radius $R_s(t) = \min\{\|\bar{p}(t) - p^{\#}(t)\|, \beta\|\bar{p}(0) - p^{\#}(0)\|\}$ to prevent failure when the flock is dispersed during shepherding and we choose $\beta = 1.25$. Further, we define that the algorithm for point-offset circling control takes the same

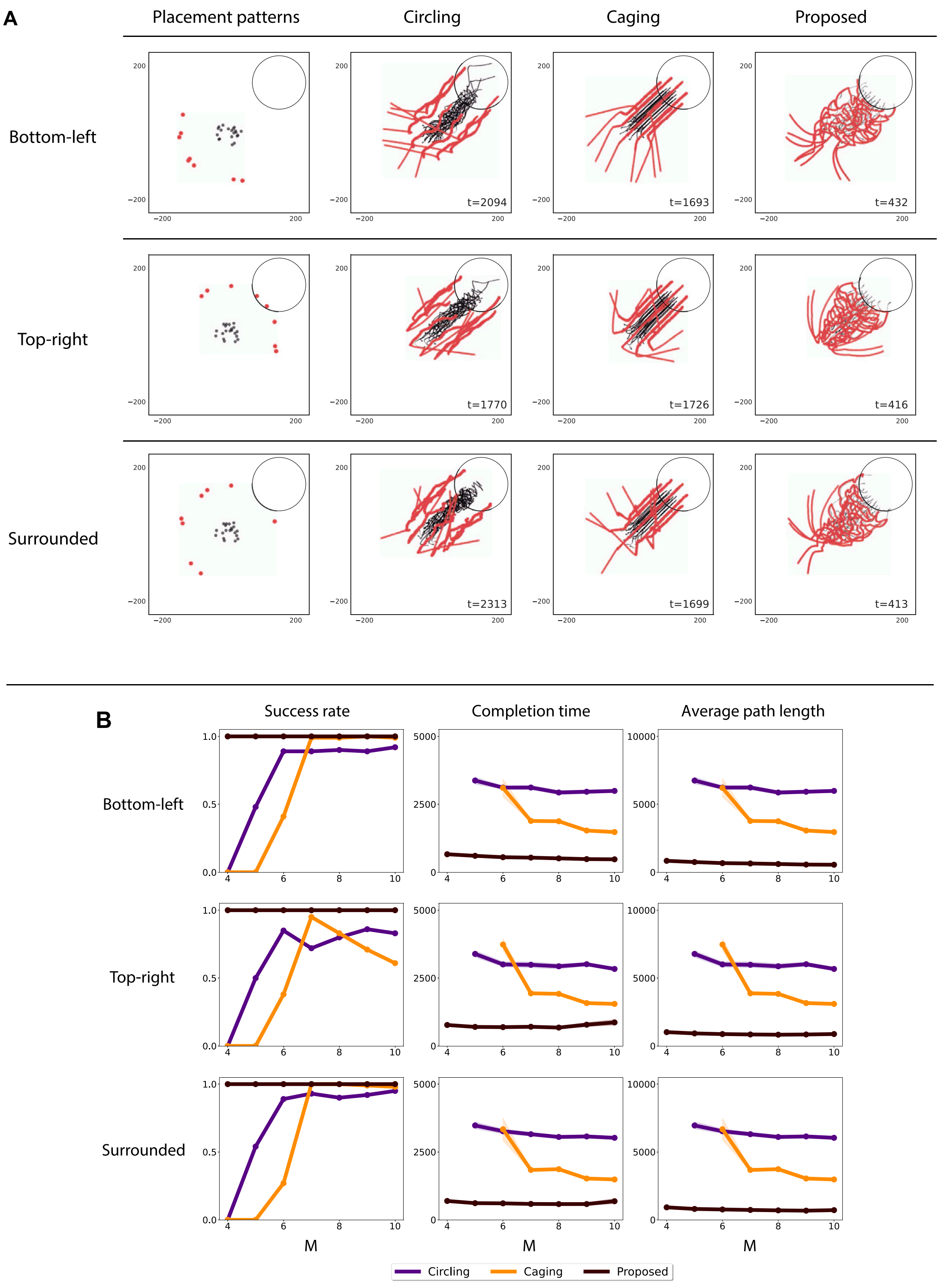


FIGURE 9
Herding a small flock in the baseline sheep model compared with centralized shepherding algorithms. **(A)** Initial configurations and corresponding trajectories. The number of shepherds is set to $M = 8$. A movie can be found at https://youtu.be/AlrRmSfvYcs. **(B)** Performances of the three algorithms. Horizontal axes represent the number of shepherds $M$ from 4 to 10. The coordinates of the $y$-axis are increased in completion time and average path length.

strategy to allocate the shepherds to their target positions. The maximum time step is set to 5000 to ensure the completion of the shepherding.

The trajectories for herding the small and large flocks are shown in Figures 9A, 10A and the performances in Figures 9B, 10B, respectively. Simulation results show that for these two

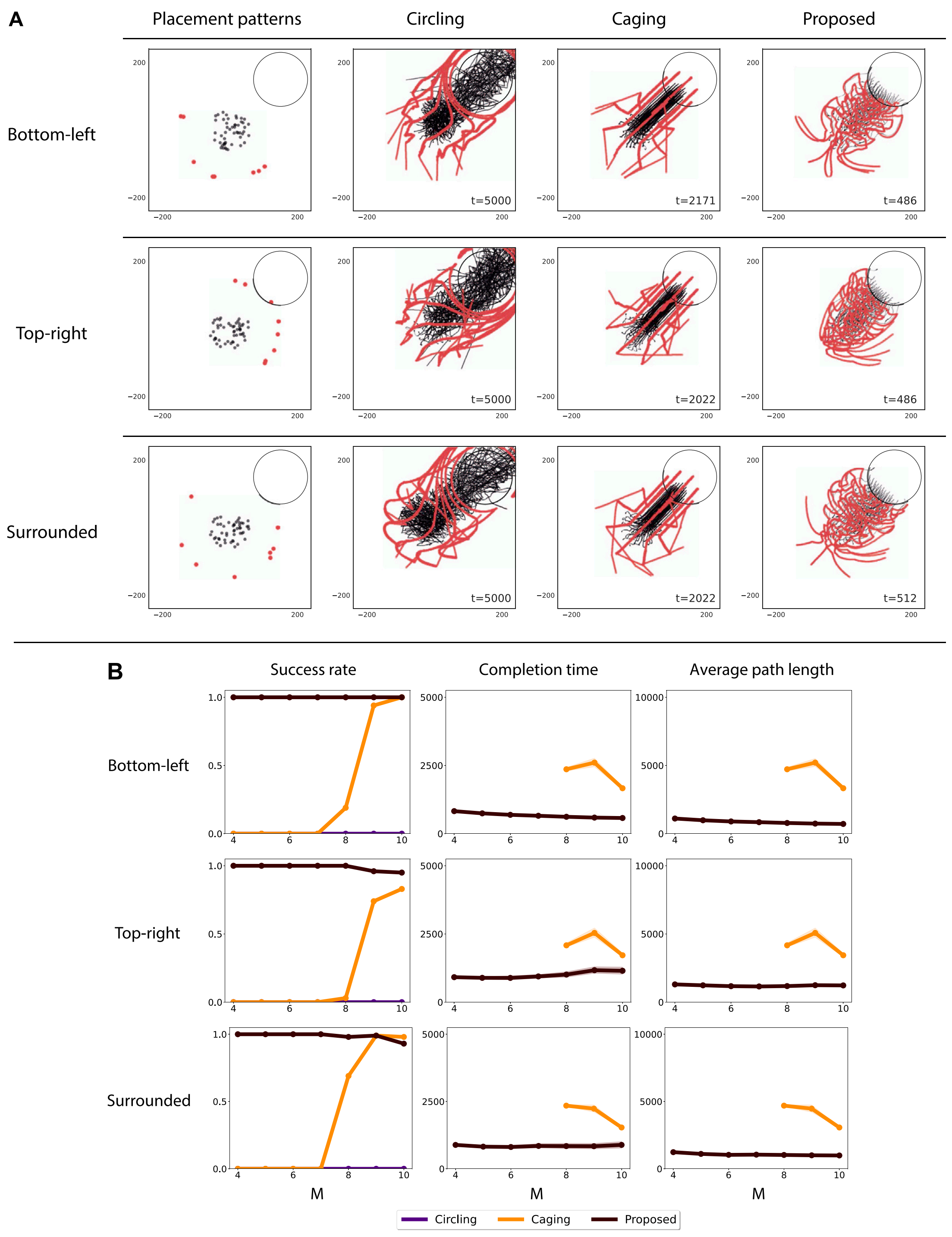
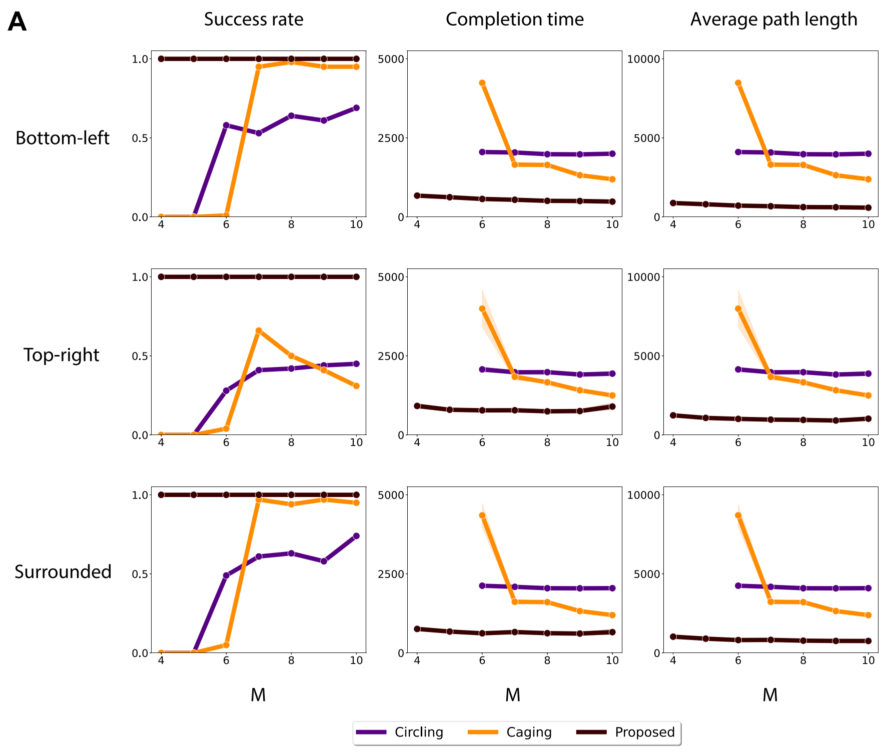


FIGURE 10
Herding a large flock in the baseline sheep model compared with centralized shepherding algorithms. **(A)** Initial configurations and corresponding trajectories. The number of shepherds is set to $M = 8$. The circling algorithm of shepherding ends until the maximum time step. A movie can be found at https://youtu.be/AIrRmSfvYcs. **(B)** Performances of the three algorithms. Horizontal axes represent the number of shepherds $M$ from 4 to 10. The coordinates of the $y$-axis are increased in completion time and average path length.

algorithms, the average completion time increases and the success rate decreases as $M$ increases. On the other hand, when herding a flock with large $N$ and $R_s(0)$, the success rate is not necessarily high. We analyze that the poor performance is due to the large interaction distance from the shepherd to the sheep flock. For the case of two separate flocks, we choose not to show the simulation results because we found through numerical simulations that these two centralized algorithms are not capable of performing the shepherding task. From the simulation results based on the two other sets of the sheep model, in Figures 11A, 12A and Figure 11B, 12B, we observe that the performance



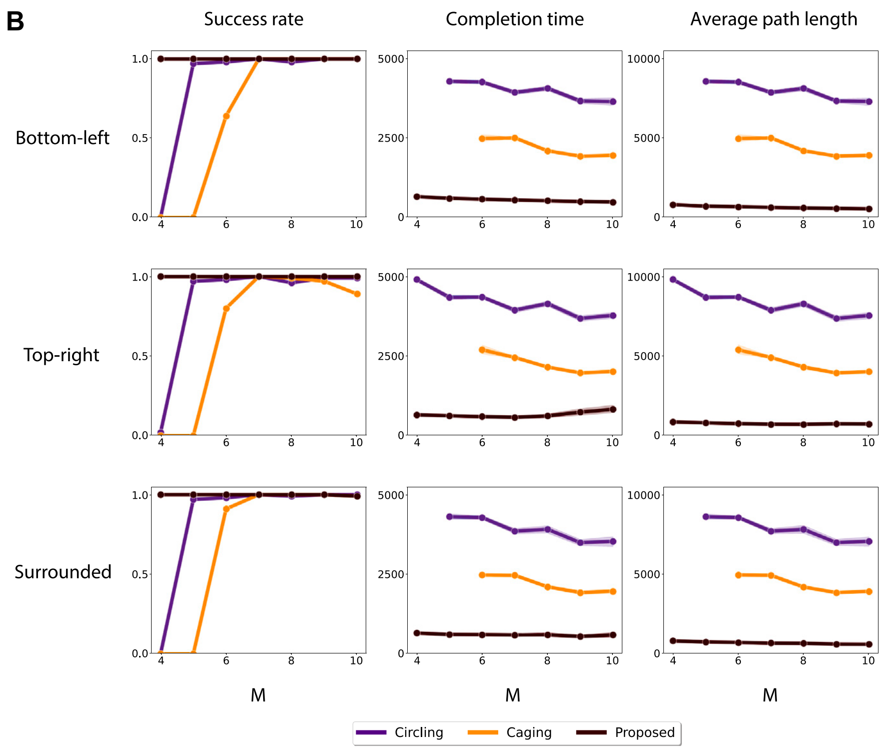


**FIGURE 11**
Herding a small flock in the sensitive and insensitive sheep model separately compared with centralized shepherding algorithms. **(A)** Performances of the three algorithms in the sensitive sheep model. Horizontal axes represent the number of shepherds $M$ from 4 to 10. The coordinates of the $y$-axis are increased in completion time and average path length. **(B)** Performances of the three algorithms in the insensitive sheep model. Horizontal axes represent the number of shepherds $M$ from 4 to 10. The coordinates of the $y$-axis are increased in completion time and average path length.

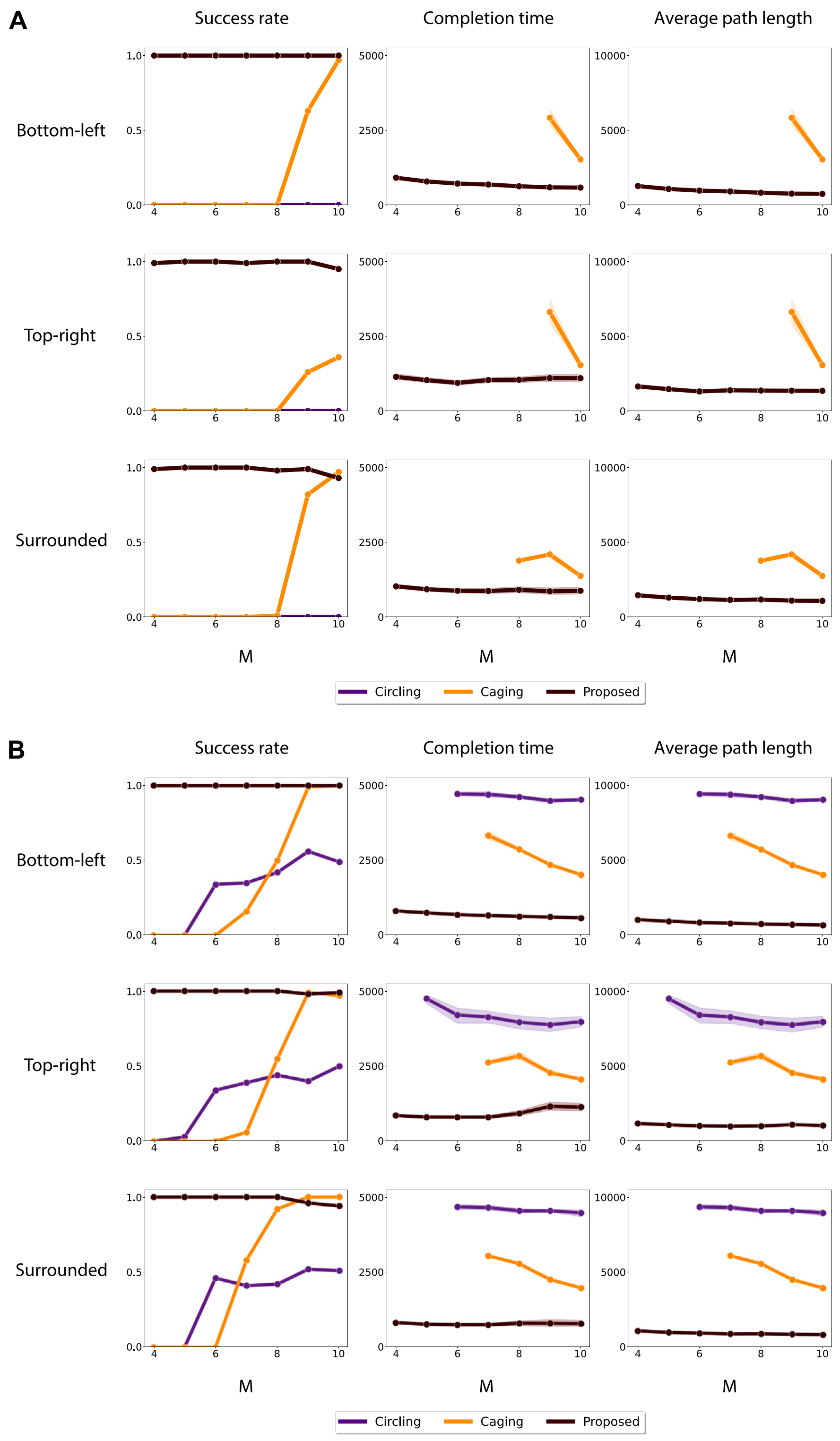


**FIGURE 12**
Herding a large flock in the sensitive and insensitive sheep model separately compared with centralized shepherding algorithms. **(A)** Performances of the three algorithms in the sensitive sheep model. Horizontal axes represent the number of shepherds *M* from 4 to 10. The coordinates of the *y*-axis are increased in completion time and average path length. **(B)** Performances of the three algorithms in the insensitive sheep model. Horizontal axes represent the number of shepherds *M* from 4 to 10. The coordinates of the *y*-axis are increased in completion time and average path length.

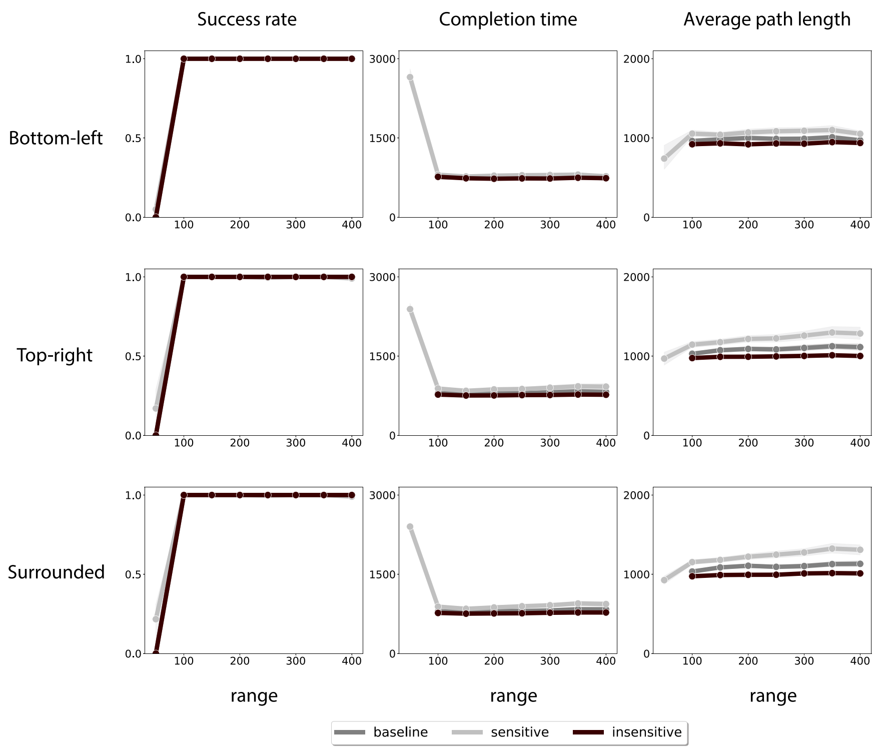


**FIGURE 13**
Performances of the proposed algorithm for shepherding three models of sheep flock in different sensing ranges for shepherds. Horizontal axes represent the sensing range of shepherds $r'$. The sensing range of the sheep is set as the default value $r$ =50.

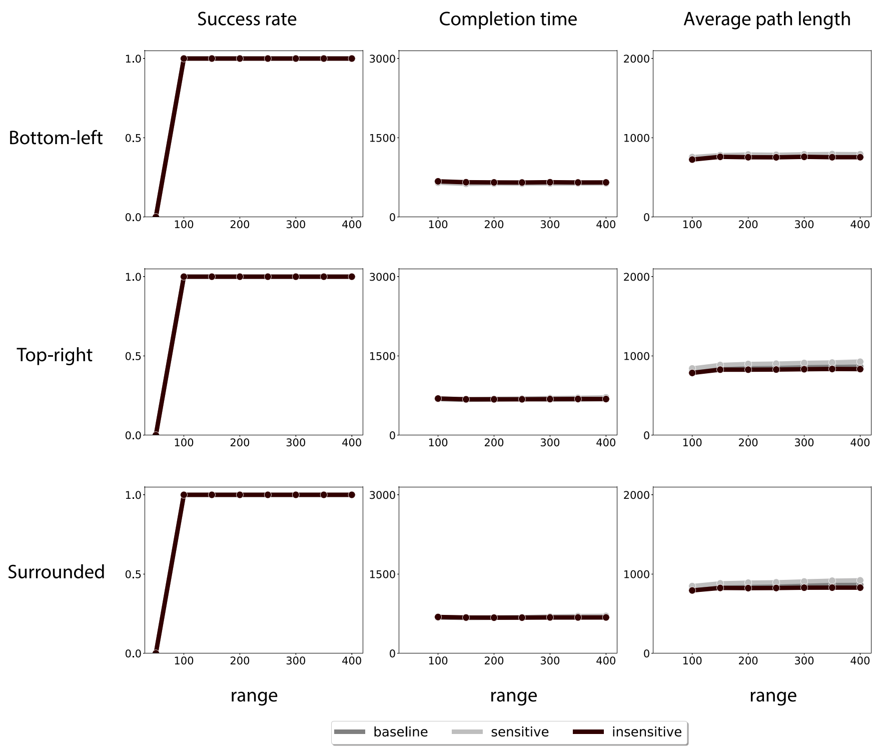


**FIGURE 14**
Performances of the proposed algorithm for shepherding three models of sheep flock in different sensing range for shepherds. Horizontal axes represent the sensing range of shepherds $r'$. The sensing range of the sheep is enlarged as $r$ =100.

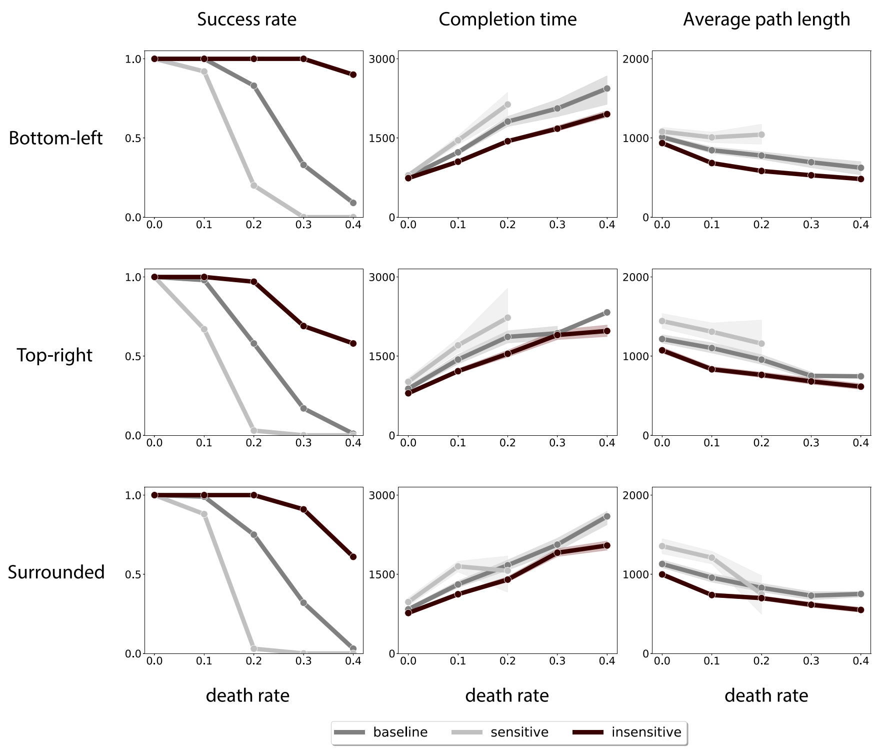


**FIGURE 15**
Performances of the proposed algorithm for shepherding three models of sheep flock by introducing death rate for shepherds. Horizontal axes represent the death rate $P_d$ for shepherds. The recovery rate $P_r$ is set as 0.1.

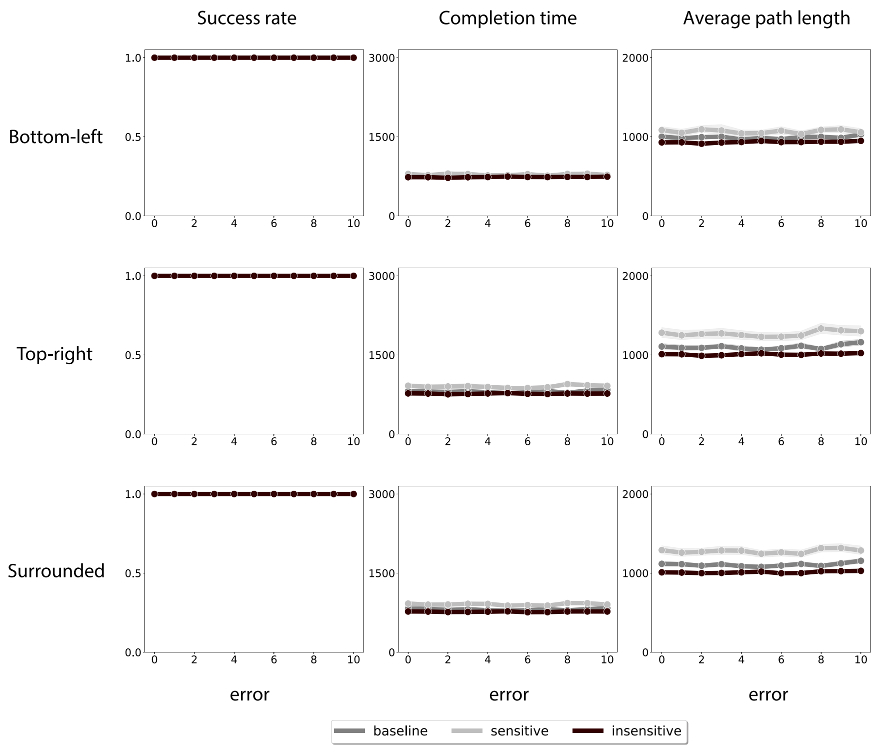


**FIGURE 16**
Performances of the proposed algorithm for shepherding three models of sheep flock in considering sensing error for shepherds. Horizontal axes represent the value $d_5$ of the sensing error for shepherds.

of these two algorithms is greatly influenced by the parameter setting of the sheep model. After examining the simulation data, we find that the centralized multi-agent shepherding algorithms are neither robust nor resilient enough to adapt to changes in the flock shape caused by the movement of sheep agents.

## 6 Robustness and resilience

In order to further demonstrate the effectiveness of the proposed algorithm, in this subsection, we numerically evaluate the robustness and resilience properties. Throughout this subsection, we use 2) the large flock to be the same set of initial configurations as the ones we used in the last subsection. The sheep were modeled as baseline, sensitive, or insensitive flocks. The number of the shepherds is fixed as $M = 5$. Because our main objective in this subsection is to investigate the robustness and resilience properties of the proposed algorithm, we do not perform simulations of existing methods.

We first examine the robustness of the proposed algorithm with respect to the change in the sensing range of the shepherd and the sheep. As for the sensing range of the shepherd, we change $r'$ from its default value 300 and vary within the set $\{50, 100, 150, \ldots, 400\}$. Also, we prepare two scenarios on the sensing range $r$ of the sheep; $r = 50$ and $r = 100$. We present how the success rates, completion times, and average path lengths depend on $r'$ in Figure 13 ($r = 50$) and Figure 14 ($r = 100$). According to the results, different sizes of the sensing ranges $r$ of sheep cause changes in the flock behaviors to influence the shepherding performance. For these two values of $r$, the success rate of shepherding drops when the sensing range $r'$ of the shepherd is short. This observation suggests that, for the proposed algorithm to be effective, we should avoid employing a shepherd having a too short sensing range.

We then examine the resilience of the shepherding algorithm against the errors and failures in the shepherd agents. For this purpose, we consider the following situation. Suppose that a shepherd is prone to failures, and a failure occurs at each time step with probability $P_f$. We suppose that a shepherd in the failure state does not move. We also assume that, when a shepherd is in the failure state, the shepherd can recover from the failure and return to the normal state with probability $P_r$. In our numerical simulation, we set the recovery probability as $P_r = 0.1$ and vary the failure probability within the set $\{0, 0.1, 0.2, 0.3, 0.4\}$. We show the results of the simulation in Figure 15. We observe that the proposed algorithm still maintains a high success rate. Specifically, although the completion time rises with the increase of $P_r$, the traversal distance remains almost unchanged. Overall, this simulation indicates that the proposed algorithm is fault-tolerant due to its decentralized mechanism.

We finally investigate the resilience of the proposed algorithm against sensing errors of the shepherds. In this simulation, we assume that the sensing of the shepherd to the positions of other agents and the goal is subject to an additive noise of the form $d_5\sigma(t)$, where $d_5$ is a positive weight and the random vector $\sigma(t)$ is generated in the same way as the random vector $u_{i5}(t)$ appearing in Eq. 1. In Figure 16, we show how the performance of the proposed algorithm depends on the weight $d_5$. We confirm that the proposed algorithm tolerates relatively large sensing error with $d_5 = 10$ in any of the initial configurations. This observation shows that the proposed algorithm is resilient to sensing errors.

## 7 Conclusion

In this paper, we have studied the shepherding problem withmultiple steering agents unable to communicate with each other. We have first presented a model of sheep agents in the presence of multiple steering agents. We have then proposed a distributed and communication-free algorithm with multiple steering agents to aggregate the sheep agents by location-based self planning. Finally, we have confirmed the effectiveness and resilience of the proposed algorithm via extensive numerical simulations in various situations. Interesting directions for future works include investigating if the proposed communication-free coordination mechanism can be extended to other types of navigation tasks. Another important research direction is to analyze the performance of the proposed shepherding algorithm mathematically. One of the particular interests is in establishing fundamental properties such as stability and convergence.

## Data availability statement

The raw data supporting the conclusions of this article will be made available by the authors, without undue reservation.

## Author contributions

All authors listed have made a substantial, direct, and intellectual contribution to the work and approved it for publication.

## Funding

This work was supported by JSPS KAKENHI Grant Numbers JP21H01352, JP21H01353, and JP22H00514 and Osaka University Institute for Datability Science “Transdisciplinary Research Project”.

## Conflict of interest

The authors declare that the research was conducted in the absence of any commercial or financial relationships that could be construed as a potential conflict of interest.

## Publisher’s note

## Supplementary Material

The Supplementary Material for this article can be found online at: https://www.frontiersin.org/articles/10.3389/fcteg.2023.989232/full#supplementary-material

## References

Bacon, M., and Olgac, N. (2012). Swarm herding using a region holding sliding mode controller. *J. Vib. Control* 18, 1056–1066. doi:10.1177/1077546311411346

Chipade, V. S., and Panagou, D. (2020). "Multi-swarm herding: Protecting against adversarial swarms," in 59th IEEE Conference on Decision and Control (Jeju, South Korea: IEEE), 5374–5379. doi:10.1109/CDC42340.2020.9303837

Chung, S.-J., Paranjape, A. A., Dames, P., Shen, S., and Kumar, V. (2018). A survey on aerial swarm robotics. *IEEE Trans. Robotics* 34, 837–855. doi:10.1109/tro.2018.2857475

Consolini, L., Morbidi, F., Prattichizzo, D., and Tosques, M. (2008). Leader–follower formation control of nonholonomic mobile robots with input constraints. *Automatica* 44, 1343–1349. doi:10.1016/j.automatica.2007.09.019

El-Fiqi, H., Campbell, B., Elsayed, S., Perry, A., Singh, H. K., Hunjet, R., et al. (2020). The limits of reactive shepherding approaches for swarm guidance. *IEEE Access* 8, 214658–214671. doi:10.1109/access.2020.3037325

Gazi, V., and Passino, K. M. (2003). Stability analysis of swarms. *IEEE Trans. Automatic Control* 48, 692–697. doi:10.1109/tac.2003.809765

Hou, Y., Xiao, S., Bi, M., Xue, Y., Pan, W., and Hu, W. (2016). Single LED beacon-based 3-D indoor positioning using off-the-shelf devices. *IEEE Photonics J.* 8, 1–11. doi:10.1109/jphot.2016.2636021

Hu, J., Turgut, A. E., Krajník, T., Lennox, B., and Arvin, F. (2020). Occlusion-based coordination protocol design for autonomous robotic shepherding tasks. *IEEE Trans. Cognitive Dev. Syst.* 14, 126–135. doi:10.1109/tcds.2020.3018549

Lee, W., and Kim, D. (2017). Autonomous shepherding behaviors of multiple target steering robots. *Sensors* 17, 2729. doi:10.3390/s17122729

Lien, J. M., Rodriguez, S., Malric, J. P., and Amato, N. (2005). "Shepherding behaviors with multiple shepherds," in Proceedings of the 2005 IEEE International Conference on Robotics and Automation (Barcelona, Spain: IEEE), 3402–3407. doi:10.1109/ROBOT.2005.1570636

Long, N. K., Sammut, K., Sgarioto, D., Garratt, M., and Abbass, H. A. (2020). A comprehensive review of shepherding as a bio-inspired swarm-robotics guidance approach. *IEEE Trans. Emerg. Top. Comput. Intell.* 4, 523–537. doi:10.1109/tetci.2020.2992778

Mou, F., Li, X., Xie, Q., Zhang, J., Xiong, K., Xu, L., et al. (2020). Active micromotor systems built from passive particles with biomimetic predator–prey interactions. *ACS Nano* 14, 406–414. doi:10.1021/acsnano.9b05996

Parrish, J., Viscido, S., and Grunbaum, D. (2002). Self-organized fish schools: An examination of emergent properties. *Biol. Bull.* 202, 296–305. doi:10.2307/1543482

Pierson, A., and Schwager, M. (2018). Controlling noncooperative herds with robotic herders. *IEEE Trans. Robotics* 34, 517–525. doi:10.1109/tro.2017.2776308

Qu, S., Abouheaf, M., Gueaieb, W., and Spinello, D. (2021). "An adaptive fuzzy reinforcement learning cooperative approach for the autonomous control of flock systems," in 2021 IEEE International Conference on Robotics and Automation (Xi'an, China: IEEE), 8927–8933. doi:10.1109/ICRA48506.2021.9561204

Reynolds, C. W. (1987). "Flocks, herds and schools: A distributed behavioral model," in Proceedings of the 14th Annual Conference on Computer Graphics and Interactive Techniques, 25–34. doi:10.1145/37402.37406

Song, H., Varava, A., Kravchenko, O., Kragic, D., Wang, M. Y., Pokorny, F. T., et al. (2021). Herding by caging: A formation-based motion planning framework for guiding mobile agents. *Aut. Robots* 45, 613–631. doi:10.1007/s10514-021-09975-8

Strömbom, D., Mann, R. P., Wilson, A. M., Hailes, S., Morton, A. J., Sumpter, D. J., et al. (2014). Solving the shepherding problem: Heuristics for herding autonomous, interacting agents. *J. R. Soc. Interface* 11, 20140719. doi:10.1098/rsif.2014.0719

Tsunoda, Y., Sueoka, Y., Sato, Y., and Osuka, K. (2018). Analysis of local-camera-based shepherding navigation. *Adv. Robot.* 32, 1217–1228. doi:10.1080/01691864.2018.1539410

Vemula, A., Muelling, K., and Oh, J. (2018). "Social attention: Modeling attention in human crowds," in 2018 IEEE International Conference on Robotics and Automation (Brisbane, QLD, Australia: IEEE), 4601–4607. doi:10.1109/ICRA.2018.8460504

Zhi, J., and Lien, J. M. (2021). Learning to herd agents amongst obstacles: Training robust shepherding behaviors using deep reinforcement learning. *IEEE Robotics Automation Lett.* 6, 4163–4168. doi:10.1109/lra.2021.3068955