SCIENCE CHINA
Information Sciences

• RESEARCH PAPER •

December 2024, Vol. 67, Iss. 12, 222201:1–222201:19
https://doi.org/10.1007/s11432-023-4040-y

A novel memetic algorithm for distributed shape formation of swarm robots with both acceleration and velocity constraints

Yun QU[1], Bin XIN[1*], Qinqin WANG[2], Ruocheng LI[1] & Zhaofeng DU[1]

[1]*National Key Lab of Autonomous Intelligent Unmanned Systems, Beijing Institute of Technology, Beijing* 100081*, China;*
[2]*Science and Technology Innovation Center, Beijing* 100012*, China*

Received 8 August 2023/Revised 26 October 2023/Accepted 16 January 2024/Published online 11 November 2024

Abstract This article investigates the problem of distributed shape formation (DSF) in swarm robot systems. DSF involves each robot autonomously selecting and moving toward a target point to achieve a desired shape. A comprehensive mathematical model for DSF of swarm robots is developed, capturing the relationships among states, behaviors, and kinematics constraints. A novel memetic algorithm (MA) is proposed to generate and evolve behavior strategies that enable real-time decision-making for each robot. The proposed MA includes specifically designed behaviors to address subproblems within DSF, such as target selection conflict, collision, and deadlock. These behaviors, along with propositions about the states of robots, are utilized to construct strategies by a tree-based encoding scheme. An evaluation mechanism based on multi-agent simulation is devised to evaluate the performance of strategies. Additionally, a repair mechanism is introduced to eliminate redundant or unreachable subtrees in a strategy. To further improve the performance of generated strategies, tree-based local search operators are employed to exploit the neighborhood of the best strategy found yet during the iteration. Experimental results and the Wilcoxon rank-sum test show that the strategies generated by the proposed MA outperform state-of-the-art algorithms, significantly reducing the completion time of DSF.

Keywords distributed shape formation, swarm, memetic algorithm, strategy, decision-making

1 Introduction

Organisms in nature, such as fish [1] moving in an orderly manner in the ocean, birds [2] soaring together in the sky, and ants [3] collaborating in the division of labor to build nests, often exhibit coordinated, ordered, and awe-inspired collective behaviors. These organisms have strong emerging stigmergy [4], cooperation [5–7], stability and adaptability [8], allowing them to accomplish complex tasks without centralized planning. Researchers have endeavored to apply emergent stigmergy observed in natural organisms to industry. Exploring the behaviors of organisms can provide valuable insights to guide swarm robots in making independent decisions based on information interaction within complex environments.

Distributed shape formation (DSF) [9–13] is one of the fundamental problems in swarm robots. DSF refers that each robot spontaneously gathers and associates with its neighbors to form a specific shape defined by a set of target positions, as shown in Figure 1. DSF takes a crucial role in complex systems, such as modular robots [14], information relay [15], and target surrounding [16–18].

Distributed algorithms [19–23] used in DSF are known for their strong fault tolerance and scalability. Most existing approaches focus on controlling the distance [19–23], orientation [20], and/or position [24–27] of robots to form a desired shape. These methods [19–23] often rely on distance measurements for robot control, which allows operation in GPS-denied environments and enables robots to stabilize to a dynamically moving shape. The methods proposed in [20–22] require a fixed set of neighbors, a complete communication graph among robots, and the assignment of leader/non-leader roles for robots, respectively. Position-based methods [24–27] are designed to prevent collisions among robots and guide each robot to move toward its target position within a global coordinate system.

* Corresponding author (email: brucebin@bit.edu.cn)

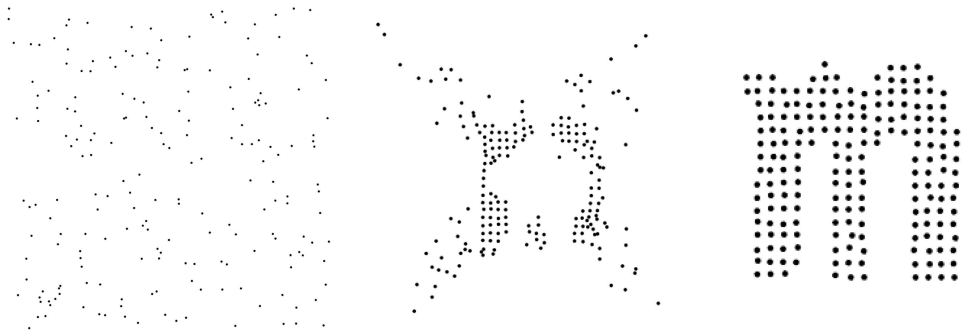

Figure 1 Still images from a 200 robot shape formation simulation. (a) 0 s; (b) 210 s; (c) 410 s.

Distributed algorithms in DSF utilize information from each robot and its neighbors to achieve the desired shape. However, these algorithms often rely heavily on the designer's experience and subjective awareness, and may not effectively discover associations between robot behaviors and states. Evolutionary algorithms, inspired by the biological evolution process, have been effectively employed to generate and evolve efficient heuristic rules for various problems, such as the table transport problem [28] and dynamic flexible job shop scheduling (DFJSS) [29]. These generated heuristics with their flexible structural representation can guide each robot in making real-time decisions. The memetic algorithm (MA) [30,31] is a typical hybrid evolutionary algorithm that combines population-based optimization with local search (LS) heuristics. It serves as a robust hybrid optimization framework that has been successfully used to explore the neighborhood of the best strategy found during iterations.

As far as we know, evolutionary algorithms have not been employed in the DSF problem. Considering the similarity between the DSF problem and DFJSS in their shared requirement of quickly adapting to dynamic environments, a novel MA is proposed to find efficient strategies for the DSF problem. Our contributions and innovations are presented as follows.

(a) Different from the existing distributed algorithms [13, 26], we propose a DSF-specific memetic algorithm (DSFMA) to autonomously search the space of heuristic rules, which is named as behavior strategies, instead of directly searching the solution space about the position-robot assignment and motion planning. The DSFMA incorporates some DSF-specific behaviors to generate and evolve strategies. These strategies enable each robot, while satisfying both acceleration and velocity constraints, to make real-time decisions. Computational experiment results show that the DSF completion time by the optimized strategies is significantly reduced as compared to that of the state-of-the-art heuristics [13, 26] in most test instances.

(b) An efficient DSFMA algorithm is designed:

- We design a terminal set that includes DSF-specific behaviors. Specifically, a DSF-specific target conflict resolver with orientation preference is proposed to solve target conflict among robots. The acceleration velocity obstacles (AVO) algorithm [32] is used to calculate the feasible speed set of each robot. Additionally, swap, aggregation, dispersion, greed, and combinational behaviors are designed to explore efficient strategies.
- A DSF-specific evaluation is proposed to evaluate the execution results of overrun and/or collision in DSF. A repair mechanism is proposed to eliminate redundancy in strategies. This mechanism not only reduces the frequency of appearance of inferior subtrees within an individual but also prevents invalid crossover operations where multiple duplicate subtrees are exchanged.
- To maintain a tradeoff between exploration and exploitation, a tree-based local search operator is designed to exploit the neighborhood of the best behavior strategy found yet during the iteration.

The rest of this paper is organized as follows. A mathematical formulation of DSF is described in Section 2. In Section 3, the proposed DSFMA algorithm is introduced. Experimental results and performance comparison are presented in Section 4. Finally, a conclusion is presented in Section 5.

2 Problem formulation

This section begins with a brief introduction to the DSF problem, followed by an explanation of the notations used in this paper, the robot model, and the decision-making model of robots. It then proceeds to describe the objective of decision-making on DSF.

2.1 Problem definition

The DSF problem is to allocate and control M robots to M different target positions in a desired shape autonomously. All target positions are known in advance. Robots are randomly distributed in the workspace. Each robot can measure its position in a shared global coordinate system. Through limited local communication, each robot selects a target position and moves toward it without collision with its neighbors. It is important to note that each robot will be assigned to a target point and vice versa.

2.2 Notations

For the sake of formulating the decision-making, notations are introduced as follows. Let $B_i(t) = [b_{ij}(t)]_{M\times M}$ denote the position-robot assignment (PRA) scheme, where $b_{ij}(t)$ is the PRA decision variable regrading the ith robot and the jth target point at time t. Specifically, $b_{ij}(t) = 1$ indicates that the ith robot will access the jth position, and $b_{ij}(t) = 0$ otherwise. $P(t) = \{\boldsymbol{p}_1(t), \boldsymbol{p}_2(t), \ldots, \boldsymbol{p}_M(t)\}$ represents absolute positions of all robots in the Cartesian coordinate system. $\forall \boldsymbol{p}_i(t) \in \widetilde{P}$, $\widetilde{P}$ is the set of all positions of robots in their workspace. The speed of the ith robot at time t is defined as $\boldsymbol{v}_i(t)$, and $\theta_i(t)$ represents the angle of ith robot. $\widetilde{V}$ is the set of all speeds of robots in their workspace. $G = \{\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_M\}$ represents absolute positions of all target points. $\boldsymbol{o}_i(t)$ represents the target position to be accessed by the ith robot at time t. The set $O_i(t) = \{\boldsymbol{g}_1, \boldsymbol{g}_2, \ldots, \boldsymbol{g}_k\}$ consists of the target positions that the ith robot has not accessed, where $k \leqslant M$. The set $C_i(t)$ comprises the positions of all neighbors of the ith robot at time t. It is defined such that $\forall i \neq j, \boldsymbol{p}_j(t) \in C_i(t)$ if and only if $\|\boldsymbol{p}_i(t) - \boldsymbol{p}_j(t)\| \leqslant R$, where R represents the communication radius of each robot. $\widetilde{C}_i(t)$ is the set composed of the not accessed target positions of the nearest neighbor of the ith robot at time t. $\psi_i(t) = \{(\boldsymbol{p}_j, \boldsymbol{o}_j(k)) \,|\forall j \in C_i(t)\}$ is the set of dualistic pairs consisting of the positions of neighbors of the ith robot and the target positions to be accessed by those neighbors at time t. Generally, $\psi_i(t)$ is time varying.

2.3 Robot model

The following robot model is given based on [13]. Each robot is treated as a circle with a radius r, capable of omnidirectional acceleration and can measure its own position in the workspace. Each robot has its label and makes decisions independently. To ensure a smooth transition to a new speed, an acceleration constraint $\|\boldsymbol{a}_i(t)\| < a^{\max}$ is enforced. The acceleration constraint allows each robot to gradually adapt to the new speed $\boldsymbol{v}_i'(t)$ over time. Additionally, each robot obeys the speed constraint $\|\boldsymbol{v}_i(t)\| < v^{\max}$, where $v^{\max}$ is the maximum speed of each robot.

2.4 Decision making model of robots

Here, we present the decision-making process for each robot, which is guided by a strategy that selects its target position to be accessed and calculates its speed for the next step. To provide a clear presentation of this process, we define a state space, decision vector, and strategy as follows.

Definition 1. A state vector $\widetilde{\Lambda}_i(t) = [\boldsymbol{p}_i(t), \boldsymbol{o}_i(t), \widetilde{C}_i(t), O_i(t), \psi_i(t)]$ includes the position and target position of the ith robot, the set of the not accessed target positions of the ith robot and its nearest neighbor, as well as the set of dualistic pairs consisting of the positions of neighbors of the ith robot and target positions to be accessed by those neighbors. $\Lambda_i = \{\widetilde{\Lambda}_i(t) | \widetilde{\Lambda}_i(t) \in \widetilde{P} \times G \times G \times G \times 2^{\widetilde{P}\setminus(p_i)\times G}\}$ is the state space of the ith robot.

Definition 2. A decision vector $\widetilde{\Xi}_i(t) = [\boldsymbol{o}_i'(t), \boldsymbol{v}_i'(t)]$ comprises the target position to be accessed and the new speed of the ith robot. $\Xi_i = \{\widetilde{\Xi}_i(t) | \widetilde{\Xi}_i(t) \in G \times \widetilde{V}\}$ is the decision space of the ith robot.

Definition 3. The strategy $f^i(\cdot)$ for the ith robot is a mapping from the state space Λ_i to the decision space Ξ_i, i.e., $f^i : \Lambda_i \to \Xi_i$. Since all robots adhere to the same strategy, the strategy followed by each robot is called $f(\cdot)$.

To sum up, each robot is guided by a strategy $f(\cdot)$ to calculate its target position and new speed for the next time step, i.e., $[\boldsymbol{o}_i'(t), \boldsymbol{v}_i'(t)] = f(\boldsymbol{p}_i(t), \boldsymbol{o}_i(t), \widetilde{C}_i(t), O_i(t), \psi_i(t))$. More specifically, the new velocity $\boldsymbol{v}_i'(t)$ satisfied with speed and acceleration constraints is calculated as follows:

$$\boldsymbol{v}_i'(t) = \underset{\boldsymbol{v}(t)\in \boldsymbol{V}_i^f(t)}{\arg\min} \left\| \boldsymbol{v}(t) - \boldsymbol{v}_i^{\text{pref}}(t) \right\|, \tag{1}$$

where $\boldsymbol{V}_i^f(t)$ is the feasible speed set of the ith robot. The concrete calculation of the set $\boldsymbol{V}_i^f(t)$ will be introduced in Section 3. The preferred speed $\boldsymbol{v}_i^{\text{pref}}(t)$ of the ith robot is defined as

$$\left\| \boldsymbol{v}_i^{\text{pref}}(t) \right\| = \min\left(\left\| \boldsymbol{o}_i'(t) - \boldsymbol{p}_i(t) \right\|, v^{\max} \right). \tag{2}$$

The direction of preferred speed $\boldsymbol{v}_i^{\text{pref}}(t)$ is determined by the orientation of the ith robot to its target point.

It is worth noting that each robot moves within the feasible region. $\mathcal{R}_i(t)$ is the feasible region of the ith robot. In $\mathcal{R}_i(t)$, the ith robot can avoid collision with its neighbors and satisfy the following equation:

$$\|\boldsymbol{p}_i(t) - \boldsymbol{p}_j(t)\| > 2 \times r, \ \forall j = 1, 2, \ldots, M, i \neq j. \tag{3}$$

2.5 Objective function

$\widetilde{t}$ is the time that each robot is guided by a strategy to complete the DSF task. The completion time of the DSF is defined as

$$\widetilde{t}(f(\cdot)) = \max_{i=1,2,\ldots,M} (t_i(f(\cdot))), \tag{4}$$

where $t_i(f(\cdot))$ presents the time when the ith robot arrives at its target position. It should be noted that the target position of each robot is time-varying. The objective of the decision-making is to find the evolved strategy f^* that minimizes the completion time of the DSF, i.e., $f^* = \arg\min_f \{\widetilde{t}(f)\}$.

3 Problem-solving approach for DSF

To discover efficient strategies for DSF, we proposed the DSFMA that generates and evolves strategies. The algorithm comprises several stages. Firstly, the encoding of strategies is presented, defining how strategies are represented. Next, the initialization of strategies and shapes is illustrated. Subsequently, the execution stage is described, where each robot is guided by a strategy to form a desired shape, and the performance data of the strategy are recorded. Finally, the evolution process of DSFMA is outlined, including the designed operators for selection, crossover, mutation, and local search. Additionally, a repair mechanism is introduced to eliminate redundant or unreachable subtrees in a strategy, and a DSF-specific evaluation mechanism is utilized to assess the performance of strategies.

3.1 Encoding

A solution for DSF is encoded as a tree structure, that is, the strategy. Each robot uses the strategy to select its target position according to its own and neighbor states. Though the permutation encoding is a simple representation, the strategy is represented using a tree structure encoding in this paper. It is primarily driven by the ability of tree structure encoding to establish logical relationships between behaviors and states. Furthermore, since there is no fixed length that can effectively represent the logical relationships among robots and states, strategies require a variable-length encoding. Using permutation encoding in this context would result in a considerable number of invalid elements.

A strategy includes propositions and behaviors. Propositions represent judgment conditions of states of robots. Each proposition consists of correlation parameters and states of robots. An example of a behavior strategy is depicted in Figure 2, where ellipses represent propositions. All propositions are placed in the function set Ψ. For clarity, Table 1 provides an illustration of the propositions and notations in the function set Ψ. Squares represent behaviors, and all behaviors are placed in the terminal set $\Upsilon = \{\text{Beh}_1, \text{Beh}_2, \ldots, \text{Beh}_w\}$, where w is the number of behaviors. The specific design of each behavior is described in the terminal set. The preorder traversal is utilized to decode a strategy, converting it into sequences that each robot can execute. Referring to the strategy depicted in Figure 2, the converted sequences are illustrated in Figure 3. Four sequences are presented, with each sequence beginning with a proposition and ending with a behavior. Each robot sequentially judges the relationship between propositions and its own states, progressing from left to right within the same sequence and top to bottom across different sequences. More specifically, if the propositions $S_i^{\text{acc}} < D_m$ and $S_i^{\text{agg}} < D_f$ are true, the ith robot uses the behavior Beh_1 to select its target position. In a word, each robot selects a concrete behavior based on propositions and the judgment identifier (True/False) following the propositions.

After illustrating the encoding and decoding of a strategy, the behaviors and propositions including the terminal set and function set, are respectively described as follows.

Table 1 Propositions and notation description in the function set Ψ and terminal set

Proposition	Notation description
$S_i^{\text{rec}} = \{g_1, g_2, \ldots, g_m\}$	The set of target positions composed of a desired shape, $m \leqslant M$
S_i^{acc}	The ratio of the number of S_i^{rec} to M
S_i^{nei}	The ratio of the number of neighbors of the ith robot to M
S_i^{cen}	The position of center of neighbors of the ith robot
$L_{\max}$	The maximum diagonal length of the workspace
S_i^{agg}	The ratio of the distance between the ith robot and S_i^{cen} to $L_{\max}$
S_i^{ang}	The orientation angle between the ith robot and $\boldsymbol{o}_i(t)$
$\psi_{ik}(t)$	The orientation angle of the vector from position $p_i(t)$ to target position g_k
$S_i^{\text{acc}} < D_s$	The proposition represents that the ith robot has accessed target positions, where $D_s \in (0,1)$
$S_i^{\text{nei}} < D_m$	The proposition is used to judge whether the ith robot arrives at its target position, where $D_m \in (0,1)$
$S_i^{\text{agg}} < D_f$	The proposition represents the congestion degree of the ith robot, where the parameter $D_f \in (0,1)$
$\left\|S_i^{\text{ang}} - \theta_i\right\|/(2\pi) < D_g$	The proposition represents the deviation of the motion direction of the ith robot, where $D_g \in (0,1)$
S^v	A threshold parameter that evaluates the motion rate of a robot, $S^v \in (0,1)$

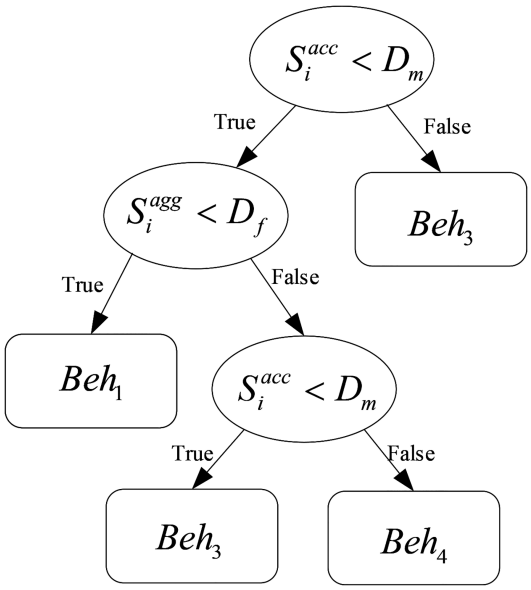

Figure 2 Example of a strategy.

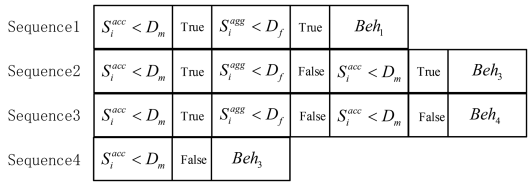

Figure 3 Sequences obtained after decoding the strategy in Figure 2.

3.1.1 *Terminal set*

The terminal set consists of a range of behaviors carefully designed to address various challenges encountered by robots during the DSF task. These behaviors play a crucial role in guiding each robot to independently make decisions and improve the overall strategies. They encompass actions such as collision avoidance, target conflict resolver, navigation distance reduction through target position exchange, congestion adjustment through aggregation and dispersion, and proximity-based target selection. Each behavior serves a specific purpose in enhancing swarm coordination and optimizing the performance of the swarm.

Each robot is guided by the designed behavior to select its target position. Considering the kinematic characteristics of a robot and the difficulty of collision avoidance in the swarm, new velocities of robots are calculated by the reciprocal collision avoidance method [32, 33] after robots select their target positions by behaviors. The following sections provide detailed descriptions of all the designed behaviors.

(a) The reciprocal collision avoidance with acceleration-velocity obstacles (Beh_1). Each robot utilizes the AVO [32] to avoid colliding with its neighbors. The AVO builds upon the velocity obstacles (VO) [33], which has been successfully employed for collision avoidance with moving obstacles.

Typically, VO algorithms estimate the future position of an obstacle based on its observed velocity and compute avoidance velocities to navigate around it. The AVO algorithm extends this concept, where the neighbors are treated as obstacles. Each robot employs proportional control to accelerate toward a new velocity, ensuring collision avoidance with its neighbors acting as obstacles. As shown in Figure 4(a), the jth robot is a moving

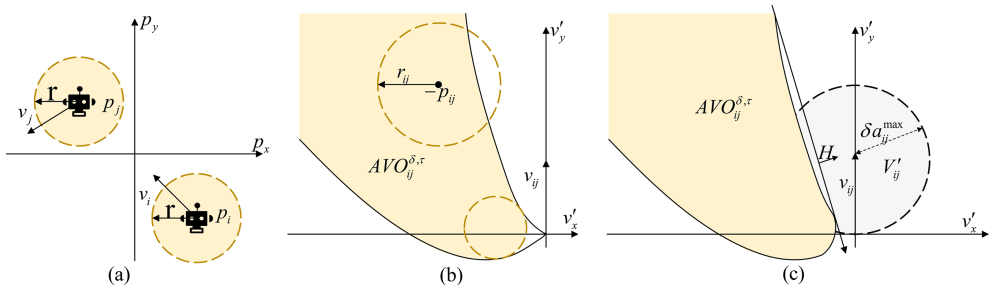

Figure 4 (Color online) (a) The ith and jth robots in the workspace; (b) the acceleration-velocity obstacle $\mathrm{AVO}_{ij}^{\delta,\tau}$; (c) the set V'_{ij} of safe new relative velocity v'_{ij}.

obstacle relative to the ith robot. The relative position $p_{ij}(t) = p_i(t) - p_j(t)$ is shown in Figure 4(b). $r_{ij} = 2 \times r$ is the combined radius of the ith and jth robots. The new position $p'_i(t)$ of the ith robot at next time is calculated as the reference [32] and given by

$$p'_i(t) = p_i(t) + t \times v'_i(t) + \delta\left(\mathrm{e}^{-t/\delta} - 1\right)\left(v'_i(t) - v_i(t)\right), \tag{5}$$

where δ is a control parameter. δ is calculated as follows:

$$\delta = 2v^{\max}/a^{\max}. \tag{6}$$

The two robots will collide at time t, if $\|p_{ij}(t)\| < r_{ij}$. $\|p_{ij}(t)\| < r_{ij}$ can rewrite as follows:

$$\left\| v'_{ij} - \frac{\delta(\mathrm{e}^{-\frac{t}{\delta}} - 1)v_{ij} - p_{ij}}{t + \delta(\mathrm{e}^{-\frac{t}{\delta}} - 1)} \right\| < \frac{r_{ij}}{t + \delta(\mathrm{e}^{-\frac{t}{\delta}} - 1)}. \tag{7}$$

So, the acceleration-velocity obstacle $\mathrm{AVO}_{ij}^{\delta,\tau}$ is a union of discs:

$$\mathrm{AVO}_{ij}^{\delta,\tau} = \bigcup_{0<t\leqslant\tau} \Phi\left(\frac{\delta(\mathrm{e}^{-\frac{t}{\delta}} - 1)v_{ij} - p_{ij}}{t + \delta(\mathrm{e}^{-\frac{t}{\delta}} - 1)}, \frac{r_{ij}}{t + \delta(\mathrm{e}^{-\frac{t}{\delta}} - 1)}\right), \tag{8}$$

where Φ is an open disc of radius r^* centered at p_{r^*}. Φ is presented as follows:

$$\Phi(p_{r^*}, r^*) = \{q \,|\, \|q - p_{r^*}\| < r^*\}. \tag{9}$$

Because each robot considers not only the velocity constraints but also the acceleration constraints, the set of safe new relative velocities V'_{ij} is calculated as described in [32], and given by

$$V'_{ij} = \Phi(v_{ij}, \delta a^{\max}) \setminus \mathrm{AVO}_{ij}^{\delta,\tau}. \tag{10}$$

To find the optimal reciprocal collision avoidance $\boldsymbol{V}_i^f$ for the ith robot induced by the jth robot, a convex subset is obtained by intersecting V'_{ij} with a halfplane H [29]:

$$\hat{V}{}'_{ij} = V'_{ij} \cap H, \tag{11}$$

where H is chosen by the maximal $|\hat{V}{}'_{ij}|$ among all possible halfplanes.

The optimal reciprocal collision avoidance $\boldsymbol{V}_i^f$ of the ith robot is given as follows:

$$\boldsymbol{V}_i^f = \alpha_{ij}\left(\hat{V}{}'_{ij} \oplus \{-v_{ij}\}\right) \oplus \{v_i\}. \tag{12}$$

AVO algorithm is used to calculate the set $\boldsymbol{V}_i^f$. The ith robot uses the new velocity $\boldsymbol{v}'_i(t)$ selecting from the set $\boldsymbol{V}_i^f$ to avoid collisions with its neighbors. The new velocity $\boldsymbol{v}'_i(t)$ is calculated as shown in (2). The more calculation details of the set $\boldsymbol{V}_i^f$ can be found in [32].

(b) The target conflict resolver with orientation preference (Beh_2). The target conflict resolver with orientation preference is used to solve target conflict among robots. The resolver includes the following two rules:

- Rule1. $\forall p_j(t) \in C_i(t)$, if $o_i(t) = o_j(t)$, $\|p_i(t) - o_i(t)\| > \|p_j(t) - o_j(t)\|$, and $\|v(t)\|/v^{\max} \geqslant S^v$, then $o_i(t) = g_{i^*}$, where $i^* = \arg\min_{k\in M} \|\theta_i(t) - \psi_{ik}(t)\|$ and $g_{i^*} \notin S_i^{\mathrm{rec}}(t)$.
- Rule2. $\forall p_j(t) \in C_i(t)$, if $o_i(t) = o_j(t)$, $\|p_i(t) - o_i(t)\| > \|p_j(t) - o_j(t)\|$, and $\|v(t)\|/v^{\max} < S^v$, then $o_i(t) = \arg\min_{g_j\in O_i,(t)g_j\neq o_j(t),g_j\neq S_i^{\mathrm{rec}}(t)} \|p_i(t) - g_j\|$.

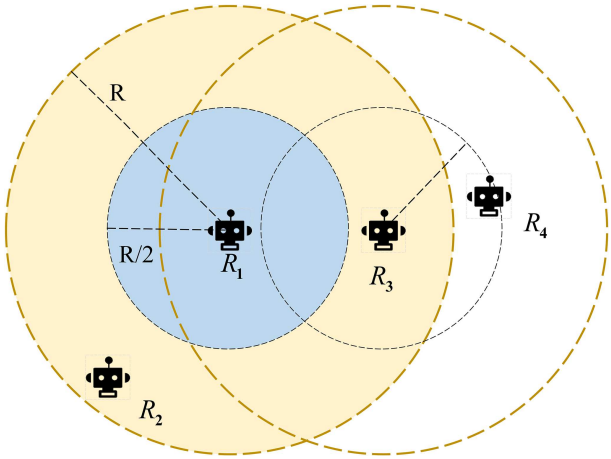

Figure 5 (Color online) Distribution of robots.

$S_i^{\mathrm{rec}}(t)$ is used to record target positions to be accessed by the ith robot and its neighbors. This prevents the ith robot from repeatedly accessing target positions already stored in $S_i^{\mathrm{rec}}(t)$. More specifically, if the robots are going to access the target position g_m at time t, g_m is added to $S_i^{\mathrm{rec}}(t)$. If target positions have not been accessed by any robot and recorded in $S_i^{\mathrm{rec}}(t)$, the target positions will lost, resulting in those target positions becoming inaccessible to any robot. Therefore, it is crucial to avoid losing target positions during the behavior design process.

Remark 1. Each robot uses the resolver to judge whether to adjust its target position based on its own states and that of its neighbors. If the ith robot accesses the same target position with its neighbors ($o_i(t) = o_j(t)$) but is farther away from its target position compared to its neighbors ($\|p_i(t) - o_i(t)\| > \|p_j(t) - o_j(t)\|$), it will select a new target position based on its speed. Since a robot moving fast cannot stop in time and move toward the opposite direction, the robot selects its target position according to its moving speed. When the robot is moving fast ($\|v(t)\|/v^{\max} \geqslant S^v$), the target position closest to its direction of motion is preferred. When the robot is moving slowly ($\|v(t)\|/v^{\max} < S^v$), the target position closest to its own position is selected.

(c) Swap (Beh_3). The swap behavior is designed to eliminate cross paths between robots, thus reducing the navigation distance of robots. It should be noted that two robots can perform the swap behavior without losing target positions, if they are each other's nearest neighbors.

Lemma 1. All robots have the same communication radius. The nearest neighbor robot to the ith robot is located within a communication radius of $R/2$ from the ith robot. Furthermore, the neighbors of the nearest neighbor robot, situated within a communication radius of $R/2$, must also be neighbors of the ith robot.

Proof. $C_i(t)$ is the set composed of positions of all neighbors of the ith robot at time t. $\forall p_j(t) \in C_i(t)$, $p_{i^*}(t) = \min_{p_j(t)} \|p_i(t) - p_j(t)\|$. $\forall p_k(t) \in C_i(t)$, if $\|p_{i^*}(t) - p_k(t)\| \leqslant R/2$ and $\|p_i(t) - p_{i^*}(t)\| \leqslant R/2$, the Euclidean distance between the ith and kth robots is defined as

$$\|p_i(t) - p_k(t)\| \leqslant \|p_i(t) - p_i^*(t)\| + \|p_i^*(t) - p_k(t)\| \leqslant R/2 + R/2 = R. \tag{13}$$

To sum up, $p_k(t) \in C_i(t)$, that is, $p_k(t)$ is the position of the neighbor robot of the ith robot. Based on Lemma 1, the ith robot can determine whether there is a robot, which is the nearest neighbor robot to each other, within the $R/2$ communication range. If $\|p_i(t) - p_{i^*}(t)\| < \|p_{i^*}(t) - p_k(t)\| \leqslant R/2$, the ith and i^*th robots are each other's nearest neighbors.

An example is given to illustrate that a robot cannot find the robot, which is the nearest neighbor robot to each other, outside its $R/2$ communication range. As shown in Figure 5, the yellow area is the cross section formed by a robot within the communication radius R, and the blue area is that of within the communication radius $R/2$. In Figure 5, the robots R_2 and R_3 are neighbors of robot R_1, and the nearest neighbor robot of robot R_1 is R_3. But the nearest neighbor robot of robot R_3 is R_4, robot R_1 cannot determine whether robot R_3 is each other's nearest neighbor.

When a robot finds a neighbor robot that is the nearest neighbor robot to each other and satisfies the following two cases, the robot will execute the swap behavior.

- In the first case, if the total navigation length of two robots decreases after swapping their target points, that is, $\|p_i(t) - o_i(t)\| + \|p_{i^*}(t) - o_{i^*}(t)\| > \|p_i(t) - o_{i^*}(t)\| + \|p_{i^*}(t) - o_i(t)\|$, the two robots perform the swap behavior.
- In the second case, if the total navigation length of two robots remains unchanged after swapping their target points, that is, $\|p_i(t) - o_i(t)\| + \|p_{i^*}(t) - o_{i^*}(t)\| = \|p_i(t) - o_{i^*}(t)\| + \|p_{i^*}(t) - o_i(t)\|$, assigning the target point farther away from the two robots to the robot closer to the target point will help alleviate the situation of robot

deceleration caused by the mutual influence between robots. So if $\max\{\|p_i(t)-o_i(t)\|, \|p_{i^*}(t)-o_{i^*}(t)\|\} > \max\{\|p_i(t)-o_{i^*}(t)\|, \|p_{i^*}(t)-o_i(t)\|\}$, the swap behavior will be executed.

(d) Aggregation (Beh_4). The initial distribution of robot positions is relatively dispersed. The aggregation behavior is designed to rapidly move robots toward a specific region forming the desired shape. Each robot calculates the position of its gathering center $\boldsymbol{S}_i^{\mathrm{cen}}(t)$ at time t. $\boldsymbol{S}_i^{\mathrm{cen}}(t)$ is given as follows:

$$\boldsymbol{S}_i^{\mathrm{cen}}(t) = \frac{1}{|\boldsymbol{C}_i(t)|} \sum_{\boldsymbol{p}_j \in \boldsymbol{C}_i(t)} \boldsymbol{p}_j(t). \tag{14}$$

Next, the target position $\boldsymbol{o}_i(t)$ of the ith robot is calculated as follows:

$$\boldsymbol{o}_i(t) = \boldsymbol{p}_i(t) + \alpha \times (\boldsymbol{S}_i^{\mathrm{cen}}(t) - \boldsymbol{p}_i(t)), \tag{15}$$

where the parameter $\alpha \in (0,1)$.

(e) Dispersion (Beh_5). The dispersion behavior can alleviate the congestion caused by the decision-making of robots independently and also avoid the collision of robots. Each robot moves away from the gathering center. The target position $\boldsymbol{o}_i(t)$ of the ith robot is calculated as follows:

$$\boldsymbol{o}_i(t) = \boldsymbol{p}_i(t) + \beta * (\boldsymbol{p}_i(t) - \boldsymbol{S}_i^{\mathrm{cen}}(t)), \tag{16}$$

where the parameter $\beta \in (0,1)$.

(f) The swap based on greedy selection (Beh_6). The swap based on greedy selection is employed to determine the target position for each robot and assess whether to swap their target positions with its nearest neighbor robot. First, each robot records the positions to be accessed by its own and neighbors at time t. The information of $S_{i^*}^{\mathrm{rec}}(t)$ of the nearest robot of the ith robot is also recorded by the ith robot.

If the ith robot accesses the same target position with its neighbors ($o_i(t) = o_j(t)$) and is farther away from its target position than that of the neighbors ($\|p_i(t)-o_i(t)\| > \|p_j(t)-o_j(t)\|$), it will select its new target position $\boldsymbol{o}_i(t)$ as follows:

$$\boldsymbol{o}_i(t) = \underset{g_j \in G, g_j \notin \boldsymbol{S}_i^{\mathrm{rec}}(t), g_j \notin \boldsymbol{S}_{i^*}^{\mathrm{rec}}(t)}{\arg\min} \|\boldsymbol{p}_i(t) - g_j\|. \tag{17}$$

Then, the robot uses swap behavior to reduce the navigation distance. It should be noted that only two robots are each other's nearest neighbors, and the robot swaps its target positions with its nearest neighbor.

(g) Combinational behaviors (Beh_7). To further explore new behaviors to guide robots to select their target positions independently based on propositions and make swarm robots generate some emergent phenomena beyond thought, we designed some combinational behaviors to further enrich behaviors in the terminal set. In combinational behaviors, the target position $\boldsymbol{o}_i(t)$ of the ith robot is calculated as follows:

$$o_i(t) = \gamma_1 \times o_i^{\mathrm{Resolver}}(t) + \gamma_2 \times o_i^{\mathrm{Swap}}(t) + \gamma_3 \times o_i^{\mathrm{Aggregation}}(t) + \gamma_4 \times o_i^{\mathrm{Dispersion}}(t) + \gamma_5 \times o_i^{\mathrm{Greed}}(t), \tag{18}$$

where $\sum_{j=1}^{5} \gamma_j = 1$, $0 < \gamma_j < 1, j = 1, 2, \ldots, 5$. $o_i^{\mathrm{Resolver}}(t)$, $o_i^{\mathrm{Swap}}(t)$, $o_i^{\mathrm{Aggregation}}(t)$, $o_i^{\mathrm{Dispersion}}(t)$, $o_i^{\mathrm{Greed}}(t)$ represent Beh_2, Beh_3, Beh_4, Beh_5, Beh_6, respectively.

3.1.2 *Function set*

The function set comprises propositions that define the logical conditions used by robots to make informed decisions about their behaviors. These propositions incorporate correlation parameters and the states of robots. By judging these propositions, each robot can assess its current conditions and determine the appropriate behavior to take.

Table 1 provides detailed illustrations and notations for the propositions included in the function set. Propositions are constructed to assist robots in selecting appropriate behaviors based on their own states and that of their neighbors. The function set includes four propositions: $S_i^{\mathrm{acc}} < D_s$, $S_i^{\mathrm{nei}} < D_m$, $S_i^{\mathrm{agg}} < D_f$, and $|S_i^{\mathrm{ang}} - \theta_i| < D_g$. Each behavior may be related to a proposition or several propositions. There is also a certain order of precedence between propositions.

3.2 Initialization

The initialization stage mainly includes two parts: initializing the population and generating shape instances.

First, the ramped half-and-half algorithm is used to generate the initial population in the proposed DSFMA. By specifying a range of depth, half of the total population and the other half are generated by the grown and half algorithm, respectively. The ramped half-and-half algorithm not only removes the constraint of single tree depth but also increases the diversity of trees.

Second, a shape generator is designed to generate shape instances with different types, number of target points and spacing of target points. These shapes share common features, and our objective is to assess the generalization capability of strategies among shapes with different types. To achieve this, shape instances are categorized into different scenarios. Within each scenario, shape instances are further divided into training and test instances. More specifically, each scenario includes various shape types, all possessing the same number of target points and target point spacing. Training instances are used to evolve strategy, while test instances are employed to evaluate strategy performance. The training instances within each scenario are split into 20 mini-batches denoted as $\Gamma_{\text{train}}^{\text{gen}}$. Each mini-batch is used to evolve strategies within different individuals (e.g., instances of $\Gamma_{\text{train}}^{1}$ in individuals 1, 21, 41, 61, 81). In summary, each scenario requires the training of a specific strategy. Consequently, each robot is guided by the optimized behavior strategy to accomplish the DSF task.

3.3 Execution stage

In the execution stage, each robot is guided by a strategy, and performance data of strategies are recorded in Data. Because training instances of a mini-batch are used to evaluate strategies, performance data Data_j of the jth robot includes task execution results of all instances of a mini-batch and is given as follows:

$$\text{Data}_j = \{\text{res}_1, \text{res}_2, \ldots, \text{res}_k\}, \tag{19}$$

where k is the number of shape instances in a mini-batch, and res_k is the task execution result of the strategy in the kth training instance.

Once training instances of a mini-batch and strategy are given, each robot uses the strategy to select and move toward its target position until the DSF is completed or failed.

At the beginning of population iteration, some strategies cannot guide robots to achieve the DSF task. It is mainly because some robots may collide with their neighbors or fall into a deadlock state, in which some robots neither collide with their neighbors nor reach their target positions. So, the value of res_k can be selected form $\left\{t_k^j, \text{col}_k^j, \text{overrun}_k^j, \text{col\_overrun}_k^j\right\}$. Each possible task execution result is defined as follows:

• t_k^j is the time that each robot is guided by the jth strategy to complete the DSF task in the kth training instances of a mini-batch. The completion time of the DSF is defined as

$$t_k^j = \max_{i=1,2,\ldots,M} \{t_i\}, \tag{20}$$

where t_i presents the time when the ith robot arrives at its target position. $\text{data}_j^{\text{time}}$ is the set composed of the time t_k^j that can complete the DSF in the training instances.

• col_k^j is the number of collisions of all robots. In this case, although all robots are guided by the jth strategy to arrive at their target positions, some robots may collide with their neighbors. If the value of res_k is col_k^j, col_k^j is recorded in the set $\text{data}_j^{\text{col}}$, where $k = 1, 2, \ldots, |\Gamma_{\text{train}}^{1+(j-1)\%20}|$.

• overrun_k^j represents the situation where all robots, following the jth strategy, are unable to complete the DSF task without colliding with their neighbors. $\text{data}_j^{\text{overrun}}$ records the number of events in which the task execution result is overrun_k^j within the mini-batch denoted as $\Gamma_{\text{train}}^{1+(j-1)\%20}$.

• col_overrun_k^j represents that all robots guided by the jth strategy cannot complete the DSF task but some of them collide with their neighbors. $\text{data}_j^{\text{col\_overrun}}$ records the number of events that task execution result is col_overrun_k^j in the mini-batch $\Gamma_{\text{train}}^{1+(j-1)\%20}$.

After the performance data of a behavior strategy is recorded, all robots are reinitialized and guided by the next strategy. Once the performance data of all individuals (strategies) has been recorded, the DSFMA algorithm is used to evolve all individuals of the current population.

3.4 Evolution process

The proposed DSFMA will generate and evolve individuals. The details of the proposed DSFMA are described as follows.

3.4.1 *Selection operation*

The selection operation is used to generate the parent individuals for crossover or/and mutation. It is worth mentioning that the tournament method has the characteristics of adjustable selection pressure, so the method is used to select the parent individuals in this paper. To elaborate, within the current population, each individual is chosen as a parent, and a second parent is determined using the tournament method. Specifically, three individuals are randomly selected from the current population, and the one with the highest fitness value is designated as the second parent.

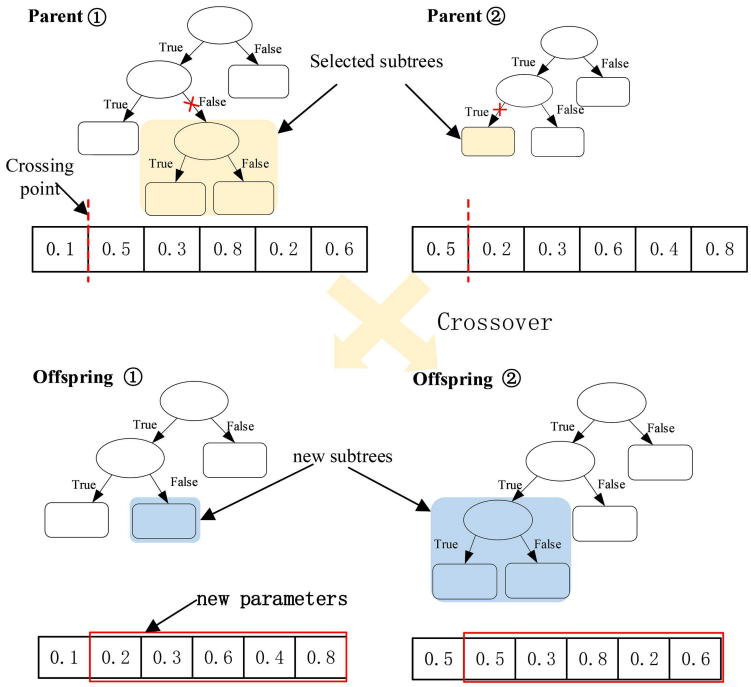

Figure 6 (Color online) Two parent individuals generate two offsprings by crossover operation. The yellow boxes represent subtrees randomly selected by the crossover operation in the parent individuals. The red dashed lines indicate the location of the single point crossing in the parameter to be optimized.

3.4.2 *Crossover operation*

The crossover operation is a typical genetic operator that involves swapping genes between parent individuals to generate new offspring. Given the tree structure-based decoding of the strategy, the single-point crossover method is employed. Furthermore, two parent individuals select subtrees at random and exchange them to produce new individuals. It is important to note that the optimization of tree structure also involves optimizing certain threshold parameters within the strategy. Consequently, while optimizing the tree structure, these parameters are also fine-tuned using the single-point crossover method. For instance, two parent individuals generate two offspring through the crossover operation, as depicted in Figure 6.

Following the crossover operation, the question arises as to which individuals should be retained in the next generation. Since performance data for each individual is obtained through multi-agent simulations, it will undoubtedly greatly prolong the time of population evolution if the simulation evaluation is carried out on the individuals after crossover or/and mutation. Therefore, the offspring generated by parent individuals with higher fitness values are selected to progress to the next generation.

3.4.3 *Mutation operation*

The mutation operation will be used to randomly generate individuals after the crossover operation and keep the population diversity. In the mutation operation, a randomly selected subtree and parameter are replaced with the newly generated ones. As shown in Figure 7, the blue boxes represent the randomly generated subtree and parameter, while the yellow boxes represent the randomly selected subtree and parameter. The parent individual utilizes the mutation operator to generate new offspring, enhancing the population diversity.

3.4.4 *Local search*

The LS can be used to improve the population. It is important to note when and how the LS operation is triggered. Referring to the trigger mechanism of the common LS operator, if the maximum objective value found yet has not been updated for the last $\mathcal{L}_{\max}$ iterations, the LS operation is activated to generate additional individuals within the population. Furthermore, the specifics of the LS operation are described as follows:

- The best individual found yet is marked as $\mathcal{I}_{\text{best}}$. A random real number in the range of $[0, 1]$ is generated and labeled as p_{ls}.
- If $0 \leqslant p_{ls} \leqslant \delta_{ls}$, two randomly selected subtrees from the individual $\mathcal{I}_{\text{best}}$ are swapped.
- If $\delta_{ls} < p_{ls} \leqslant 1$, insert a randomly selected subtrees from the individual $\mathcal{I}_{\text{best}}$ at the location randomly selected by the individual $\mathcal{I}_{\text{best}}$.

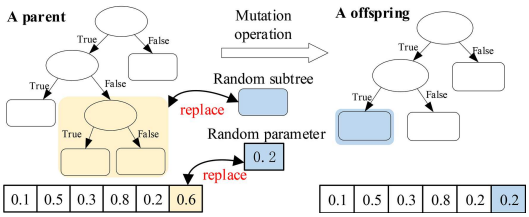

Figure 7 (Color online) Parent individual generates a offspring by mutation operation.

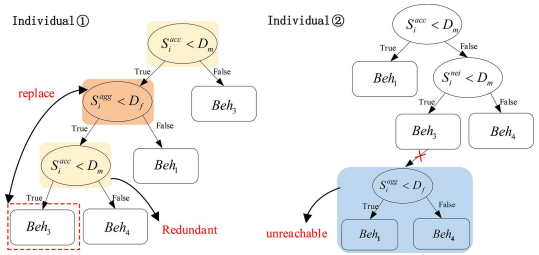

Figure 8 (Color online) Two behavior strategies to be repaired. The yellow and blue boxes represent redundant and unreachable subtrees.

3.4.5 *Repair mechanism*

The repair mechanism is proposed to cut redundant or unreachable subtrees in a strategy. Redundant subtrees mean that the data (proposition) of a child node is the same as the data (proposition) contained in the subtrees of the child node. An unreachable node refers to a node whose data content of the parent node is a behavior. In Figure 8, two strategies to be repaired are given.

In the proposed DSFMA, there are two reasons for the existence of redundant or unreachable subtrees in behavior strategies. On the one hand, the ramped half-and-half algorithm will generate some redundant subtrees within individuals in the stage of population initialization. On the other hand, some unreachable subtrees will be generated within individuals under the operation of crossover, mutation, or local search. The details of the repair mechanism are described as follows:

• For any child node (called the first node), if a child node (called the second node) of the first node has the same proposition as the first node, the tree structures between the two nodes are deleted, and the subtrees of the second node replace those of the first node.

• If a node is an unreachable node, the repair mechanism will delete the left and right subtrees of the node.

3.4.6 *DSF-specific evaluation mechanism of strategies*

Due to the swarm robots encountering collisions and congestion conditions during the execution of the DSF, it is one-sided to evaluate the performance of strategies only depending on whether the swarm can complete the DSF task. Therefore, a DSF-specific solution evaluation mechanism is proposed in this paper, which takes into account the situation in which robots can not complete the DSF task and the total number of collisions of robots.

In the evaluation mechanism, the maximum runtime is used to penalize strategies that can not guide robots to complete the DSF or guide robots to take a long time for the DSF. On the one hand, some robots make reciprocating motions, so that swarm cannot complete the DSF. On the other hand, robots guided by a poor strategy may take a long time to complete DSF. At the same time, whether robots complete the DSF after several collisions is integrated into the solution evaluation mechanism. More specifically, individuals with lower average task completion time have the highest priority. If the performance data of individuals have the same average task completion time or cannot guide robots to complete the DSF, the individuals with fewer collisions have the higher priority. Finally, the individuals whose task timed out have a higher priority than the individuals whose task timed out and collided. The details of the evaluation mechanism are shown as follows:

• If the set $\mathrm{data}_j^{\mathrm{time}}$ is not empty, the average task completion time dat_j is calculated as follows:

$$\mathrm{dat}_j = \frac{1}{\left|\mathrm{data}_j^{\mathrm{time}}\right|} \sum_{t_k^j \in \mathrm{data}_j^{\mathrm{time}}} t_k^j, \tag{21}$$

where $k = 1, 2, \ldots, |\Gamma_{\mathrm{train}}^{1+(j-1)\%20}|$. The smaller the value of dat_j, the higher the priority of the jth individual.

• If the set $\mathrm{data}_j^{\mathrm{time}}$ is empty, the average number of collisions $\mathrm{dat}_j^{\mathrm{col}}$ is given as follows:

$$\mathrm{dat}_j^{\mathrm{col}} = \frac{1}{\left|\mathrm{data}_j^{\mathrm{col}}\right|} \sum_{\mathrm{col}_k^j \in \mathrm{data}_j^{\mathrm{col}}} \mathrm{col}_k^j, \tag{22}$$

where $k = 1, 2, \ldots, |\Gamma_{\mathrm{train}}^{1+(j-1)\%20}|$. The smaller the value of $\mathrm{dat}_j^{\mathrm{col}}$, the higher the priority of the jth individual.

• If the set $\mathrm{data}_j^{\mathrm{time}}$ and $\mathrm{dat}_j^{\mathrm{time}}$ are empty, the smaller $\mathrm{data}_j^{\mathrm{overrun}}$, the higher the priority of the jth individual. If two individuals have the same value of $\mathrm{data}^{\mathrm{overrun}}$, the individuals with the smaller value of $\mathrm{data}^{\mathrm{col\_overrun}}$ have the higher priority.

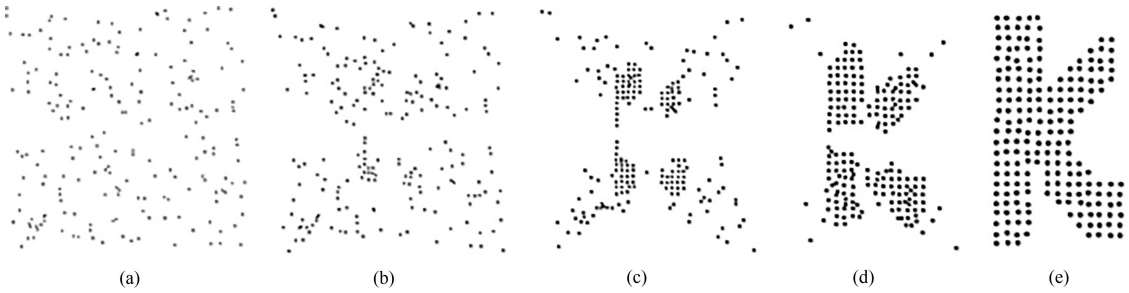

Figure 9 200 robots try to form shape K in the simulation. (a) 0 s, (b) 89 s, (c) 199 s, (d) 310 s, and (e) 330 s.

4 Experiments and results

The performance evaluation of the evolved strategy $f_*(\cdot)$ is presented in this section. First, data sets are given to provide training and testing instances of various scales. Then, the comparison algorithms are briefly described. Besides, the parameter setting and numerical results are given. In addition, the effectiveness of DSFMA is analyzed, and the behaviors or states that contribute most to the performance of the DSF are further found by statistical analysis and comparison experiments. Finally, the computational complexity of the generated behavior strategy is analyzed.

4.1 Data sets

To improve the generalization of strategies, that is, the strategies can guide each robot to complete the DSF task of different types of shapes; 40 different shapes are designed in the training set. The training instances include closed and non-closed English letters, numbers, and Chinese characters. At the same time, 12 scenarios are designed to analyze the algorithm more comprehensively. All scenarios are named by the number (u) of robots and the spacing (s) of pixels in a shape, e.g., u50s10. u is chosen from $\{50, 100, 150, 200\}$, and s is chosen from $\{10, 15, 20\}$.

In each scenario, 40 different shapes, which satisfy the constraints of the number and the spacing, are used for obtaining the performance data of the population. The proposed DSFMA is used to evolve individuals of the population.

After the best individual $f_*(\cdot)$ found yet is acquired, its performance will be evaluated on test instances, such as Figure 9. More specifically, the types of 15 test instances are different from that of training shapes. For example, u50s10n1 represents that 50 robots test the performance of $f_*(\cdot)$ in the shape labeled 1 with the spacing of 10 m.

4.2 Algorithms for comparison

To verify the effectiveness and superiority of the best strategy $f_*(\cdot)$ found yet. The following algorithms are compared with $f_*(\cdot)$.

(1) LTS: The state-of-art distributed algorithm [13] which includes the motion planner and the goal selector. The workspace is transformed into a discrete grid. Each robot has the same speed $v^{\max}$.

(2) DUD: The shape formation approach based on the artificial method [26]. The attraction field keeps all robots in a given shape, and the repulsive force filed among robots makes them distribute uniformly.

To illustrate the superiority of the proposed DSFMA algorithm, the following standard algorithms are compared with DSFMA.

(3) GP_T: The traditional GP approach using tournament selection with size 7. This is the standard GP for dynamic flexible job shop scheduling. The operations use the common operators [34].

(4) GP_LS: The GP approach with designed repair and evaluation mechanism but without LS operator.

(5) GP_RM: The GP approach with designed LS operator and evaluation mechanism but without repair mechanism.

(6) GP_EM: The GP approach with a designed repair mechanism and LS but without a designed evaluation mechanism.

Remark 2. GP_T is the standard algorithm. DSFMA is compared with GP_LS, which is helpful in verifying the effectiveness of the proposed LS operator. To verify the effectiveness of the repair mechanism and evaluation mechanism, DSFMA is compared with GP_RM and GP_EM, respectively.

4.3 Parameter setting

The proposed DSFMA is executed based on the AnyLogic simulation software with Java code. The parameter settings of DSFMA follow the general settings [29,34] as shown in Table 2. All targets and robots are initialized in an 800 m $\times$800 m space. Since the small unit grid is too dense to be seen clearly on the AnyLogic simulation, the

Table 2 Parameter setting of DSFMA

Parameter	Value
Population size	100
Number of generations	50
Initial population depth	4
Population initialization method	ramp-half-and-half
Parent selection	Tournament selection
Crossover rate	0.9
Mutation rate	0.05
$\mathcal{I}_{\text{best}}$	2, 5, 8, 11, 14
δ_{ls}	0.1, 0.5, 0.9
The workspace of robots	800 m × 800 m
$v^{\max}$	5 m/s
$a^{\max}$	2 m/s^2
R	50 m

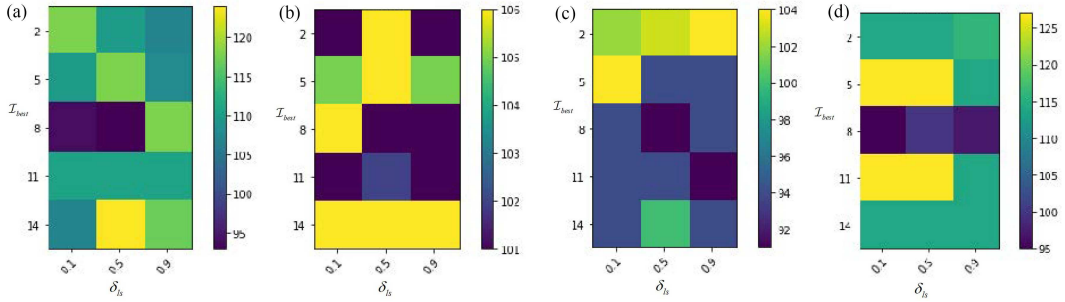

Figure 10 (Color online) Heat maps of test performance of the proposed DSFMA with different values of the parameter $\mathcal{I}_{\text{best}}$ and δ_{ls} on four scenarios. (a) u100s10; (b) u100s15; (c) u100s20; (d) u150s10.

Table 3 12 parameter values obtained using the DSFMA in three scenarios

Scenario	D_f	D_s	D_m	D_g	S_v	α	β	γ_1	γ_2	γ_3	γ_4	γ_5
u100s10	0.78	0.01	0.818	0.655	0.361	0.734	0.536	0.74	0.092	0.047	0.420	0.946
u150s10	0.36	0.373	0.993	0.735	0.08	0.972	0.088	0.043	0.045	0.013	0.923	0.031
u200s10	0.18	0.731	0.418	0.735	0.08	0.972	0.905	0.74	0.323	0.013	0.504	0.72

communication radius R and speed of a robot are proportionally enlarged based on the settings in [13]. Table 2 shows the relevant parameters of the robot.

To perform a comprehensive sensitivity analysis of $\mathcal{I}_{\text{best}}$ and δ_{ls}, the parameter $\mathcal{I}_{\text{best}}$ ranges from $\{2, 5, 8, 11, 14\}$, while δ_{ls} ranges from $\{0.1, 0.5, 0.9\}$. In Figure 10, we observe heat maps showing the test performance of the proposed DSFMA for different values of the parameters $\mathcal{I}_{\text{best}}$ and δ_{ls} across four scenarios. Darker colors represent smaller values, indicating better performance, while lighter colors represent larger values, indicating worse performance. The different scenarios exhibit varying distributions of test performance in Figure 10. However, the best regions of the test performance for all scenarios are located around the center basically. Among these 15 settings for each scenario, $(0.5, 8)$ configuration outperforms the others. Hence, the parameters $\mathcal{I}_{\text{best}} = 8$ and $\delta_{ls} = 0.5$ are adopted in the following experiments. Table 3 shows the 12 parameter values obtained by DSFMA for the three scenarios.

4.4 Numerical results and comparison of strategies

In this part, the best individual $f_*(\cdot)$ found yet is used to compare with the LTS. It is worth noting that each scenario corresponds to the best individual $f_*(\cdot)$ found yet. Figure 11 shows the ratio of the DSF task completion time to the maximum task completion time of 10 independent runs on the test instances. Each scenario includes 15 shape instances. The blue, yellow, and orange lines represent the DSF completion time of robots guided by the strategy, LTS, and DUD, respectively.

In most instances, the completion time of the strategy (blue line) is significantly better than that of the LTS (yellow line) and DUD (orange line) based on the Wilcoxon rank sum test with a significance level of 0.05. This improvement can primarily be attributed to the target conflict resolver with orientation preference, which enables each robot to select its target position based on its historical memory and moving speed, as opposed to randomly

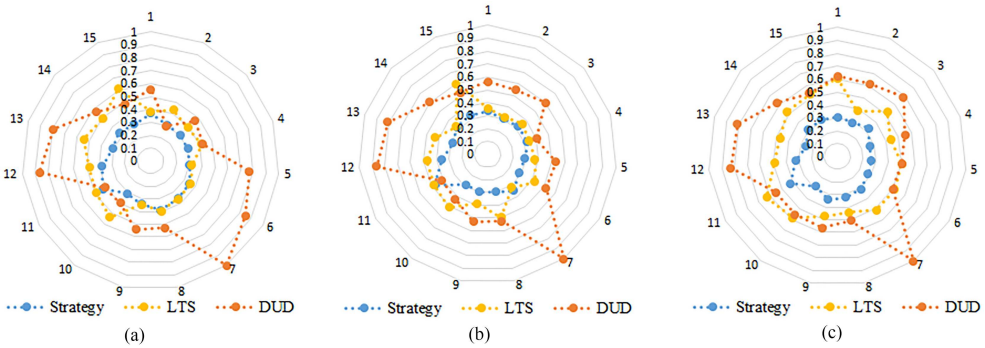

Figure 11 (Color online) Ratio of the DSF task completion time to the maximum task completion time of 10 independent runs. The strategy is generated by DSFMA. (a) u100s10; (b) u150s10; (c) u200s10.

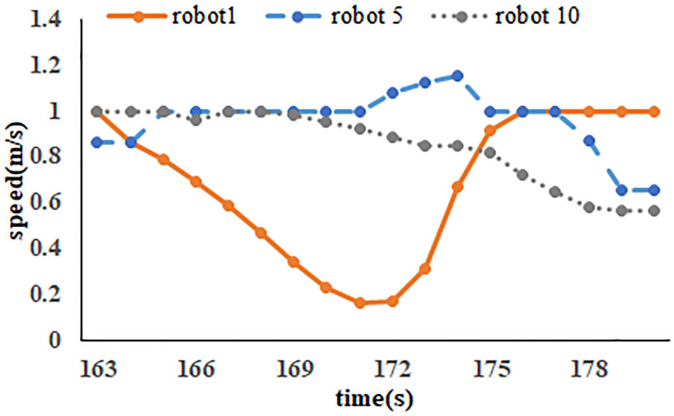

Figure 12 (Color online) Curves of the speeds of three robots.

choosing unvisited targets as in LTS. As the number of robots increases, the DUD algorithm prolongs the process of moving robots from non-uniform areas to uniform areas. The generated strategy including AVO can predict speeds of neighbors of robots to reduce the reciprocating path caused by avoiding neighbors. The speed curves of three robots are presented in Figure 12. The speeds of three robots satisfy the limitation of the maximum speed, and their velocity change rates adhere to the limitations of maximum acceleration.

To comprehensively analyze the performance of the generated strategies, each strategy is employed to guide each robot to achieve DSF 10 times on test instances of each scenario. The average and standard deviation of the completion time are recorded for each instance. As depicted in Figure 13, there are a total of 12 scenarios. From the overall trend of the curves, the standard deviation of task completion time is relatively stable. In the case of 50 robots, instances 5 and 11, characterized by a large spacing of robots, have a larger standard deviation than the others. This suggests that the initial distribution of robot positions has a significant impact on the completion time, particularly for sparse and asymmetrical shapes with fewer robots. When the spacing between robots is small (indicated by the blue solid line), the average completion time for the letter F shape is longer than for the others. On one hand, congestion may occur among robots when they are closely spaced. On the other hand, the sparse and asymmetrical shape F has more void spaces within its middle section. Each robot will move through these blank spaces to find its target position to be accessed. As the increase of the number of robots, the average task completion time experiences a gradual increase. In summary, both the spacing and the number of robots are crucial factors that influence the DSF completion time.

4.5 Analysis of the proposed DSFMA

In this section, the four comparisons in terms of test performance verify the effectiveness of the LS operator, repair mechanism, and the DSF-specific evaluation mechanism, respectively. GP_T is the baseline GP with the standard tournament selection. The curves fluctuate because of the shape instances sampled from the training set in different generations.

4.5.1 *Effectiveness of LS operator*

To verify the effectiveness of the LS operator, DSFMA is compared with GP_LS and the standard GP named GP_T.

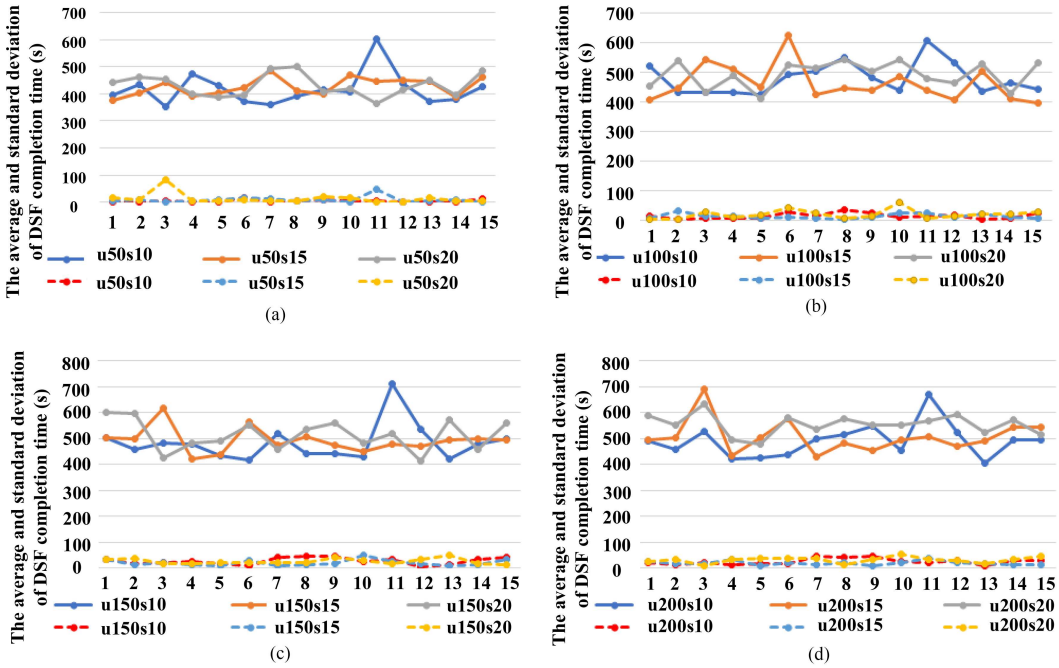

Figure 13 (Color online) Average and standard deviation of completion time of generated behavior strategies in different instances. The solid and dotted lines represent the average and standard deviation, respectively. (a) 50 robots; (b) 100 robots; (c) 150 robots; (d) 200 robots.

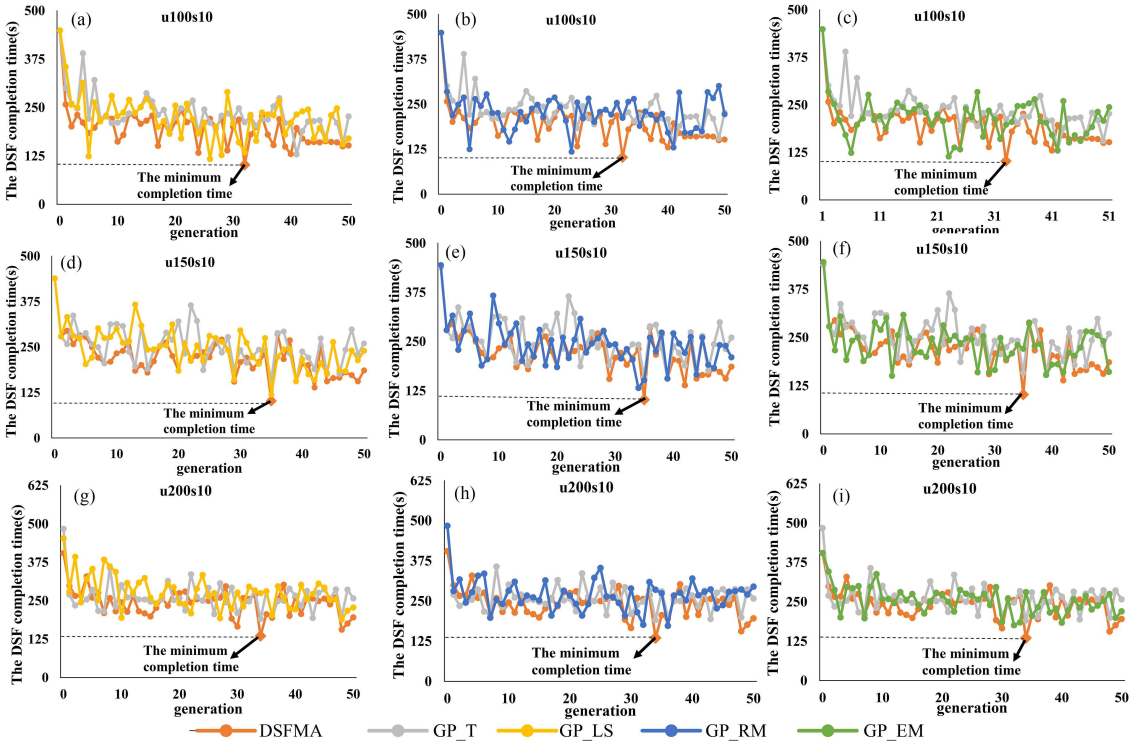

Figure 14 (Color online) Convergence curves of the test performance of compared algorithms on different scenarios. (a–c) u100s10; (d–f) u150s10; (g–i) u200s10.

Figures 14(a), (d), and (g) show the mean of the DSF completion time of 10 independent runs of DSFMA, GP_T and GP_LS on the 3 scenarios. DSFMA and GP_LS have the better test performance than GP_T on the 3 scenarios. The convergence curves of DSFMA are always below that of GP_LS and GP_T for all the scenarios. The minimum DSF task completion time found yet is always in DSFMA on the 3 scenarios. It is mainly because

Table 4 Representation relationships between elements and numbers in terminal and function set

Num	Type
1	$S_i^{\text{agg}}\&D_f$
2	$S_i^{\text{acc}}\&D_s$
3	$S_i^{\text{nei}}\&D_m$
4	$\left\|S_i^{\text{ang}}-\theta_i\right\|\&D_g$
5	The target conflict resolver
6	Dispersion
7	The swap based on greedy selection
8	Aggregation
9	Swap
10	Combinational

the tree-based local search (LS) operator is used to exploit the neighborhood of the best behavior strategy found yet during the iteration. Overall, DSFMA algorithm outperforms GP_T and GP_LS.

4.5.2 *Effectiveness of repair mechanism*

To verify the effectiveness of the repair mechanism, DSFMA is compared with GP_RM and GP_T.

From Figures 14(b), (e), and (h), the convergence curves of DSFMA are always below that of GP_RM and GP_T for all the scenarios. The proposed repair mechanism is used to cut redundant or unreachable subtrees in the behavior strategy. During the process of population evolution, multiple high-quality subtrees will hope to remain in different individuals, rather than one individual having many duplicate subtrees. Compared with DSFMA, GP_RM has a larger fluctuation during the late iteration. On the one hand, unreachable subtrees will cause the DSF task to fail, and multiple high-quality genes of these unreachable subtrees will be deleted in the population. On the other hand, invalid cross operation will increase as the multiple duplicate subtrees of individuals. It is important to cut redundant or unreachable subtrees from the behavior strategy. Overall, DSFMA outperforms GP_T and GP_RM.

4.5.3 *Effectiveness of DSF-specific evaluation mechanism*

To verify the effectiveness of the repair mechanism, DSFMA is compared with GP_EM and GP_T.

From Figures 14(c), (f), and (i), the convergence curves of GP_EM are always below that of DSFMA and GP_T for all scenarios. It is mainly because GP_EM evaluates the performance of behavior strategies only depending on whether the swarm can complete the DSF. Individuals who are unable to complete the task are eliminated by the tournament, so GP_EM is prone to premature convergence. In the late iteration, the convergence curves of DSFMA are always below that of GP_EM and GP_T for all the scenarios. Due to the swarm robots encountering collisions and congestion conditions during the execution of DSF, it is one-sided to evaluate the performance of behavior strategies only depending on whether the swarm can complete the DSF task. Hence the DSF-specific evaluation mechanism considers the situation that robots cannot complete the DSF task and the total number of collisions of robots. Hence, the high-quality genes of individuals are retained in the population. Overall, DSFMA algorithm outperforms GP_T and GP_EM.

4.5.4 *Feature analysis*

To comprehensively analyze the significant influence of propositions and behaviors on behavior strategies, we record the number of propositions and behaviors that appear in the best behavior strategy found within each population during the iteration.

Representation relationships between elements and numbers in terminal and function set are shown in Table 4. The numbers 0–4 sequentially represent four propositions in the terminal set, respectively, and the numbers 5–10 sequentially represent six behaviors in the function set, respectively.

From Figure 15, the swap based on greedy selection behavior is the most frequently of appearing element in the best behavior strategy found in each population across three scenarios. It is mainly because the behavior not only can solve the selection conflict of target positions among neighbors but also can reduce the navigation length and ease the congestion of robots.

4.5.5 *Computational complexity*

In this part, we discuss the computational complexity of the optimized strategy and LTS algorithm [13].

Remark 3. It should be noted that DSFMA is used to offline optimize strategies for each scenario using training shapes. Hence, the solution to be implemented in a real-time manner refers to the optimized behavior strategy.

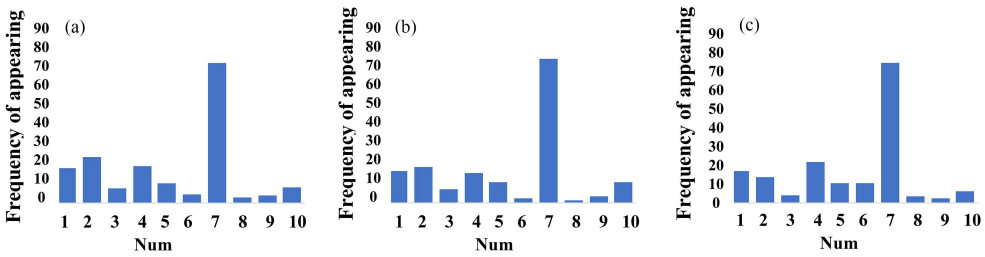

Figure 15 (Color online) Number of behaviors and propositions present in the best strategy found in each population across three scenarios. (a) u100s10; (b) u150s10; (c) u200s10.

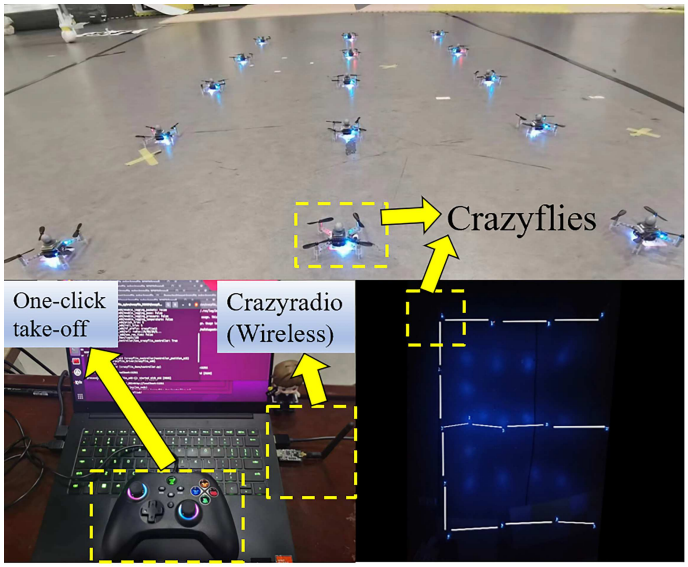

Figure 16 (Color online) 15 Crazyflies form the shape E. A crazyradio is used to receive and transmit data. A hand shank is used to control 15 Crazyflies take-off.

After acquiring the optimized strategy in each scenario including different training shape types, each robot is guided by it to achieve test shapes in real time. The test shapes are different from the training shapes. Since DSFMA does not directly control the robots to complete DSF, but rather the strategy optimized by DSFMA, the comparison should be made between the optimized strategy and the state-of-the-art methods [13].

A behavior strategy includes propositions and behaviors as shown in Figure 2. Child nodes of tree structures consist of propositions, while leaf nodes consist of behaviors. Each robot uses the strategy to select a behavior based on the propositions.

There are seven designed behaviors in this paper. Beh_1 is used to calculate the new velocity at the next time. Other behaviors are designed to select or change the target position of each robot. The Beh_7 which is the combination behaviors including $\mathrm{Beh}_2, \mathrm{Beh}_3, \mathrm{Beh}_4, \mathrm{Beh}_5, \mathrm{Beh}_6$, has the maximum computational complexity in these behaviors. The parameter setting and computational complexity of Beh_1 are the same with [32]. As the description in [32], though the number of robots is up to 1000, new velocity computation took under 30 ms and the simulation ran at over 30 FPS. In behavior Beh_7, each robot uses the positions and the target positions of its neighbors, as well as the target label accessed by its nearest neighbor robot. The maximum number of neighbors of a robot is $M-1$. The computational complexity of Beh_7 is $O(2M-2)$. Each robot is guided by the behavior strategy to select a concrete behavior. Hence, the maximum computational complexity of a behavior strategy is $O(2M-2)$.

In the LTS algorithm, all robots work in a graph with unit edge lengths and capacities. This implies that an edge takes a unit of time to cross. In the LTS algorithm, each robot can have at most $\lfloor \frac{2R}{l} \rfloor^2$ [13] amount of neighbors. l is the unit edge length. The complexity of the decision-making time of a robot at each moment is $O(\lfloor \frac{R}{l} \rfloor^2)$ [13].

4.5.6 *Physical experiment*

To verify the efficiency of the optimized strategy, a custom-made swarm comprising 15 Crazyflies 2.1 is employed to execute the DSF in a 6 m × 4 m workspace, as illustrated in Figure 16. Localization is achieved by the NOKOV

motion capture system.

A computer workstation, utilizing separate processes for each robot, receives and transmits its own position data. Commands are transmitted directly to the Crazyflies 2.1 via their custom radio communication. 15 Crazyflies 2.1 are guided by the generated strategy to form English letters B, E, and N, respectively. The recordings of the experiments are included on the website[1)].

5 Conclusion

In this paper, a novel genetic programming algorithm is proposed to generate and evolve strategies that can guide each robot to make decisions for DSF in real time. Through experimental data statistics, the completion time of the generated behavior strategy is significantly better than that of the LTS and DUD algorithms. In the future, we will further study the shape assembly problem which is free of target assignment and do not need label of robots.

Acknowledgements This work was supported by Basic Science Center Programs of National Natural Science Foundation of China (Grant No. 62088101).

References

1 Rahman S R, Sajjad I, Mansoor M M, et al. School formation characteristics and stimuli based modeling of tetra fish. Bioinspir Biomim, 2020, 15: 065002
2 Yinka-Banjo C O, Owolabi W A, Akala A O. Birds control in farmland using swarm of UAVs: a behavioural model approach. In: Proceedings of Advances in Intelligent Systems and Computing, 2019. 333–345
3 Khaluf Y, Vanhee S, Simoens P. Local ant system for allocating robot swarms to time-constrained tasks. J Comput Sci, 2019, 31: 33–44
4 Grassé P P. La reconstruction du nid et les coordinations interindividuelles chezBellicositermes natalensis etCubitermes sp. la théorie de la stigmergie: Essai d'interprétation du comportement des termites constructeurs. Ins Soc, 1959, 6: 41–80
5 Li Y, Li K, Tong S. An observer-based fuzzy adaptive consensus control method for nonlinear multiagent systems. IEEE Trans Fuzzy Syst, 2022, 30: 4667–4678
6 Wu W, Li Y, Tong S. Neural network output-feedback consensus fault-tolerant control for nonlinear multiagent systems with intermittent actuator faults. IEEE Trans Neural Netw Learn Syst, 2023, 34: 4728–4740
7 Lv M, Li Y, Pan W, et al. Finite-time fuzzy adaptive constrained tracking control for hypersonic flight vehicles with singularity-free switching. IEEE ASME Trans Mechatron, 2021, 27: 1594–1605
8 Zhang Z S, Huangfu W, Long K P, et al. On the designing principles and optimization approaches of bio-inspired self-organized network: a survey. Sci China Inf Sci, 2013, 56: 071301
9 Li G, St-Onge D, Pinciroli C, et al. Decentralized progressive shape formation with robot swarms. Auton Robot, 2019, 43: 1505–1521
10 Chu W J, Zhang W, Zhao H Y, et al. Massive self-organized shape formation in grid environments. Sci China Inf Sci, 2022, 65: 164101
11 Liu Y, Liu J, He Z, et al. A survey of multi-agent systems on distributed formation control. Un Sys, 2024, 12: 913–926
12 Yang H A, Li Y H, Duan X, et al. A parallel shape formation method for swarm robotics. Robot Autonom Syst, 2022, 151: 104043
13 Wang H, Rubenstein M. Shape formation in homogeneous swarms using local task swapping. IEEE Trans Robot, 2020, 36: 597–612
14 Thalamy P, Piranda B, Bourgeois J. A survey of autonomous self-reconfiguration methods for robot-based programmable matter. Robot Autonom Syst, 2019, 120: 103242
15 Pinciroli C, Birattari M, Tuci E, et al. Self-organizing and scalable shape formation for a swarm of pico satellites. In: Proceedings of NASA/ESA Conference on Adaptive Hardware and Systems, 2008. 57–61
16 Shirazi A R, Jin Y. Regulated morphogen gradients for target surrounding and adaptive shape formation. IEEE Trans Cogn Dev Syst, 2020, 13: 818–826
17 Wang C, Shi Z, Gu M, et al. Revolutionary entrapment model of uniformly distributed swarm robots in morphogenetic formation. Defence Tech, 2024, 31: 496–509
18 Xin B, Zhang J, Chen J, et al. Overview of research on transformation of multi-AUV formations. Complex Syst Model Simul, 2021, 1: 1–14
19 Guo J, Lin Z, Cao M, et al. Adaptive control schemes for mobile robot formations with triangularised structures. IET Control Theor Appl, 2010, 4: 1817–1827
20 Aranda M, Lopez-Nicolas G, Sagues C, et al. Distributed formation stabilization using relative position measurements in local coordinates. IEEE Trans Automat Contr, 2016, 61: 3925–3935

1) https://t.hk.uy/b5Gg.

21 Macdonald E A. Multi-robot assignment and formation control. Dissertation for Master's Degree. Atlanta: Georgia Institute of Technology, 2011

22 Falconi R, Gowal S, Martinoli A. Graph based distributed control of non-holonomic vehicles endowed with local positioning information engaged in escorting missions. In: Proceedings of IEEE International Conference on Robotics and Automation, 2010. 3207–3214

23 Lin Z, Wang L, Han Z, et al. Distributed formation control of multi-agent systems using complex Laplacian. IEEE Trans Automat Contr, 2014, 59: 1765–1777

24 Chaimowicz L, Michael N, Kumar V. Controlling swarms of robots using interpolated implicit functions. In: Proceedings of the IEEE International Conference on Robotics and Automation, 2005. 2487–2492

25 van den Berg J, Lin M, Manocha D. Reciprocal velocity obstacles for real-time multi-agent navigation. In: Proceedings of IEEE International Conference on Robotics and Automation, 2008. 1928–1935

26 Bi Q, Huang Y. A self-organized shape formation method for swarm controlling. In: Proceedings of the 37th Chinese Control Conference, 2018. 7205–7209

27 Izzo D, Pettazzi L. Autonomous and distributed motion planning for satellite swarm. J Guidance Control Dyn, 2007, 30: 449–459

28 Zhang B T, Cho D Y. Evolving complex group behaviors using genetic programming with fitness switching. Artif Life Robot, 2000, 4: 103–108

29 Zhang F, Mei Y, Nguyen S, et al. Evolving scheduling heuristics via genetic programming with feature selection in dynamic flexible job-shop scheduling. IEEE Trans Cybern, 2020, 51: 1797–1811

30 Krasnogor N, Smith J. A tutorial for competent memetic algorithms: model, taxonomy, and design issues. IEEE Trans Evol Computat, 2005, 9: 474–488

31 Benlic U, Hao J K. Memetic search for the quadratic assignment problem. Expert Syst Appl, 2015, 42: 584–595

32 van den Berg J, Snape J, Guy S J, et al. Reciprocal collision avoidance with acceleration-velocity obstacles. In: Proceedings of IEEE International Conference on Robotics and Automation, 2011. 3475–3482

33 Fiorini P, Shiller Z. Motion planning in dynamic environments using velocity obstacles. Int J Robot Res, 1998, 17: 760–772

34 Gao G, Mei Y, Xin B, et al. Automated coordination strategy design using genetic programming for dynamic multipoint dynamic aggregation. IEEE Trans Cybern, 2021, 52: 13521–13535