

Accepted Manuscript

A Two-Stage Imitation Learning Framework for the Multi-Target Search Problem in Swarm Robotics

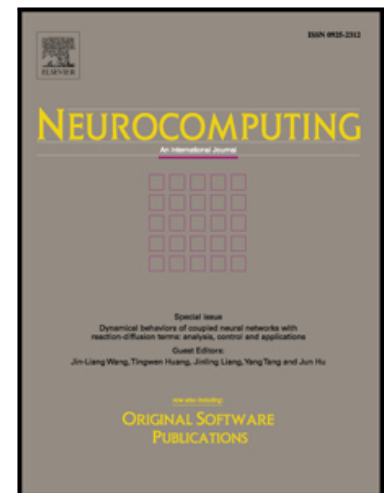
Jie Li, Ying Tan

PII: S0925-2312(19)30055-4
DOI: <https://doi.org/10.1016/j.neucom.2019.01.035>
Reference: NEUCOM 20339

To appear in: *Neurocomputing*

Received date: 31 March 2018
Revised date: 5 November 2018
Accepted date: 16 January 2019

Please cite this article as: Jie Li, Ying Tan, A Two-Stage Imitation Learning Framework for the Multi-Target Search Problem in Swarm Robotics, *Neurocomputing* (2019), doi: <https://doi.org/10.1016/j.neucom.2019.01.035>



This is a PDF file of an unedited manuscript that has been accepted for publication. As a service to our customers we are providing this early version of the manuscript. The manuscript will undergo copyediting, typesetting, and review of the resulting proof before it is published in its final form. Please note that during the production process errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

Highlights

- For the multi-target search problem in swarm robotics, most existing work is about strategic design while this article focuses on strategy imitation.
- To our knowledge, it is the first time that the combination of deep learning technologies and evolutionary algorithms is used for the problem.
- The strategy obtained from the framework is close to the target strategy on multiple indicators.
- The two-stage imitation learning framework can also be used for other swarm tasks.
- The network design and evolutionary algorithm settings can be a good reference.

A Two-Stage Imitation Learning Framework for the Multi-Target Search Problem in Swarm Robotics

Jie Li^{a,1}, Ying Tan^{a,*}

^aKey Laboratory of Machine Perception (Ministry of Education), Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing, 100871, P.R. China.

Abstract

As a distributed system with a large number of individuals, swarm robotics is particularly suitable for multi-target search problems. Most existing work is about strategic design while this article focuses on strategy imitation. Sometimes we can observe the behavior of individuals and obtain a large amount of data, but we do not know the specific details of the strategy behind the behavior. Imitating the self-organizing behavior of organisms is of great significance for us to design efficient swarm strategies and to reveal the underlying mechanisms. The actual strategy adopted by individuals can be called the target strategy, and in this article, a two-stage imitation learning framework is proposed to approach the target strategy. In the first stage, a deep neural network is trained using the behavioral data of individuals, and in the second stage, the parameters of the neural network are further fine-tuned using the evolutionary algorithm. After two stages of learning and evolution, the resulting strategy RNSE is very close to the target strategy in terms of multiple indicators, including search efficiency, stability, parallel processing capability, and collaborative processing capability. In addition to multi-target search, the framework can also be used for other collective tasks such as aggregation and dispersion. In this paper, the design of neural networks and the settings of the evolutionary algorithm are

*Corresponding author

Email address: ytan@pku.edu.cn (Ying Tan)

¹Email address: ustbljie@126.com

discussed in detail, which is of great significance for the migration application of the framework.

Keywords: Swarm robotics, Multi-target search, Swarm intelligence optimization, Deep learning, Evolutionary algorithm, Strategy imitation

1. Introduction

Swarm robotics is a relatively new field, inspired by natural self-organizing behaviors [1] such as ants foraging, bees nesting, and bird migration. In these swarms, the ability of each individual is simple, but through local interactions
5 among individuals and between individuals and the environment, the system can emerge complex behaviors at the swarm level. Compared to single robot or multi-robot systems, swarm robotics has some unique advantages, such as robustness, scalability, and flexibility [2, 3]. In addition, due to the simplicity and low cost of the individuals, swarm robotics has potential applications in
10 many fields, such as post-disaster rescue [4], intrusion detection [5], human interaction [6], and planetary exploration [7], etc.

As a distributed system, swarm robotics is very suitable for tasks involving area coverage, such as collaborative mapping [8], target search [9], area monitoring [10], and so on. And the multi-target search problem of swarm robotics
15 studied in this paper is also such kind of a task [11], in which the entire swarm can search multiple targets in parallel, thereby improving the search efficiency. Some practical scenarios can be abstracted as the problem, such as shipwreck rescue, submarine search, search and destruction of battlefield targets.

There are various research directions in swarm robotics [12], one of which
20 is the system design. There are roughly two types of design approaches [13]. One is the behavior-based approach, which is the most commonly used. It is a bottom-up approach, achieving the desired collective behavior through the realization, research and improvement of individual behaviors iteratively. And the specific methods including probability finite state machine [14], artificial
25 potential field [15], etc. The other is the learning-based approach (also called

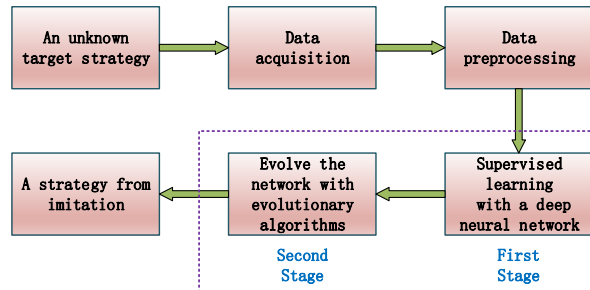


Figure 1: A two-stage imitation learning framework for the multi-target search problem in swarm robotics.

the automatic design method), which is generally based on some global indicators to adjust and improve individual behaviors from top to bottom. The specific methods mainly include reinforcement learning [16] and evolutionary robotics [17], both of which use the feedback from the environment to improve the individual behaviors.

The behavior-based design method is usually inspired by the observation of individual behaviors. If we can obtain a large amount of behavioral data but have no idea about specific details of the strategy followed by the individual, such as the AlphaGo's ignorance of the human expert's strategy [18], then we can learn the unknown target strategy through deep learning. Further, based on the metrics of the task, we can optimize the strategy to approach the performance of the target strategy. Thus, a two-stage imitation learning framework is proposed for the multi-target search problem in swarm robotics (Fig. 1), and for the first time some relevant technologies of deep learning are used for this task, including the ReLU [19], self-normalizing neural networks [20], stacked auto-encoders [21], dropout [22], etc. In the second stage, a recently proposed evolutionary algorithm GFWA (Guided Fireworks Algorithm) [23] is selected to optimize the strategy. After learning and evolution, the performance of the final strategy is very close to the target strategy.

The rest of the article is organized as follows. In section II, the multi-target search strategies and some of the tools in deep learning are introduced. In

section III, the multi-target search problem in swarm robotics and the target strategy are stated. In section IV, the two-stage imitation learning framework is described in detail, which is based on deep learning and evolutionary computation. In section V, the simulation results and discussions are presented. Finally, section VI summarizes the whole article.

2. Related Work

Current multi-target search strategies in swarm robotics are generally behavior-based, which can be roughly divided into two categories. One is inspired by swarm intelligence optimization algorithms, such as particle swarm optimization algorithm (PSO) [24], bee swarm optimization algorithm (BSO) [25], glowworm swarm optimization algorithm (GSO) [26], fireworks algorithm (FWA) [27], etc. The other is inspired by random walk strategies, such as Lévy flight [28], intermittent search [29]. There are still other search strategies, such as those based on artificial potential field [30, 31] and formation-based search strategies [32]. This section will introduce several strategies inspired by swarm intelligence optimization algorithms and a formation-based search strategy, some of which will participate in the comparative experiments.

- RPSO: Robotic Particle Swarm Optimization (RPSO) [24] is an extended version of the PSO algorithm, in which an obstacle avoidance component is considered in the velocity update formula. With the original RPSO, the robots may fall into the local oscillatory motions during the search, so a random component is introduced to alleviate the problem in this paper. It is worth noting that some improved strategies are proposed recently, such as the RDPSO (Robotic Darwinian PSO) [33] and RbRDPSO (Repulsion-based RDPSO) [34]. The RDPSO allows multiple dynamic swarms, and in RbRDPSO a new component based on the repulsion between similar ions is introduced to prevent robots to get too close to each other.
- A-RPSO: In the Adaptive Robotic PSO (A-RPSO) [35], the velocity update formula of each robot is similar to that in RPSO, but the inertia

weight is related to the individual and the number of iterations. In each iteration, the inertia weight of each individual robot depends on the “evolutionary speed” and “aggregation degree” [36]. The “evolutionary speed” is described by the difference in the fitness values of the best individuals in the adjacent two iterations. The “aggregation degree” is represented by the difference between the optimal fitness value and the average fitness in the current iteration. The original A-RPSO algorithm focuses on the single-target search problem. In this paper, the swarm is divided into multiple subpopulations by limiting the communication range of the robot, so that it can better adapt to the multi-target search problem. The velocity update formula also introduces a random component to avoid trapping in local oscillatory motions.

- GES: Group Explosion Strategy (GES) [27] borrowed some of the mechanisms of the fireworks algorithm (FWA) [37]. FWA is a swarm intelligence optimization algorithm inspired by fireworks explosion. In GES, the entire population is automatically divided into multiple groups according to each individual’s perception range. The key coordination mechanism is to move the geometric center of the group to one of the best individuals in the group. In order to improve the parallelism of the strategy, the group will split into two subgroups when the group size exceeds a certain threshold.
- IGES: Improved Group Explosion Strategy (IGES) [38] is an improved version of GES. In GES, if multiple individuals in a group have the same optimal fitness values, randomly selecting one may let the group fall into local oscillations or fall back to a worse region. In order to solve the problems, IGES combines four simple sub-strategies to deal with different situations. There are two main improvements in IGES: first, the reference position of the movement is no longer a random selection of an optimal individual, but the center of all optimal individuals; second, if the size of the group is greater than a certain threshold, or the group member share the same fitness value, the robots will move away from the center of the

group. The performance of IGES is much better than that of GES, so only IGES is used in the comparative experiments in this paper.

- TFS: In the Triangle Formation Search strategy (TFS) [32], the entire robot swarm is divided into three-robot formations as much as possible. Each team is roughly formed into an equilateral triangle, including a leader and two members. The team leader is responsible for determining the moving direction, and the team members will follow the team leader and maintain the formation. In order to enhance the exploration ability of the swarm, an initial diffusion stage is introduced in the strategy. During the search process, a triangulation technique is proposed to estimate the gradient direction and enhances the team's exploitation ability.
- PFSMS: This strategy serves as the target strategy in this paper, which is a probabilistic finite state machine based strategy. This strategy contains three states: diffusion, search, and target processing, and it will be introduced in section 3.3.

Most of the learning-based methods adopt evolutionary robotics, such as in foraging [39], aggregation [40], and the area monitoring [41]. Due to the credit assignment problem and large action space [13], the reinforcement learning is faced with many challenges in multi-robot systems. It is even more difficult to apply them to swarm robotics. In recent years, with the rapid development of deep learning technologies, some methods of multi-agent reinforcement learning have been proposed one after another [42, 43], and deep neural networks have also begun to be applied in the field of swarm robotics [44]. Next, we will briefly introduce the key technologies and algorithms used in this paper for deep learning and evolutionary computation.

- ReLU [19]: Compared with the most commonly used sigmoid neurons and hyperbolic tangent neurons, the rectifier activation function ($\text{rectifier}(x) = \max(0, x)$) can get a more sparse representation, which is conducive to training deep neural networks. It is difficult to train deep neural networks,

135 which generally requires unsupervised pre-training. With ReLU functions,
the deep networks can obtain good performance without pre-training.

- SNN [20]: The Self-Normalizing Neural Networks can be used for high-
level abstract representations. In SNN, the outputs of neurons can auto-
140 matically converge towards zero mean and unit variance. The activation
function used is called the “scaled exponential linear unit” (SELU), as is
shown in Eq. 1. The unit has self-normalized properties, which makes
it possible for the SNN to train deep networks with many layers, avoid-
ing gradient vanishment and explosion under certain conditions. In this
article, a comparison is made between ReLU and SELU.

$$selu(x) = \lambda \begin{cases} x & , \text{if } x > 0 \\ \alpha e^x - \alpha & , \text{if } x \leq 0 \end{cases} \quad (1)$$

145 - SAE [21]: The representation capability of the neural network will be
enhanced as the depth increases, but it’s harder to train a deeper neural
network. Stacked auto-encoders are useful for training deep networks [45],
especially for the unsupervised pre-training [46]. In this article, neural
networks with and without SAE pre-training are compared.

150 - dropout [22]: Dropout is an effective way to prevent neural networks from
over-fitting. Neural networks with a large number of parameters are pow-
erful machine learning tools, but the over-fitting is a serious problem.
Compared to other regularization methods, dropout can avoid over-fitting
more effectively by repeatedly sampling and training the sub-models from
155 the network.

- GFWA [23]: Inspired by the process of generating sparks in fireworks ex-
plosions, the fireworks algorithm [37] is a swarm intelligence optimization
algorithm that can be used to solve optimization problems. The FWA
algorithm has been improved several times since it was proposed. GFWA
160 is a simple and effective version, which is recently proposed. In this paper,

GFWA is used to evolve the parameters of neural networks in the second stage of the framework.

3. Problem Statement

In the multi-target search problem of swarm robotics, multiple targets are randomly scattered in a vast and unknown space. A swarm of robots are delivered into the space, and the robots are supposed to search and collect (or destroy) all the targets as soon as possible. In the simplest case, only three kinds of objects are involved: the environment, the robots, and the targets. In addition, some other constraints may be introduced to the environment, such as obstacles, decoys [47] and inference sources.

3.1. Assumptions

In order to establish the model, some basic assumptions for the problem are made as follows.

- Environment: the space of the environment is very large compared with the size and the perception range of each robot.
- Targets: the targets are static and small in size (similar to robots), with a wide range of influence. The farther the distance from the target, the smaller the influence of the target. The influence of the target remains unchanged in the collecting process, but the influence will disappear instantly once the target is collected. In the overlapping influence areas of multiple target, the influence is determined by the targets that have the greatest one. Multiple targets are randomly and uniformly distributed over the search space.
- Robots: local communication and interaction, limited speed and storage, no prior knowledge of the environment.
- Robot swarm: no global leader, no centralized control, all individuals start from the same region in the environment.

- Fitness value: the degree of influence of targets can be quantified by the fitness value, which is a monotonically decreasing function of the distance to the target. A higher influence is corresponding to a greater fitness value. Robots can perceive the local fitness value within the influence range of the target. Considering the limitation of sensor accuracy and the influence of ambient noise, the robot uses discretized fitness value information to improve the fault tolerance of the system.
- Iteration frequency: the frequency should be high enough to ensure that the difference in fitness values in two consecutive iterations is small enough to avoid missing promising positions. In each iteration, each robot collects its own sensor information, its own historical information, neighbors' individual information, and then makes independent decisions. The iteration frequency is fixed and the robot swarm can work asynchronously.

3.2. An Idealized Model

Based on the above assumptions, an idealized model for the simplest case of the problem is established, as is shown in Fig. 2.

- Environment: a square with a size of $1000 * 1000$.
- Target: a circle with a radius of r_t , in which robots can directly sense the target. The targets are distributed uniformly over the environment space. Each target needs 10 steps to be fully processed, which can be completed by one robot alone in 10 iterations or by 10 robots in 1 iteration, i.e. the collection times of each target are 10.
- Fitness value: the range of influence of the target is a series of annuli with a width of 5 units. The fitness value of each ring decreases linearly from the inside out, from about 40 (center) to 1 (outermost ring). The difference in fitness values of adjacent rings is 1.
- Robots: a square with a size of $1 * 1$. The sensing and communication range is a circular area with a radius of 20 units (i.e. the robot can only

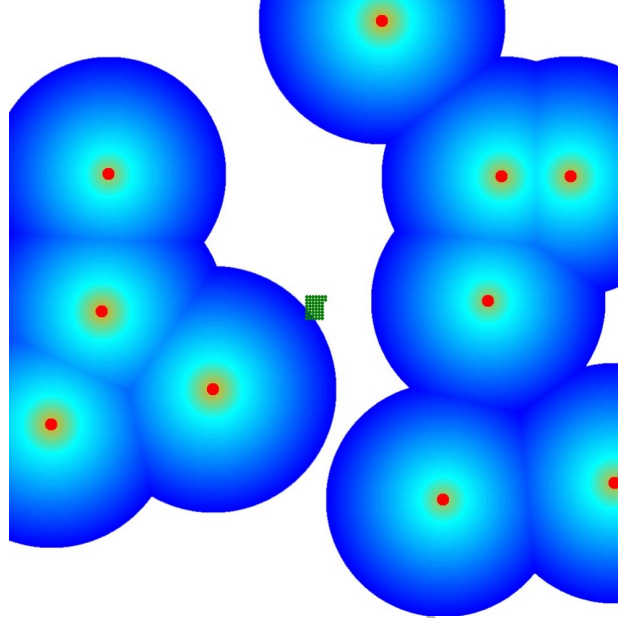


Figure 2: A screenshot of the problem at the beginning of a simulation. Red circles stand for the targets while the central green dot array represents the initial robot swarm. The color of the circles around the target illustrates the fitness value of that position.

communicate with its perceived neighbors). The robot can learn three kinds of information about the neighbors: the location, the fitness value of the location, whether the target is perceived or not. The maximum speed limit for each robot is 5 units (the maximum distance that can be moved in one iteration), and 10 iterations of historical information (positions and fitness values) can be memorized.

- Robot swarm: typical population size setting is 50.

3.3. The Target Strategy

For the multi-target search problem in swarm robotics, a two-stage imitation learning framework is proposed in this paper. The first stage is to use the deep neural network to learn the behaviors of the target strategy. The second stage is to optimize the obtained network. In this article, the target strategy is used

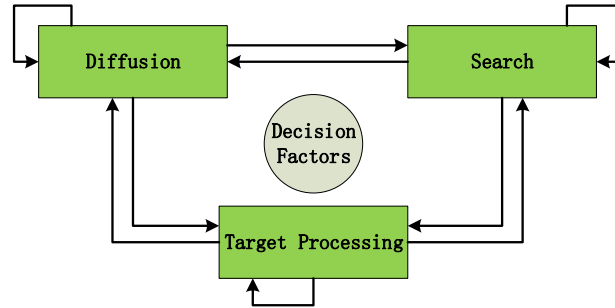


Figure 3: Target strategy PFSMS: A three-state probabilistic finite state machine for multi-target search, including diffusion, search, and target processing. Each state will transfer to other states or maintain the current state with a certain probability. The probability is determined by the decision factors.

to generate the behavioral data needed for deep learning. It should be pointed out that the target strategy itself can be regarded as a black box strategy, that is, we cannot know the specific implementation details of the strategy, but we can observe the behaviors of individual robots adopting the strategy, and thus we can obtain a large amount of behavioral data, usually expressed as “input information-output action” pairs. In order to facilitate intuitive understanding, the basic principles of the target strategy are briefly introduced in this section, but this is not necessary, and only the input and output need to be concerned.

The target strategy is a probabilistic finite state machine based strategy. It can be abbreviated as PFSMS, containing three states: diffusion, search, and target processing, as shown in Figure 3. In this state machine, the robots in each state have a certain probability of transferring to other states or maintaining the current state, and the probability is determined by the decision factors. The decision factors depend on the specific input information, including the robot’s own information and information obtained from neighbors. The robot’s own information includes historical locations and fitness values, fitness value of current location, whether the target is found or not, and the location of the target. The robot can obtain information from neighbors, including the current location and fitness value, whether the target is found or not, and the location

of the target. Based on this information, a 30-dimensional feature is constructed as the network input (as shown in the table 1) and a 2-dimensional action as the network output (as shown in the table 2).

250 In dimension 4 – 6, dimension 4 indicates whether a target is found (1 indicates ‘found’ and 0 indicates ‘not found’), and dimension 5 and dimension 6 indicate the x and y coordinates of the target (x and y are both 0 if no target is found). The process of acquiring target information is as follows. First, if the robot finds the target itself, it will enter the target processing state, and the nearest one will be processed first if multiple targets are found. If the robot 255 does not find the target, it will query the robots in its neighborhood in a certain order (such as from near to far). If a neighbor finds the target, it will stop the query after knowing the target position. If all the robots in the neighborhood do not find the target, then no target is found.

260 In this paper, only one target strategy is considered, although a general imitation framework should be suitable for a variety of strategies, and there are mainly three reasons to do so. Firstly, the initial motivation of this paper is to propose a framework that can imitate the superior strategy which has excellent performance. The PFSMS is chosen as the target strategy for its performance is significantly better than other strategies in this paper. Secondly, the structure 265 of the neural network itself has certain versatility. If a network can imitate an excellent strategy for the task, then the expressive ability of the network structure is guaranteed to some extent, so the neural network structure used in this paper can provide a direct reference for the imitation of other strategies. 270 Thirdly, the specialization work of feature engineering is required for the target strategy. The more consistent the selected network input features are with the target strategy, the more likely it is to learn the behaviors successfully. Data preprocessing and feature selection generally require special study, so it is not proper to imitate various strategies in one article.

275 As is listed in table 1, only two neighbors and two historical records are selected. It is worth noting that in order to avoid feature engineering work, a promising research direction is to use all available input information for end-

Table 1: Descriptions for each dimension of the 30-dimensional input.

Index	Descriptions
0	Current state, three values, -1 for search, 1 for diffusion, 0 for target processing.
1	The number of iterations that the current state has persisted.
2-3	Current velocity, x and y.
4-6	Whether the target is found by the neighbors, and the target's location (x and y).
7-11	The total number of neighbors, and number of neighbors in the up, down, left, and right directions.
12	The number of history records.
13-15	The best position in history and its fitness value.
16-18	The worst position in history and its fitness value.
19	The fitness value of the current position.
20-22	The position of the best neighbor and its fitness value.
23-25	The position of the worst neighbor and its fitness value.
26-29	4 random numbers for probabilistic decision making.

Table 2: Descriptions for each dimension of the 2-dimensional output.

Index	Descriptions
0	The next moving direction (angle value), in the range of $(-\pi, \pi]$
1	The next state.

to-end network training, including all the neighbors and historical information. Obviously, considering all the input information will make the network input wide and affect the simplicity of the network. And in this paper, the cost of simplicity is the need for manually selection and design of the input features, which requires certain domain knowledge. In theory, the closer the selected features are to the information required by the target strategy, the more similar the performance can be imitated.

4. Proposed Methods

4.1. The Motivation

In nature, we can observe the self-organizing behaviors of many biological swarms, such as ant foraging, bees nesting, and bird migration [28] etc., but we do not know the specific strategies followed by biological individuals. Using some equipments and technologies, we can collect and extract behavioral data

of biological swarms, and then we can use some tools to model biological behavior patterns. Imitating the self-organizing behavior of organisms is of great significance for us to reveal the underlying mechanisms and to design efficient swarm strategies.

295 In addition, if the performance metrics of behavior are known, evolutionary algorithms can be used to further optimize the model. For example, in the multi-target search problem, the performance metric is the number of iterations required to find and collect all the targets. Through evolutionary optimization, a more efficient strategy can be obtained, which is very helpful for the practical
300 application of the model.

Based on the above ideas, a two-stage imitation learning framework is proposed in this section. The first stage is to imitate the strategy through deep learning technologies, and the second stage is to evolve the network through some evolutionary algorithm. Apart from the multi-target search, the frame-
305 work can also be used for other swarm tasks.

4.2. The First Stage: Strategy Imitation based on Deep Learning

Training a deep neural network generally requires a large amount of data. In order to learn a good model, the data is usually preprocessed, such as mean subtraction, normalization, batch normalization [48], and so on. The next work
310 is to design the network, including the network type (feedforward, recurrent, convolution, etc.), the number of layers, the activation function, the optimization algorithm, the learning rate, the regularization technique, and the weight initialization technique, etc.

4.2.1. The Data Acquisition and Preprocessing

315 In order to test different strategies, a simulation environment is built on the C# language and the Microsoft XNA framework. In this paper, the data is generated by the interaction between the simulation environment and the target strategy. The simulation environment contains objects such as maps, targets, and robots. The target strategy can access the information of the robots, and

320 then make decisions for the robots. According to the decisions, the environment
 will update the corresponding information (such as the location change, target
 processing, etc.). The 30-dimensional input data includes information that may
 be used by the target strategy, and the target strategy uses only a portion of it
 in an iteration. The output data is the decision of the robot (the next moving
 325 direction and state).

In order to ensure that the data obeys the independent and identical distribu-
 tion, for generating each sample, the seed of the map, the individual number of
 the robot, and the number of iterations are uniformly distributed. The map seed
 value range is $[0, 1)$, and the robot individual number range is $\{0, 1, 2, \dots, 49\}$,
 330 and the iteration number value range is $\{1, 2, 3, \dots, 199\}$.

At first, the collected raw data is used directly for training networks, but
 visual simulations show that the neural network can't learn the robot's search
 behavior well. The search behavior is that the robot will approach the target
 when the individual is within the target influence range (the perceived fitness
 335 value is greater than 0). By analyzing the collected data, we find that the
 samples in the search state ($inputs[0] = -1$) and the fitness value greater than
 0 ($inputs[19] > 0$) occupies only around 1/5. Since this kind of samples are
 more important for improving the robot's search capability, we balance the data
 so that its proportion increases from 1/5 to about 1/2. From the simulation
 340 results, data balance facilitates the neural network learning the search behavior
 of robots.

The balanced data set includes 1.5 million samples, which is divided into
 a training set (900,000), a validation set (300,000), and a test set (300,000)
 by 3:1:1. In addition, considering the inconsistent scales of the input features
 345 in different dimensions, each dimension of the input feature is normalized so
 that the range of each dimension is approximately equal, as is shown in Eq. 2,
 where x^i represents the value of the i th dimension feature, μ^i and s^i represent
 the mean and standard deviation of the dimension feature, both of which are
 calculated from the training set.

$$x^i = \frac{x^i - \mu^i}{s^i} \quad (2)$$

350 4.2.2. The Structure of the Neural Network

The neural network structures used in deep learning can basically be divided into three categories: feed-forward neural networks (FNN), convolutional neural networks (CNN), and recurrent neural networks (RNN). The convolutional neural network is suitable for image processing and the recurrent neural network is suitable for sequence processing. Considering the characteristics and structure of the data set, the FNN is used for deep learning. Next, this section will introduce the specific network configurations, such as the number of network layers and hidden units, activation functions, initialization techniques, regularization techniques, learning algorithms, and learning rates.

360 For the number of network layers and hidden units, a series of simple tests are carried out. Considering performance and model complexity, finally the network structure is setting as 30 – 128 – 256 – 256 – 128 – 2, as is shown in Fig. 4. The network contains 4 hidden layers, and the numbers of hidden units in each layer are 128, 256, 256, and 128, respectively.

365 Two types of activation functions are selected: the ReLU and the SeLU. The former facilitates generating the sparse network, while the latter enhances the normalization of the data stream (zero mean and unit variance). Since both are conducive to the training of deep networks, as a contrasting component, both are involved in the construction of the network. It should be noted that, compared with SeLU, the calculations of ReLU are simpler and easier to be implemented in parallel. If the performance of the two structures are similar, ReLU should be a more appropriate choice. For the settings of SeLU, the configuration in article [20] are adopted: $\lambda \approx 1.0507$ and $\alpha \approx 1.6733$, input normalized to zero mean unit variance (in data preprocessing). The network weights are initialized with variance $1/n$ (n is the number of hidden units in the previous layer). In addition to conventional initialization methods, the unsupervised pre-training with SAE is also investigated for the network. Comparisons of the networks

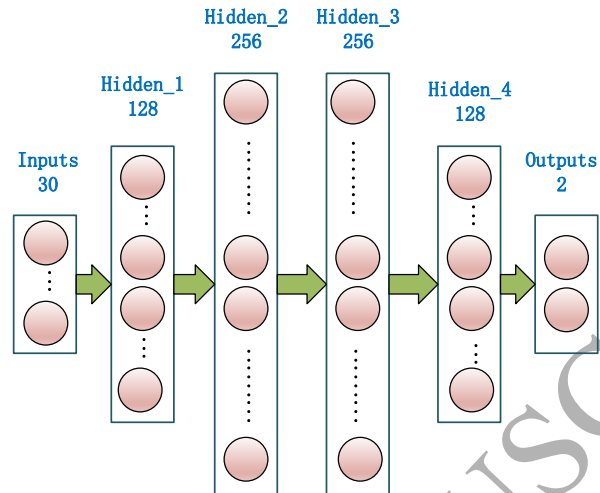


Figure 4: The FNN structure adopted in this paper, containing a 30-dimensional input layer, a 2-dimensional output layer, and 4 hidden layers. The numbers of hidden layer units are 128, 256, 256, and 128, respectively.

with and without SAE were also conducted in experiments.

In order to prevent the network from over-fitting, the network with the
 380 lowest verification error will be stored in time, and the dropout technology is
 also adopted. It needs to be explained that the “alpha-dropout” technology
 designed for SeLU is not used, for it would lead to a larger validation error in
 the experiments. The Adam [49] is used as the optimization algorithm. And
 a dynamic learning rate is adopted, which is updated by the Eq. 3, where
 385 $lr_{\max} = 3e - 3$, $lr_{\min} = 1e - 6$, $decay_factor = 1e5$, and $count$ is the number of
 iterations.

$$lr = lr_{\min} + (lr_{\max} - lr_{\min}) \times e^{-count/decay_factor} \quad (3)$$

4.2.3. Experimental Results of Four Types of Networks

The basic structure of the neural network is shown in Fig. 4. Based on
 the type of activation function (ReLU or SeLU) and whether the SAE is used,
 390 four types of networks are selected for training and simulation: *SeLU_SAE*,

SeLU_NoSAE, *ReLU_SAE*, *ReLU_NoSAE*. During the training, *batch_size* is set to 500 and the maximum number of iterations is $1e6$ (one iteration for each batch). All four types of networks use the dropout technique, and the keep probability of SeLU is set to 0.97, while the keep probability of ReLU is set to
 395 0.98.

Firstly, each type of network is trained 10 times and 10 networks are obtained. The average training error, average verification error, and average test error for 10 networks are expressed as *train_error*, *vali_error*, and *test_error* respectively. In the simulation, the trained network is used for multi-target
 400 search experiments, and the performance metric is the “number of iterations” required to find and collected all the targets. There are 50 robots and 10 targets in each experiment. For each type of network, each trained network run 3,000 times, and the mean and standard deviation of the “number of iterations” for a total of 30000 experiments over 10 networks are expressed as *mI* and *dI*, re-
 405 spectively. Finally, experimental results of four types of networks are obtained, as is shown in Tab. 3.

As shown in Tab. 3, the *vali_error* and the *test_error* of each type of network are similar, indicating that the trained networks bear good generalization. Comparing *SeLU_SAE* and *SeLU_NoSAE*, using SAE will reduce the *vali_error*,
 410 but the simulation performance will be worse (*mI* is larger), which is similar for *ReLU_SAE* and *ReLU_NoSAE*. On the one hand it illustrates the SAE facilitating deep network training. On the other hand, it shows that there is a certain deviation between the behavioral data used for training network and the target strategy for generating data. The deviation may result from the incon-
 415 sistency of the raw data and the preprocessed data, for the latter is balanced to improve the robot’s search ability.

As can be seen from table 3, the *vali_error* of *ReLU_SAE* is smaller than that of *ReLU_NoSAE*, but the *train_error* of *ReLU_SAE* is larger, which indicates that the over-fitting problem of the network without SAE is more
 420 serious. A small *vali_error* means that the network has better generalization performance. In *ReLU_SAE*, an SAE is used for pre-training, so the initial

Table 3: The *train_error*, *vali_error*, *test_error*, *mI*, *dI* of four types of networks.

	<i>train_error</i>	<i>vali_error</i>	<i>test_error</i>	<i>mI</i>	<i>dI</i>
<i>SeLU_SAE</i>	0.23	0.37	0.37	166.51	29.44
<i>SeLU_NoSAE</i>	0.23	0.38	0.37	163.24	27.36
<i>ReLU_SAE</i>	0.25	0.39	0.39	182.70	46.30
<i>ReLU_NoSAE</i>	0.21	0.42	0.42	164.22	28.78

weight of the network learns some overall structure of the data set, which may, to some extent, weaken the over-fitting effect resulting from local optima, although this is not obvious for *SeLU_SAE* and *SeLU_NoSAE*.

425 One of the four types of networks is supposed to be selected for the second stage. There are mainly two points to be considered: First, the simulation performance of the network should be good, so the SAE will not be adopted; second, the calculation should be simple and easy to be implemented in parallel. Since the simulation performances of *SeLU_NoSAE* and *ReLU_NoSAE* are similar
430 and the calculation of ReLU unit is much simpler, finally the *ReLU_NoSAE* is chosen. From 10 networks of *ReLU_NoSAE* type, the one with the best simulation performance ($mI \approx 159$) is selected for the evolutionary optimization in the second stage.

4.3. The Second Stage: Strategy Optimization based on Evolutionary Computation

435 *tion*

Through the first stage of deep learning, a strategy is obtained (a network of *ReLU_NoSAE* type). In order to further fine tune the strategy, evolutionary algorithms are adopted in the second stage to improve the network performance. Since the combination of deep networks and evolutionary algorithms is important,
440 it's necessary to analyze and solve the problems caused by the attributes of these two parts.

4.3.1. Reduce the Dimensions to be Optimized

Deep neural networks have strong abilities of representation, but the parameters of the network will increase significantly with the increase of network depth. For example, the number of weights of the FNN in Fig. 4 is greater than $1e5$. The resulting network is used as a initial solution, which will be optimized locally by evolutionary algorithms. However, generally the problem dimension for existing evolutionary algorithms is limited in application, otherwise the efficiency and performance will be reduced. Even if the large-scale global optimization problem is only about 1000-dimensional [50], and usually the optimization dimension is not more than 100 dimensions [51]. Some evolutionary algorithms such as CMA-ES [52] involve nonlinear operations such as matrix decomposition, so they are not suitable for problems with too high dimensions.

Therefore, the first problem to be solved is how to reduce the dimensions to be optimized, and we can choose a part of dimensions for optimization, such as optimizing only the offset of the network or optimizing only one layer of the network. For FNN networks, the previous layer affects the later layers and it is generally difficult to be locally optimized. Considering that the last layer has a direct and critical impact on the output, we only optimize the last layer of the network, similar to the mechanism of the extreme learning machine [53]. In this way, the number of dimensions to be optimized (connection weights + offsets) is $128 * 2 + 2 = 258$, which accounts for approximately 0.2% of the original dimension, thus reducing greatly the number of dimensions that need to be optimized.

4.3.2. The Setting of the Evolutionary Algorithm

Firstly, the operation of evolutionary algorithms requires a good performance metric, which is supposed to be relatively stable and suitable for the task. For the multi-target search problem in swarm robotics, the performance metric is the number of iterations required for the swarm to find and collect all the targets. But it should be noted that the task involves many random factors:

the targets on the map are randomly generated, and the robot search strategy also has a certain randomness. In order to reduce the impact of the randomness, the average performance in many runs is adopted as the performance metric.

475 Many tests on the network show that the standard deviation σ of the number of iterations required for the swarm is less than 30 ($\sigma < 30$). According to the central limit theorem, if the mean number of iterations over 1000 runs is the performance metric, the corresponding standard deviation $\sigma' = \frac{\sigma}{\sqrt{1000}} < 1$. In the experiments, 40 maps are generated, and the network runs 25 times in each map, and finally the mean number of iterations over 1000 runs is chosen as the performance metric, which is relatively stable.

On the other hand, compared with gradient descent methods, the speed of progress of evolutionary algorithms is limited. Generally, many evaluations are needed in order to achieve good results. Therefore, the time-consuming evaluation has a large impact on the algorithm [54]. Since each evaluation in the evolutionary algorithm requires 1000 runs, the evaluation of this task is time-consuming. In order to improve the efficiency of the algorithm, it is a feasible way to enhance the parallelism. Specifically, multiple threads can be used to evaluate candidate solutions (candidate networks) in parallel. And four threads are used in this article, significantly improving the efficiency of the algorithm.

490 Finally, because the evolutionary algorithm is used to fine-tune the network obtained in the first stage, and the number of evaluations should not be too large (the evaluation is time-consuming), the learning rate of the algorithm is supposed to be small. In GFWA, the learning rate corresponds to the explosion amplitudes of fireworks, which is set to $3e-3$ initially in this article.

4.3.3. Optimizing the Neural Network using GFWA

With the above settings, the GFWA is used to optimize the *ReLU_NoSAE* network obtained in the first stage. For the parameters of GFWA, the number of firework is 1 and the number of sparks is 11. The convergence curve of optimizing neural networks using GFWA is shown in Fig. 5. The abscissa indicates the number of evaluations in GFWA, and the ordinate mI indicates the average

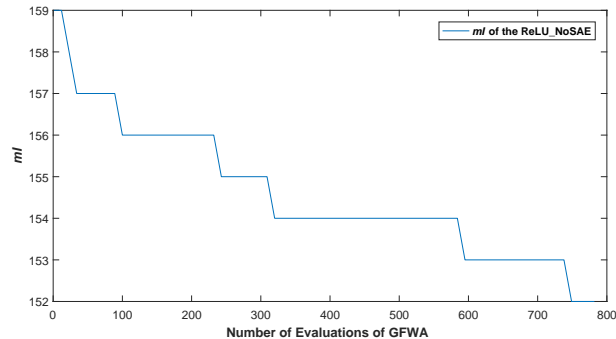


Figure 5: The convergence curve of optimizing neural networks using GFWA.

number of iterations required for the network to collect all the targets. As is shown in Fig. 5, through the optimization of GFWA, the performance of *ReLU_NoSAE* is improved from 159 to 152. However, the evaluation still has a certain randomness, and the actual simulation performance is about 155. For the convenience of distinction, the *ReLU_NoSAE* obtained in the first stage is named as *RNS*, and the evolved *ReLU_NoSAE* in the second stage is named as *RNSE* (*ReLU_NoSAE_Evolution*).

Compared to the first stage of deep learning, the second stage of optimization is limited in performance improvement, which is around 2.5% (from 159 to 155). The optimization is only used to locally fine-tune the network obtained from the first stage, and the limited improvement is also useful in some situations. For example, if the overall energy consumption of the robotic swarm is high, then 2.5% improvement can also save a lot of energy. This improvement is also statistically significant, and two-side Wilcoxon rank sum tests (with confidence level 95%) are conducted in the following section to support the conclusion.

5. Simulation Results and Discussions

In this section, a series of comparative experiments are conducted under different problem settings. In each experiment, each strategy runs 1,000 times on 40 random maps (25 runs in each map), and the final result is the average

performance of 1,000 runs. The performance metrics for strategies are the mean and standard deviation of the number of iterations required by the swarm to collect all the targets in 1,000 runs, which are expressed as mI and dI , respectively. The mI shows the search efficiency of the strategy, and the dI reflects the stability of the strategy. In addition, another two indicators are introduced to measure the parallel processing capability and the collaborative processing capability of search strategies. So there are four main indicators, as shown below:

- Efficiency: the mean of the number of iterations required to collect all the targets, represented by mI .
- Stability: the standard deviation of the number of iterations required to collect all the targets, represented by dI .
- Parallel processing capability: with fixed number of robots, the average number of extra iterations (extra mI) needed for one additional target. The smaller the number of extra iterations, the stronger the parallel processing capability. This indicator shows the ability of a search strategy to process multiple targets in parallel.
- Collaborative processing capability: with fixed number of robots, the average number of extra iterations (extra mI) needed for one additional collection time of targets. Normally, the collection times of each target are 10. The smaller the number of extra iterations, the stronger the collaborative processing capability. This indicator shows the ability of a search strategy to process one target collaboratively.

5.1. Algorithms for Comparison

All the parameters of the comparison algorithms are tuned under the same experimental settings. The map size is 1000*1000. The population size is 50, and the number of targets is 10. All algorithms and their parameters are as follows:

- RPSO: Robotic Particle Swarm Optimization, inertia weight $w = 3.0$,
 social coefficient $c_2 = 2.0$, cognition coefficient $c_1 = 1.0$, random coefficient
 $c_3 = 0.1$.
- A-RPSO: Adaptive Robotic Particle Swarm Optimization, inertia weight
 $w_{ini} = 1.0$, social coefficient $c_2 = 0.4$, cognition coefficient $c_1 = 0.6$, scale
 factor $\alpha = 0.3$, scale factor $\beta = 0.7$, random coefficient $c_3 = 0.1$.
- IGES: Improved Group Explosion Strategy, group size threshold $size = 5$.
 Since the performance of IGES is much better than GES, the latter is not
 used for comparison.
- TFS: Triangle Formation Search, the parameter configuration is the same
 as that in [32].
- PFSMS: PFSM-based Search, this strategy is the target strategy to be
 imitated. Although we do not care about the details of the strategy, the
 parameters are also given: initial probability for holding state $P_{ini} =$
 0.9997 , inertia weight $w = 0.55$, diffusion threshold $T_b = 2.3$.
- RNS: *ReLU_NoSAE*, the strategy is obtained from the deep learning in
 the first stage, and there is no parameters to be optimized other than the
 weights and offsets of the neural network.
- RNSE: *ReLU_NoSAE_Evolution*, the strategy is obtained from evolving
RNS in the second stage, and there is no parameters to be optimized
 other than the weights and offsets of the neural network.

Although only PFSMS is used for imitation, the performance of other com-
 parison strategies is also provided to enhance the intuitiveness of the conclusions.
 The initial motivation for this paper is to propose a framework that can imitate
 superior strategies. With the performance evaluation of other search strategies,
 the superiority of PFSMS can be highlighted, and the similarity in performance
 between the PFSMS and the strategies from imitation is also emphasized visu-

Table 4: mI and dI of search strategies with various population size and 10 targets.

Population	RPSO		A-RPSO		IGES		TFS		PFSMS		RNS		RNSE	
	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI
25	347.4	88.0	338.6	79.3	286.7	61.2	316.7	121.0	188.4	40.2	204.4	52.6	197.5	45.2
50	295.4	76.6	280.5	69.7	229.7	39.4	212.3	59.0	148.4	20.5	157.3	25.0	154.6	22.3
75	260.0	68.9	254.5	62.7	212.6	33.6	180.9	39.4	137.9	18.2	148.0	20.4	143.3	18.8
100	245.0	63.6	238.2	60.1	203.3	31.4	164.4	28.5	132.6	16.0	141.8	19.1	138.7	17.2
125	229.1	57.1	222.1	50.7	194.8	28.2	154.3	22.8	129.0	15.0	137.0	17.3	133.8	15.5
150	219.5	53.3	215.3	48.6	190.3	27.2	149.4	22.0	127.1	15.0	134.9	16.5	131.7	15.4
175	214.7	52.8	209.6	47.3	186.4	26.1	143.7	19.9	125.2	14.5	132.3	15.0	129.6	14.5
200	207.1	50.5	202.2	43.3	182.5	24.7	141.0	18.4	123.8	14.7	131.4	15.0	128.1	13.8

ally (ie, the performance curves of RNS and RNSE are close to that of PFSMS but significantly better than those of other strategies).

5.2. Different Population Sizes

In this section, the search efficiency of various comparison algorithms under different population sizes is investigated. In experiments, each strategy is tested at the population size of 25, 50, 75, 100, 125, 150, 175, 200 in turn, and the map size is 1000*1000, containing 10 targets. The experimental results are shown in Tab. 4, Fig. 6 and Fig. 7.

The Significance of RNS and RNSE: As is shown in Tab. 4 and Fig. 6, the mI s of RNS and RNSE are close to that of PFSMS, and the performance of RNSE is slightly better. In addition, under different population sizes, two-side Wilcoxon rank sum tests (with confidence level 95%) are conducted on the RNS and other algorithms, as well as on the RNSE and other algorithms, which are based on the data of 1000 runs. And the results of statistical test prove that RNS is significantly better than other strategies except RNSE and PFSMS, while RNSE is significantly better than other strategies except PFSMS.

As is shown in the results, although a random component is introduced to prevent robots from falling into local oscillation, the search efficiency of RPSO is still the lowest. Despite the introduction of a simple niche technology (limiting the communication range of robots), A-RPSO also performs poorly. By limiting the search space into 2D or 3D, some strategies can be adapted from high-

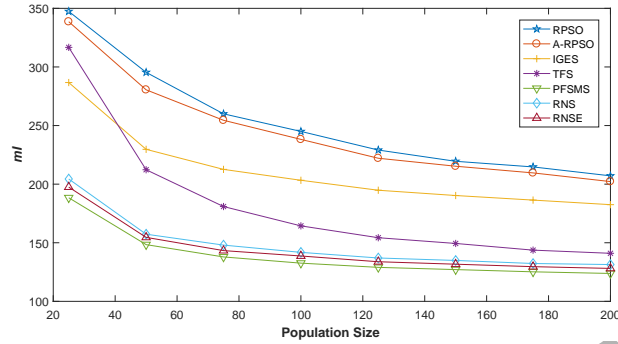


Figure 6: mI of search strategies with various population sizes.

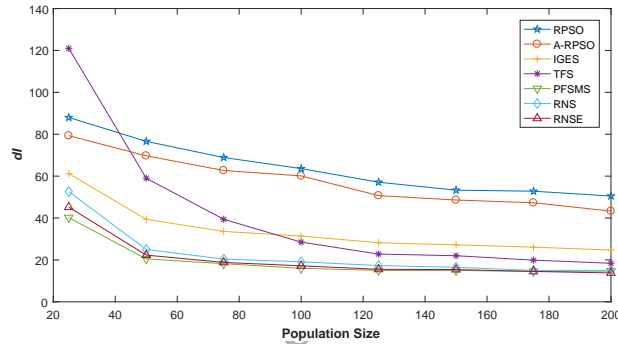


Figure 7: dI of search strategies with various population sizes.

dimensional algorithms (such as PSO). But the adapted strategies may not be suitable for the multi-target search task in swarm robotics, which focuses on 2D or 3D scenarios. Compared with RPSO and A-RPSO, the performance of IGES has significant advantages. This is mainly because IGES pays attention to the diffusion mechanism, which improves the swarm's exploration ability. For TFS, the population size is a particularly important influence factor. As the population size increases, the performance of TFS increases significantly.

As is shown in Fig. 6, except for RNS and RNSE, the performance of the target strategy (PFSMS) is significantly better than other strategies, showing the highest search efficiency. With a population size of 50, compared to the other six strategies (RPSO, A-RPSO, IGES, TFS, RNS, and RNSE), the efficiency of

PFSMS increased by 50%, 47%, 35% , 30%, 6%, and 4% respectively. It can be seen that after the first stage of deep learning, the search efficiency of RNS is very close to the target strategy, and after the second stage of evolution, the efficiency of RNSE is closer to the PFSMS. Except for the PFSMS, the performance of RNS and RNSE is significantly better other comparison algorithms, which verifies the effectiveness of the proposed two-stage imitation learning framework.

Comparing Fig. 6 and Fig. 7, the trends of the curves are roughly the same, which means the more efficient strategy is generally more stable. After two stages of learning and evolution, the stability of the RNSE is very close to the target strategy (PFSMS), even beyond the target strategy with a large population (such as 200 robots). The stability of the RNSE originates from the imitation of the diffusion mechanism of the target strategy. Therefore, the difference in efficiency between RNSE and the target strategy is mainly due to the difference in local exploitation capacity, and further enlarging the proportion of relevant samples in data set may improve the performance to some extent. When the population size is 50, according to stability, the search strategies can be sorted as $PFSMS \approx RNSE \approx RNS > IGES > TFS > A - RPSO > RPSO$.

As is shown in the experimental results, under different population size, RNS and RNSE are very close to the target strategy in efficiency and stability, and the performance of RNSE is slightly better than RNS.

Experiments in the Environment with Small Obstacles: There is certain independence between the search strategy and the obstacle avoidance strategy, and simple obstacle avoidance approaches can be applied to search strategies. Thus, it is reasonable to focus on improving the efficiency of the search strategy in environments without obstacles. Still, a problem that cannot be ignored is that the obstacle avoidance strategy may have an impact on the performance of the search strategy. In order to show the rationality of the study without obstacles, experiments in the environment with small obstacles were carried out in this section.

As is shown in Fig. 8, the environment contains 100 small obstacle, which

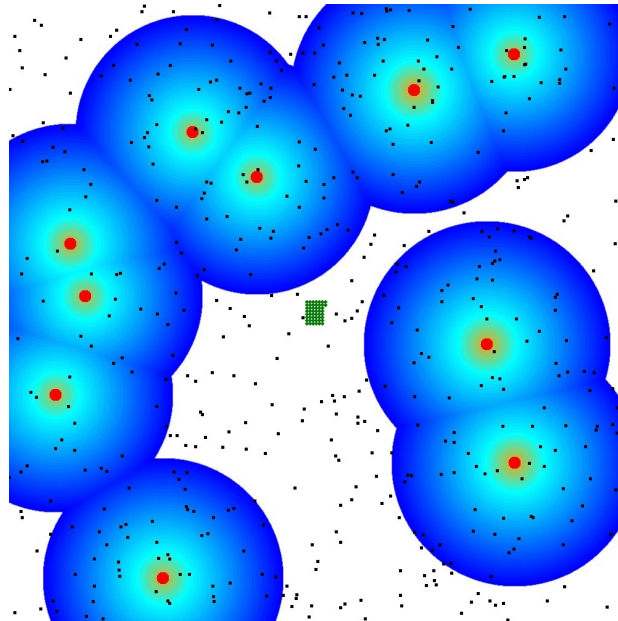


Figure 8: A screenshot of the problem with obstacles at the beginning of a simulation. Black squares stand for the obstacles. Red circles stand for the targets while the central green dot array represents the initial robot swarm.

are represented by black squares (each with a size of $1 * 1$). In this paper, a
 640 simple avoiding strategy is applied for search strategies. In this strategy [11],
 the robot will check if it will run into obstacles with current velocity. If so,
 a small repulsive force perpendicular to the velocity from the obstacle will be
 added to avoid the collision.

As is shown in Fig. 9, the mI of search strategies in the environment with
 645 small obstacles are similar to that in Fig. 6. Although the mI in obstructive
 environments is a little higher, the overall trend of curves and relative rank-
 ings of different search strategies remain unchanged. Therefore, small obstacles
 basically do not affect the relative performance of search strategies.

5.3. Different Number of Targets

650 In this section, the search efficiency of various comparison algorithms under
 different number of targets is investigated. In experiments, each strategy is

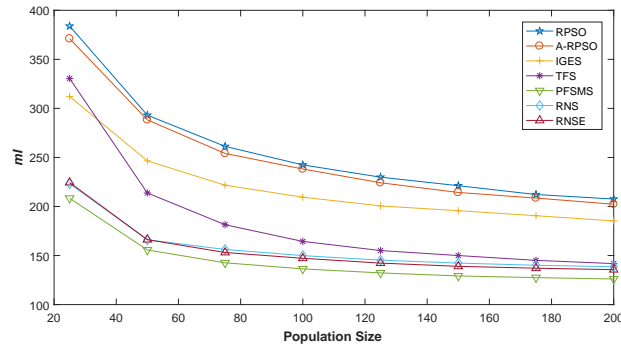


Figure 9: mI of search strategies with various population sizes in the environment with small obstacles.

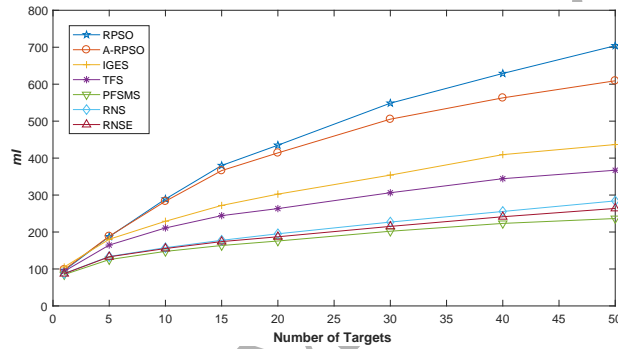
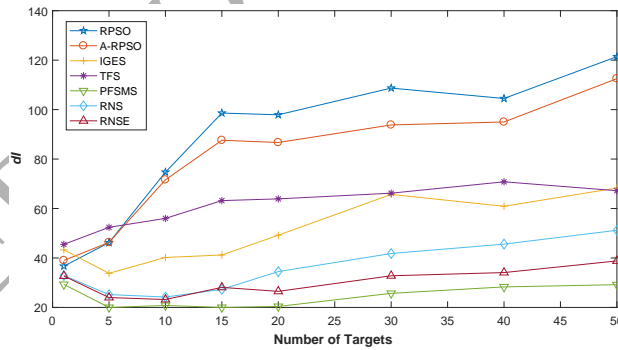
tested at the target number of 1, 5, 10, 15, 20, 30, 40, 50 in turn, and the map size is 1000*1000, containing 50 robots. The experimental results are shown in Tab. 5, Fig. 10 and Fig. 11.

655 **The Significance of RNS and RNSE:** As is shown in Tab 5 and Fig. 10, the mI s of RNS and RNSE are close to that of PFSMS, and the performance of RNSE is slightly better. In addition, under different target numbers, two-side Wilcoxon rank sum tests (with confidence level 95%) are conducted on the RNS and other algorithms, as well as on the RNSE and other algorithms, which are
 660 based on the data of 1000 runs. Statistical tests show that there is no significant difference between TFS, RNS, and RNSE when the target number is 1. And there is no significant difference between RNS and RNSE when the target number is 5. In other cases, RNS is significantly better than other algorithms except for PFSMS and RNSE, and RNSE is significantly better than other algorithms
 665 except PFSMS. With the increase in the number of targets, the advantages of RNS and RNSE gradually emerge, showing excellent parallel search capabilities.

As is shown in Tab. 5 and Fig. 10, when there is only one target, all search strategies have similar performance, and there is no statistically significant difference in the performance of TFS, RNS, and RNSE. The excellent local exploitation capability of TFS makes up for its lack of global exploration capabilities.
 670 From the results, the performance of RNS and RNSE is especially close,

Table 5: mI and dI of search strategies with various numbers of targets and 50 robots.

Targets	RPSO		A-RPSO		IGES		TFS		PFSMS		RNS		RNSE	
	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI
1	96.0	36.7	98.2	39.0	105.3	43.3	93.8	45.5	84.8	29.3	87.6	32.9	87.3	32.7
5	187.4	46.2	188.5	46.3	180.4	33.8	164.7	52.4	125.4	20.0	133.7	25.2	132.9	24.0
10	289.2	74.7	283.3	71.6	229.1	40.2	211.0	56.0	147.6	20.8	157.6	24.2	155.4	23.2
15	380.0	98.6	366.2	87.6	271.9	41.2	244.3	63.2	163.5	20.0	177.6	27.3	174.1	28.1
20	434.9	97.9	414.1	86.7	302.5	49.2	263.4	63.9	175.7	20.4	195.2	34.5	187.1	26.5
30	548.6	108.7	505.3	93.8	354.0	65.7	306.3	66.2	202.1	25.7	226.9	41.8	215.2	32.8
40	629.0	104.5	563.0	95.0	409.4	60.9	344.3	70.8	223.3	28.3	255.6	45.6	241.3	34.1
50	704.1	121.4	609.2	112.5	436.7	68.3	367.2	67.2	236.5	29.2	284.2	51.2	263.8	38.8

Figure 10: mI of search strategies with various number of targets.Figure 11: dI of search strategies with various number of targets.

which means that the two strategies have similar local exploitation capabilities, and the difference in performance lies mainly in the parallel search of multiple targets.

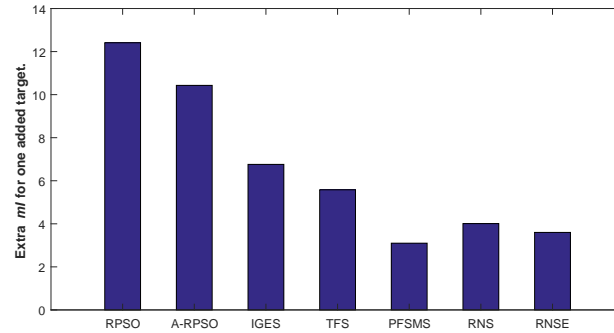


Figure 12: Extra mI required for one added target of different search strategies.

As is shown in Fig. 10, the slope of each curve represents the number of
 675 extra iterations needed for one additional target, so a small slope means the
 algorithm has better parallelism. For each strategy (RPSO, A-RPSO, IGES,
 TFS, PFSMS, RNS, and RNSE), the slopes of the curves are 12.41, 10.43, 6.76,
 5.58, 3.10, 4.01, and 3.60, respectively, as is shown in Fig. 12. RNS and RNSE
 680 have similar parallelism with PFSMS, while RNSE is slightly better than RNS.
 Considering that RNS and RNSE have similar local exploitation capabilities,
 the evolutionary stage mainly enhances the parallel search capability of the
 strategy. According to parallelism, the strategies can be sorted as $PFSMS >$
 $RNSE > RNS > TFS > IGES > A - RPSO > RPSO$.

With different number of targets, the stability of each search strategy is
 685 shown in Fig. 11. As the number of targets increases, the curves of IGES, RNS,
 RNSE, and PFSMS drop first and then increase, which means that a proper
 increase in targets helps to reduce the randomness of the problem. As can be
 seen from the curve trend, compared with other algorithms, the stability of
 690 TFS, RNSE and PFSMS is less sensitive to the number of targets. On the other
 hand, as the number of targets increases, the stability of PRSO and A-RPSO
 significantly deteriorates, indicating that they are not very good at parallel
 search for multiple targets.

As is shown in the experimental results, with different number of targets,
 695 RNS and RNSE are very close to the target strategy in efficiency and stability,

and the performance of RNSE is slightly better than RNS.

5.4. Different Collection Times of Targets

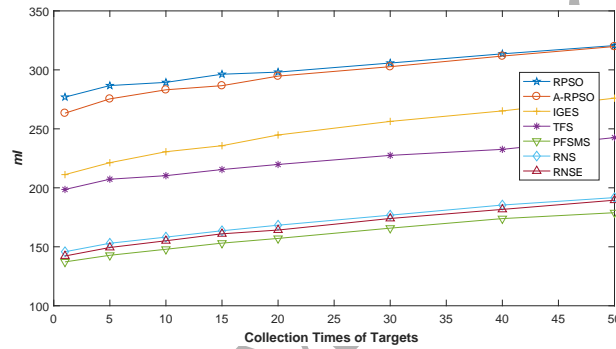
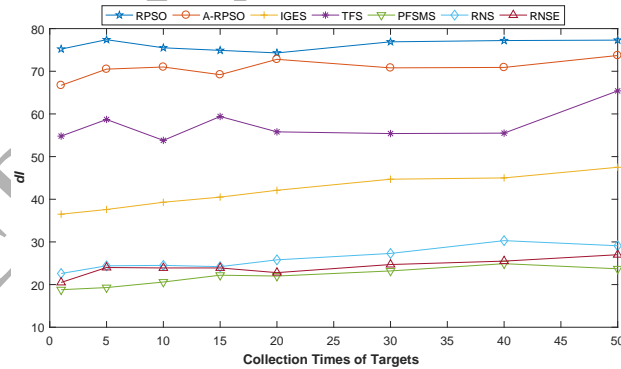
As described in the problem statement, a target requires 10 steps to be processed, which can be done by a single robot in 10 iterations, or 10 robots in one iteration. The “collection times of targets” refers to the number of iterations that a robot needs to collect a target. If the “collection times of targets” is 50, then it takes a robot 50 iterations to collect one target. With different collection times of targets, collaborative collection capabilities of the swarm in different strategies can be investigated, which is complementary to the parallel search capabilities of the swarm. In this section, the search efficiency of various comparison algorithms under different collection times of targets is investigated. In experiments, each strategy is tested at the collection time of 1, 5, 10, 15, 20, 30, 40, 50 in turn, and the map size is 1000*1000, containing 10 targets and 50 robots. The experimental results are shown in Tab. 6, Fig. 13 and Fig. 14.

The Significance of RNS and RNSE: As is shown in Tab 6 and Fig. 13, the mI s of RNS and RNSE are close to that of PFSMS, and the performance of RNSE is slightly better. In addition, under different collection times of targets, two-side Wilcoxon rank sum tests (with confidence level 95%) are conducted on the RNS and other algorithms, as well as on the RNSE and other algorithms, which are based on the data of 1000 runs. Statistical tests show that there is no significant difference between RNS and RNSE when the collection times of targets are 30, 40 and 50. In other cases, RNS is significantly better than other algorithms except for PFSMS and RNSE, and RNSE is significantly better than other algorithms except PFSMS.

As shown by Fig. 13, as the collection time of targets increases, the mI for different search strategies grows approximately linearly. With different collection times of targets, the relative ranking of the efficiency of each strategy has not changed, and the performance of RPSO gradually approaches A-RPSO, indicating that the former has a stronger collaborative collection capability.

Table 6: mI and dI of search strategies with various collection times of targets.

Collection Times	RPSO		A-RPSO		IGES		TFS		PFSMS		RNS		RNSE	
	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI
1	276.9	75.2	263.3	66.7	211.2	36.5	198.6	54.8	137.2	18.8	145.8	22.6	142.2	20.5
5	286.7	77.4	275.4	70.5	221.3	37.6	207.3	58.7	142.8	19.3	153.0	24.4	149.4	24.0
10	289.4	75.5	283.1	71.0	230.6	39.3	210.3	53.8	147.9	20.6	158.2	24.5	155.1	23.9
15	296.3	74.9	286.6	69.2	235.7	40.5	215.5	59.4	153.1	22.2	163.6	24.2	161.0	23.9
20	298.1	74.3	294.7	72.8	244.8	42.1	219.8	55.8	157.1	22.0	168.3	25.8	164.2	22.8
30	305.8	76.9	302.7	70.8	256.3	44.7	227.5	55.4	165.8	23.2	176.8	27.3	174.0	24.7
40	313.6	77.2	311.7	70.9	265.2	45.0	232.6	55.5	173.9	24.9	185.4	30.3	181.7	25.5
50	320.6	77.3	319.8	73.7	276.0	47.5	242.6	65.4	178.9	23.7	191.7	29.1	189.5	27.0

Figure 13: mI of search strategies with various collection times of targets.Figure 14: dI of search strategies with various collection times of targets.

The slope of each curve in the figure indicates the number of extra iterations needed for one additional collection time of targets, indicating the collaborative

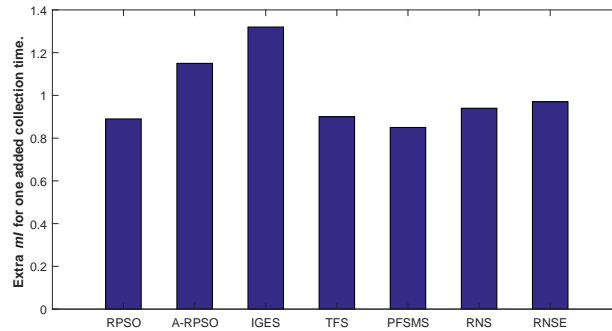


Figure 15: Extra mI required for one added collection time of different search strategies.

collection ability of the swarm, thus a small slope means that the search strategy has better collaborative processing capabilities. For each strategy (RPSO, A-RPSO, IGES, TFS, PFSMS, RNS, and RNSE), the slope of the curve is 0.89, 1.15, 1.32, 0.90, 0.85, 0.94, 0.97, as is shown in Fig. 15. According to the collaborative processing capability, the strategies can be sorted as: $PFSMS > RPSO \approx TFS > RNS > RNSE > A-RPSO > IGES$. Compared with RPSO, niche technology limits the swarm collaboration of A-RPSO and reduces the ability of collaborative processing. In TFS, three-robot formations allow a single target to be processed by at least three robots, resulting in good collaborative collection capabilities. IGES tends to allow the group to spread out slowly, weakening the collaboration among individuals. Compared with the target strategy PFSMS, the RNS and RNSE have a weaker ability for collaborative processing, and the RNSE is slightly weaker than RNS, which means that the evolution of the second stage slightly sacrificed the swarm's collaborative processing capability to enhance parallel search capabilities.

As is shown in Fig. 14, the stability of each strategy is less sensitive to the collection times of targets. The stability of RNS and RNSE is close to PFSMS, which is significantly better than other comparison algorithms. Because this article focuses on the search efficiency of the strategy, rather than the collaborative processing, the collection times of targets in the problem is small (10 times), and only linear acceleration is considered (the collection efficiency of the

target is proportional to the number of participating robots), so the performance
750 improvement brought by good collaborative collection capability is not significant.
If the collection times of targets increases significantly, the performance
of various strategies may be different.

As is shown in the experimental results, with different collection times of
targets, RNS and RNSE are very close to the target strategy in efficiency and
755 stability, and the performance of RNSE is slightly better than RNS.

6. Conclusions

The existing multi-target search strategies in swarm robotics are generally
behavior-based, that is, the multi-target search behavior of the swarm is
achieved by designing individual behaviors. The purpose of this article is not to
760 design new strategies, but to approach a target strategy through imitation and
evolution. In this paper, it is assumed that the specific details of the target strategy
are unknown, but the individual robot's behavioral data and performance
metric can be obtained. This paper proposes a two-stage imitation learning
framework to approach the target strategy. In the first stage, the neural network
765 is used to learn the behavioral data of individual robots. In the second
stage, the network parameters are optimized by evolutionary algorithms. Experimental
results show that the final strategy RNSE is very close to the target strategy
PFSMS in terms of efficiency and stability, showing a balance between
exploration and exploitation, and a good trade-off between parallelism and
770 collaboration, which validates the effectiveness of the framework. The method
section in this article is useful for the design of the network structure (layers,
activation functions, initialization techniques, regularization techniques, etc.)
and the setting of evolutionary algorithms (stability of the performance metric,
learning rate settings) to facilitate the framework promotion and migration
775 applications.

In this paper, the evolutionary algorithm is only used to optimize the last
layer of the network, and the study of evolving the entire network can be con-

ducted in the future. In addition, there are certain artificial traces in the construction and selection of the network input features in this paper. Since deep
780 networks have strong feature extraction and representation capabilities, in the future research, the original data can be directly used as the input of the network for training. In addition to the multi-target search, the framework can also be used for other tasks in swarm robotics, such as aggregation, dispersion, object transport, and so on.

785 **Acknowledgment**

This work was supported by the Natural Science Foundation of China (NSFC) under grant no. 61375119 and 61673025 and also Supported by Beijing Natural Science Foundation (4162029), and partially supported by National Key Basic Research Development Plan (973 Plan) Project of China under grant no.
790 2015CB352302.

References

- [1] S. Camazine, *Self-organization in biological systems*, Princeton University Press, 2003.
- [2] E. Şahin, *Swarm robotics: From sources of inspiration to domains of application*, in: *International workshop on swarm robotics*, Springer, 2004, pp. 10–20.
795
- [3] E. Şahin, S. Girgin, L. Bayindir, A. E. Turgut, *Swarm robotics*, in: *Swarm intelligence*, Springer, 2008, pp. 87–100.
- [4] M. S. Couceiro, D. Portugal, R. P. Rocha, *A collective robotic architecture in search and rescue scenarios*, in: *Proceedings of the 28th Annual ACM Symposium on Applied Computing*, ACM, 2013, pp. 64–69.
800
- [5] J. Clark, R. Fierro, *Mobile robotic sensors for perimeter detection and tracking*, *ISA transactions* 46 (1) (2007) 3–13.

- [6] J. Alonso-Mora, S. H. Lohaus, P. Leemann, R. Siegwart, P. Beardsley,
805 Gesture based human-multi-robot swarm interaction and its application to
an interactive display, in: Robotics and Automation (ICRA), 2015 IEEE
International Conference on, IEEE, 2015, pp. 5948–5953.
- [7] G. A. Landis, Robots and humans: synergy in planetary exploration, *Acta
aeronautica* 55 (12) (2004) 985–990.
- [8] W. Wu, A. Song, P. Varnell, F. Zhang, Cooperatively mapping of the un-
810 derwater acoustic channel by robot swarms, in: Proceedings of the Interna-
tional Conference on Underwater Networks & Systems, ACM, 2014,
p. 20.
- [9] B. Yang, Y. Ding, Y. Jin, K. Hao, Self-organized swarm robot for target
815 search and trapping inspired by bacterial chemotaxis, *Robotics and Au-
tonomous Systems* 72 (2015) 83–92.
- [10] A. Jaimes, S. Kota, J. Gomez, An approach to surveillance an area using
swarm of fixed wing and quad-rotor unmanned aerial vehicles uav (s), in:
System of Systems Engineering, 2008. SoSE'08. IEEE International Con-
820 ference on, IEEE, 2008, pp. 1–6.
- [11] J. Li, Y. Tan, The multi-target search problem with environmental restric-
tions in swarm robotics, in: Robotics and Biomimetics (ROBIO), 2014
IEEE International Conference on, IEEE, 2014, pp. 2685–2690.
- [12] L. Bayindir, E. Şahin, A review of studies in swarm robotics, *Turkish Jour-
825 nal of Electrical Engineering & Computer Sciences* 15 (2) (2007) 115–
147.
- [13] M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: a
review from the swarm engineering perspective, *Swarm Intelligence* 7 (1)
(2013) 1–41.

- 830 [14] O. Soysal, et al., Probabilistic aggregation strategies in swarm robotic systems, in: Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE, IEEE, 2005, pp. 325–332.
- [15] O. Khatib, Real-time obstacle avoidance for manipulators and mobile robots, *The international journal of robotics research* 5 (1) (1986) 90–98.
- 835 [16] R. S. Sutton, A. G. Barto, Reinforcement learning: An introduction, Vol. 1, MIT press Cambridge, 1998.
- [17] S. Nolfi, D. Floreano, *Evolutionary robotics* (2000).
- [18] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al.,
840 Mastering the game of go with deep neural networks and tree search, *nature* 529 (7587) (2016) 484–489.
- [19] X. Glorot, A. Bordes, Y. Bengio, Deep sparse rectifier neural networks, in: *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, 2011, pp. 315–323.
- 845 [20] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, in: *Advances in Neural Information Processing Systems*, 2017, pp. 972–981.
- [21] Y. Bengio, P. Lamblin, D. Popovici, H. Larochelle, Greedy layer-wise training of deep networks, in: *Advances in neural information processing systems*, 2007, pp. 153–160.
850
- [22] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: A simple way to prevent neural networks from overfitting, *The Journal of Machine Learning Research* 15 (1) (2014) 1929–1958.
- 855 [23] J. Li, S. Zheng, Y. Tan, The effect of information utilization: introducing a novel guiding spark in the fireworks algorithm, *IEEE Transactions on Evolutionary Computation* 21 (1) (2017) 153–166.

- [24] M. S. Couceiro, R. P. Rocha, N. M. Ferreira, A novel multi-robot exploration approach based on particle swarm optimization algorithms, in: 2011 IEEE International Symposium on Safety, Security, and Rescue Robotics, IEEE, 2011, pp. 327–332.
- [25] H. N. Ataei, K. Ziarati, M. Eghtesad, A bso-based algorithm for multi-robot and multi-target search, in: International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, Springer, 2013, pp. 312–321.
- [26] K. Krishnanand, D. Ghose, A glowworm swarm optimization based multi-robot system for signal source localization, in: Design and control of intelligent robotic systems, Springer, 2009, pp. 49–68.
- [27] Z. Zheng, Y. Tan, Group explosion strategy for searching multiple targets using swarm robotic, in: 2013 IEEE Congress on Evolutionary Computation, IEEE, 2013, pp. 821–828.
- [28] G. M. Viswanathan, V. Afanasyev, S. Buldyrev, E. Murphy, P. Prince, H. E. Stanley, Lévy flight search patterns of wandering albatrosses, *Nature* 381 (6581) (1996) 413.
- [29] O. Bénichou, C. Loverdo, M. Moreau, R. Voituriez, Two-dimensional intermittent search processes: An alternative to lévy flight strategies, *Physical Review E* 74 (2) (2006) 020102.
- [30] V. Gazi, K. M. Passino, Stability analysis of social foraging swarms, *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)* 34 (1) (2004) 539–557.
- [31] L. Xie, G. Yang, J. Zeng, Z. Cui, Swarm robots search based on artificial physics optimisation algorithm, *International Journal of Computing Science and Mathematics* 4 (1) (2013) 62–71.

- [32] J. Li, Y. Tan, Triangle formation based multiple targets search using a swarm of robots, in: International Conference in Swarm Intelligence, Springer, 2016, pp. 544–552. 885
- [33] M. S. Couceiro, P. A. Vargas, R. P. Rocha, N. M. F. Ferreira, Benchmark of swarm robotics distributed techniques in a search task, *Robotics and Autonomous Systems* 62 (2) (2014) 200–213.
- [34] M. Dadgar, M. S. Couceiro, A. Hamzeh, Rdpso diversity enhancement based on repulsion between similar ions for robotic target searching, in: 890 Artificial Intelligence and Signal Processing Conference, 2018, pp. 275–280.
- [35] M. Dadgar, S. Jafari, A. Hamzeh, A pso-based multi-robot cooperation method for target searching in unknown environments, *Neurocomputing* 177 (2016) 62–74.
- [36] X. Yang, J. Yuan, J. Yuan, H. Mao, A modified particle swarm optimizer with dynamic adaptation, *Applied Mathematics and Computation* 189 (2) 895 (2007) 1205–1213.
- [37] Y. Tan, Y. Zhu, Fireworks algorithm for optimization, in: International Conference in Swarm Intelligence, Springer, 2010, pp. 355–364.
- [38] Z. Zheng, J. Li, J. Li, Y. Tan, Improved group explosion strategy for searching multiple targets using swarm robotics, in: 2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC), IEEE, 2014, pp. 900 246–251.
- [39] J. P. Hecker, M. E. Moses, Beyond pheromones: evolving error-tolerant, flexible, and scalable ant-inspired robot swarms, *Swarm Intelligence* 9 (1) 905 (2015) 43–70.
- [40] J. Gomes, P. Urbano, A. L. Christensen, Evolution of swarm robotics systems with novelty search, *Swarm Intelligence* 7 (2-3) (2013) 115–144.

- [41] M. Duarte, V. Costa, J. Gomes, T. Rodrigues, F. Silva, S. M. Oliveira,
910 A. L. Christensen, Evolution of collective behaviors for a real swarm of
aquatic surface robots, *PloS one* 11 (3) (2016) e0151834.
- [42] J. Foerster, I. A. Assael, N. de Freitas, S. Whiteson, Learning to communi-
cate with deep multi-agent reinforcement learning, in: *Advances in Neural
Information Processing Systems*, 2016, pp. 2137–2145.
- 915 [43] J. Foerster, N. Nardelli, G. Farquhar, P. Torr, P. Kohli, S. Whiteson, et al.,
Stabilising experience replay for deep multi-agent reinforcement learning,
arXiv preprint arXiv:1702.08887.
- [44] Q. Li, X. Du, Y. Huang, Q. Sykora, A. P. Schoellig, Learning of coordina-
tion policies for robotic swarms, arXiv preprint arXiv:1709.06620.
- 920 [45] Y. Bengio, et al., Learning deep architectures for ai, *Foundations and
trends® in Machine Learning* 2 (1) (2009) 1–127.
- [46] D. Erhan, Y. Bengio, A. Courville, P.-A. Manzagol, P. Vincent, S. Ben-
gio, Why does unsupervised pre-training help deep learning?, *Journal of
Machine Learning Research* 11 (Feb) (2010) 625–660.
- 925 [47] Z. Zheng, J. Li, J. Li, Y. Tan, Avoiding decoys in multiple targets searching
problems using swarm robotics, in: *2014 IEEE Congress on Evolutionary
Computation (CEC)*, IEEE, 2014, pp. 784–791.
- [48] S. Ioffe, C. Szegedy, Batch normalization: Accelerating deep network train-
ing by reducing internal covariate shift, in: *International conference on
machine learning*, 2015, pp. 448–456.
- 930 [49] D. P. Kingma, J. Ba, Adam: A method for stochastic optimization, arXiv
preprint arXiv:1412.6980.
- [50] X. Li, K. Tang, M. N. Omidvar, Z. Yang, K. Qin, H. China, Benchmark
functions for the cec 2013 special session and competition on large-scale
935 global optimization, *gene* 7 (33) (2013) 8.

- [51] J. Liang, B. Qu, P. Suganthan, Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization, Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore.
- [52] N. Hansen, The cma evolution strategy: A tutorial, arXiv preprint arXiv:1604.00772.
- [53] G.-B. Huang, Q.-Y. Zhu, C.-K. Siew, Extreme learning machine: theory and applications, *Neurocomputing* 70 (1-3) (2006) 489–501.
- [54] Q. Chen, B. Liu, Q. Zhang, J. Liang, P. Suganthan, B. Qu, Problem definition and evaluation criteria for cec 2015 special session and competition on bound constrained single-objective computationally expensive numerical optimization, Computational Intelligence Laboratory, Zhengzhou University, China and Nanyang Technological University, Singapore, Technical report.



Jie Li, received a Bachelor of Intelligence Science and Technology from University of Science and Technology Beijing, China, in 2013. He is currently majoring in Computer Science and working towards the PhD degree at Key Laboratory of Machine Perception (Ministry of Education) and Department of Machine Intelligence, School of Electronics Engineering and Computer Science, Peking University, Beijing, China. His research interests include Swarm Robotics, Evolutionary Computation, Machine Learning and Artificial Neural Networks.



Ying Tan, is a full professor and PhD advisor at the School
of Electronics Engineering and Computer Science of Peking University, and di-
960 rector of Computational Intelligence Laboratory at Peking University. He re-
ceived his B.Eng, M.S., and PhD from Southeast University, in 1985, 1988, and
1997, respectively. He is the inventor of Fireworks Algorithm (FWA). He serves
as the Editor-in-Chief of International Journal of Computational Intelligence
965 and Pattern Recognition (IJCIPR), the Associate Editor of IEEE Transaction on
Cybernetics (CYB), IEEE Transaction on Neural Networks and Learning Sys-
tems (NNLS), International Journal of Swarm Intelligence Research (IJSIR), In-
ternational Journal of Artificial Intelligence (IJAI), etc. He also served as an Ed-
itor of Springer's Lecture Notes on Computer Science (LNCS) for 20+ volumes,
970 and Guest Editors of several referred Journals, including IEEE/ACM Transac-
tions on Computational Biology and Bioinformatics (IEEE/ACM TCBB), In-
formation Science, Softcomputing, Neurocomputing, IJSIR, IJAI, B&B, CJ, etc.
He is an IEEE senior member and a member of Emergent Technologies Techni-
cal Committee (ETTC) of IEEE Computational Intelligence Society since 2010.
975 He is the founder general chair of the ICSI International Conference series. He
won the 2nd-Class Natural Science Award of China in 2009.