A Probabilistic Finite State Machine based Strategy for Multi-Target Search Using Swarm Robotics

Jie Li and Ying Tan

Key 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.
Email: ustblijie@126.com,ytan@pku.edu.cn

Abstract

As a distributed system, swarm robotics is well suited for the multi-target search task where a single robot is rather inefficient. In this paper, a model of the multi-target search problem in swarm robotics and its approximate mathematical representation are given, based on which a lower bound of the expected number of iterations is drawn. Two categories of behavior-based strategies for target search are introduced: one is inspired from swarm intelligence optimization while the other from random walk. A novel search strategy based on probabilistic finite state machine is put forward, showing the highest efficiency in all presented algorithms, which is very close to the optimal value in situations with a large number of robots. It has been demonstrated by extensive experiments that the novel strategy has excellent stability, striking a good balance between exploration and exploitation, as well as a good trade-off between parallelism and cooperative capability.

Keywords: swarm robotics, multi-target search, swarm intelligence optimization, random walk, probabilistic finite state machine.

1. Introduction

Swarm robotics is a field inspired by the self-organized behaviors of social animals [1], aiming at designing a large number of simple robots to complete some complex tasks in a low-cost way with high reliability and efficiency [2, 3] or to simulate some expected collective behaviors [4], through local interactions among robots and between the robots and environment [5], in which

people have done lots of various research and survey work [6, 7, 8, 9]. As a distributed system, swarm robotic system is well-suited for tasks involving area coverage[5], such as searching for multiple targets in a large space. Strategies for multi-target search have a broad prospect of application, such as hunting submarines[10], searching for victims and wreckage after air crash or shipwreck, monitoring the leak water quality[5], exploring and destroying battlefield targets, and so on.

Similar to the general foraging task, the search space in the multi-target search problem is very large. The large space emphasizes the importance of good exploration ability or high diffusion rate of the swarm. In addition, the influence scope of targets is also wide and robots can only perceive the local fitness information generated by targets (similar to radiation intensity but non-directional). The wide scope implies the necessity of good exploitation ability of robots [11]. For simplicity, the targets are static, and the fitness information will remain unchanged in the collecting process of the corresponding target, but will disappear immediately once the collection is finished. We may consider a more realistic scenario, such as the salvage task at sea. The people waiting for rescue are scattered in different areas, and a swarm of robots (ships or aircrafts) equipped with specific sensors (for life sign detection or other signals) are launched and instructed to search a designated region and find the people as soon as possible, and the signals will disappear once the people are saved.

The methods for designing collaborative mechanisms of swarm robotics can be divided into two categories: behavior-based design and automatic design [8]. Generally, behavior-based design is a bottom-up process, in which individual behaviors of robots are iteratively adjusted and tuned until the desired collective behavior is obtained. Automatic design methods mainly include reinforcement learning and evolutionary robotics, which can be considered as top-down approaches and generate behaviors automatically without the explicit intervention of the developer. Basically, the design methods of current strategies for multi-target search task can be classified as the behavior-based category [12, 13], such as methods based on artificial potential functions[14, 15], and methods adapted from some swarm intelligence optimization algorithms, such as Glowworm Swarm Optimization (GSO) [16], Particle Swarm Optimization (PSO) [17], Bee Swarm Optimization (BSO) [18], Fireworks Algorithm (FWA) [19], and Differential Evolution (DE) [20, 21].

In addition to swarm intelligence optimization, another important behavior-

based perspective is using mathematical physics methods to model and analyze the foraging and migratory behaviors of animals, which is often referred to as "random search"[22] or "stochastic optimal foraging theory"[23]. Lévy flight [24] is a typical random search strategy, which can also be called Lévy walk[25] except for a negligible difference (In Lévy walk models, "jumps are not instantaneous but a time interval related to a finite velocity to complete the jump is involved" [23]). Random walk strategies can find many applications in swarm robotics [3], and we put forward a benchmark algorithm combining linear ballistic motion with triangle estimation technology [26]. Random walk strategy such as Lévy flight, bears excellent exploration ability (i.e. high diffusion rate), which is important for searching tasks in large space. Furthermore, if the targets are distributed sparsely and can be located directly by robots in their influence scope, then random walk strategy may be the only reasonable solution. In the problem scenario of this article, random walk strategies are generally used in areas without fitness information to help robots move quickly to other areas.

The probabilistic finite state machine (PFSM) is also a behavior-based design method for swarm robotics [8], which is convenient for describing various kinds of algorithms in an intuitive way. In this paper, the PFSM is used as an approach to describe, analyze and design the strategy. The motivation is from the previous work about triangular formation search [26], where robots in a team can roughly maintain a triangular formation to improve the exploitation ability of robots. However, the algorithm is rather complex, causing difficulties in maintenance and expansion, thus the disbandment and reorganization of the formation are not considered. In addition, to a certain extent, the triangular formation leads to excessive concentration of resources, limiting the exploration ability of robots. Therefore, we want to design a self-organized triangular formation strategy, in which the formation can be disbanded or reorganized freely. However, it's found unnecessary to maintain a formation explicitly, which will increase the computation and communication load. In the new strategy, there is no grouping operation nor rigid triangular formation, and only the triangle estimation technology is adopted. The results in [26] are probably not easy to relate to those in the current paper for two reasons: the parameters of RPSO strategy in [26] are not well tuned, and the IS strategy in [26] is enhanced with the inertia mechanism and corresponds to the BMS in current paper.

This article mainly includes three contributions. Firstly, an approximate mathematical model is established for the multi-target search problem in

swarm robotics, based on which a lower bound of the expected number of iterations is derived. Secondly, three kinds of independent search strategies are introduced, each of which is combined by a kind of random walk strategy, the triangle estimation technology, and the inertia mechanism. Finally, a novel search strategy based on PFSM is proposed, showing the highest efficiency and the best stability in all presented algorithms. With a large population, the efficiency of the PFSMS is close to the theoretical lower bound of iterations for the problem. The PFSMS also strikes a good balance between exploration and exploitation, and a good trade-off between parallelism and cooperative capability.

The rest of the paper is organized as follows. In section II, both of two categories of searching algorithms and a formation search strategy are introduced. In section III, the model and analysis of the multi-target search problem in swarm robotics are stated. In section IV, the behavior-based strategies proposed in this paper are described in detail. In section V, experimental results and discussions are presented. Finally, the work is concluded in section VI.

2. Related Work

There are mainly two categories of behavior-based algorithms for target search, one inspired from swarm intelligence optimization algorithms while the other from random walk strategies. Similar to our problem model[11], swarm algorithms focus on search tasks in information-rich environment (the influence range of targets is wide), bearing strong exploitation but weak exploration abilities due to high degree of swarm connectivity. In contrast, random walk strategies mainly consider the situations of lacking fitness information (the influence of targets is narrow), possessing strong exploration but weak exploitation abilities owing to great diffusion capacities.

2.1. Swarm Algorithms for Target Search

To design the collaborative mechanisms of swarm robotics for the multi-target search task, a natural idea is to learn from the existing swarm optimization algorithms, such as PSO [27, 28], GSO [29, 16], BSO [30, 18] and so on. Robots, collaborative mechanisms among robots, and target search are analogous to candidate solutions, information mining mechanisms and optimization process respectively. In this section, four strategies inspired from swarm algorithms are introduced as follows.

$$v_i^{t+1} = wv_i^t + c_1 r_1 \left(pBest_i^t - x_i^t\right) + c_2 r_2 \left(gBest_i^t - x_i^t\right) \quad (1)$$

RPSO: Robotic Particle Swarm Optimization (RPSO) [17] is an extension of PSO, in which an obstacle avoidance component is added to the velocity update formula. In PSO, as shown in Eq. 1, three components are considered: inertia, cognition and social components. In iteration t, the position and velocity of particle i are denoted by x_i^t and v_i^t, and w, c_1, c_2 are named "inertia weight", "cognition coefficient" and "social coefficient", respectively. $pBest_i^t$ and $gBest_i^t$ are the best historical position of particle i and the best previous position among all the particles, and r_1, r_2 are random numbers uniformly distributed within [0,1]. In RPSO, as shown in Eq. 2, a component for obstacle avoidance is introduced, where p_i^t is an attractive position located away from obstacles, and c_3, r_3 are the corresponding coefficient and random number. In this paper, as shown in Eq. 3, a random component is introduced into RPSO to help robots to escape from local oscillation, where c_4 is the random efficient and m_i^t is a unit direction vector with the angle uniformly distributed within the range [0,2π].

$$v_i^{t+1} = wv_i^t + c_1 r_1 \left(pBest_i^t - x_i^t\right) + c_2 r_2 \left(gBest_i^t - x_i^t\right) + c_3 r_3 \left(p_i^t - x_i^t\right) \quad (2)$$

$$v_i^{t+1} = wv_i^t + c_1 r_1 \left(pBest_i^t - x_i^t\right) + c_2 r_2 \left(gBest_i^t - x_i^t\right) + c_3 r_3 \left(p_i^t - x_i^t\right) + c_4 m_i^t \quad (3)$$

A-RPSO: In Adaptive Robotic PSO (A-RPSO) [13], as shown in Eq. 4, the velocity update formula of each robot is similar to that in RPSO, except that the inertia weight w_i^t is a variable for each robot i and each iteration t, and its value depends on "evolutionary speed" and "aggregation degree" [31]. The "evolutionary speed" factor is described by the difference between personal best fitness values in adjacent iterations, and "aggregation degree" is determined by the difference between the best fitness and mean fitness in current iteration. The original A-RPSO algorithm focuses on the single target search problem, and to adapt to the multi-target search problem, we divide the swarm into several sub-swarms through limiting the communication range of robots. In this paper, as shown in Eq. 5, a random component m_i^t is also introduced into A-RPSO to help robots to avoid local oscillation.

$$v_i^{t+1} = w_i^t v_i^t + c_1 r_1 \left(pBest_i^t - x_i^t\right) + c_2 r_2 \left(gBest_i^t - x_i^t\right) + c_3 r_3 \left(p_i^t - x_i^t\right) \quad (4)$$

$$v_i^{t+1} = w_i^t v_i^t + c_1 r_1 \left(pBest_i^t - x_i^t\right) + c_2 r_2 \left(gBest_i^t - x_i^t\right) + c_3 r_3 \left(p_i^t - x_i^t\right) + c_4 m_i^t \quad (5)$$

GES: Group Explosion Strategy (GES) [19] borrows some ideas from the FWA [32], a swarm intelligence optimization algorithm inspired by the firework explosion. In GES, the entire swarm is divided into small groups automatically according to the perception range of each robot. The key idea is to move the geometric center of a group to the position of an optimal individual in the group, and the group will split if the size is larger than a threshold. In iteration t, as shown in Eq. 6, the guidance vector of robot i from its group is denoted by g_i^t, where N_i^t is the set including robot i and all its neighbors and $|N_i^t|$ is the cardinality of N_i^t, and x_j^t is the current position of robot j and x_b^t is one of the best positions in the group.

$$g_i^t = x_b^t - \frac{\sum_{j \in N_i^t} x_j^t}{|N_i^t|} \quad (6)$$

IGES: Improved Group Explosion Strategy (IGES) [33] is an improved version of GES. In the simulation experiments of GES, it is found that if all robots in a group share an optimal fitness, selecting an optimal individual randomly may cause robots to fall into local oscillation or fall back to worse regions. In IGES, four strategies have been developed to deal with various situations, and there are mainly two improvements. Firstly, as shown in Eq. 7, the moving reference point is the center of all best positions in the group instead of one of the best positions, where B_i^t is the set of robots in N_i^t that have the maximum fitness and $|B_i^t|$ is the cardinality of B_i^t. Secondly, the group will split if all robots share the same fitness value or the group size is larger than a threshold.

$$g_i^t = \frac{\sum_{j \in B_i^t} x_j^t}{|B_i^t|} - \frac{\sum_{j \in N_i^t} x_j^t}{|N_i^t|} \quad (7)$$

2.2. Random Walk Strategies for Target Search

Random search strategies [34] are derived from the study of the migration behaviors of foraging organisms [35, 36]. Due to the sparseness, renewability, fractal property of the distribution of food resources, Lévy Flights are

popular to explain the trajectories of foraging organisms [37], which are a special class of random walk strategies whose step lengths come from probability distributions with heavy power-law tails [24]. An alternative is the intermittent search strategy, which combines phases of slow motion, allowing searchers to detect the target, and phases of fast motion during which targets cannot be detected [38]. A limiting case of Lévy Flights is the ballistic motion strategy, in which the searcher selects a direction randomly and keeps moving in a straight line until targets or boundaries are detected. In the case of dense distribution of targets, strategies with power-law or exponential distributions are more efficient than the ballistic motion strategy. Conclusions about situations of sparse targets are stated in the following part.

Lévy Flight: In this model, the step length l of searchers obeys the power law distribution $p(l) = l^{-u}$, where $1 < u < 3$. If targets can be regenerated at the same location after a finite time, the foraging is non-destructive and $u \approx 2$ is the optimal value for a search in any dimension. In the case of destructive foraging, targets are non-renewable and lower values of u lead to more efficient search, thus the optimal strategy reduces to a linear ballistic motion [22].

Intermittent Search: This model is a two state search processes for non-renewable targets, including slow reactive phases (phase 1) randomly interrupted by fast relocating ballistic flights with a constant velocity v and a random direction (phase 2). The duration of each phase i is exponentially distributed with the mean τ_i, and the searcher can only find a target during reactive phases, for fast motion usually strongly degrades perception abilities. Intermittent strategies constitute optimal strategies with proper values of τ_i [39].

2.3. Formation Technologies in Search Strategies

Apart from swarm algorithms and random walk strategies, another thing needed to be introduced here is the formation control technology for multiple robots or vehicles, which are usually considered in aircraft formation [40]. To improve the exploitation ability, a triangle formation search strategy [26] adopts the behavior-based control[41] to maintain the formation, where each robot determines its proper position based on a reference point[42].

Triangle Formation Search: In this strategy [26], the swarm is divided into three-robot teams as many as possible, and each team forms roughly an equilateral triangle, including a leader and two members. The leader determines the moving direction while the members follow the leader and

maintain the formation. The strategy contains five stages: "initial grouping", "initial diffusion", "search in areas without fitness", "search in areas with fitness" and "target collecting". The "initial grouping" stage is to divide the swarm into three-robot teams, and in "initial diffusion" stage the leader will select a sparse direction and lead the team forward. In areas without fitness, the leader will search randomly whose step size obeys exponential distribution, and in areas with fitness, the leader will estimate the gradient direction according to the team information, and the estimation technology is also adopted by the proposed strategy (as shown in Sec. 4.1.2). In the "target collecting" stage, robots having found targets will broadcast the information within the team and the other two will move towards the target.

A proper formation control will help robots to complete tasks in an organized and efficient way. However, the communication and computation costs associated with formation technology will add complexity to system maintenance.

3. Problem Statement

In the multi-target search problem, a swarm of robots are delivered into a vast unknown space where multiple targets are distributed randomly, and the task for robots is to search and collect (or destroy) the targets as soon as possible through certain collaborative mechanism. In the simplest case, only three kinds of objects are considered: environment space, robots and targets. In addition, obstacles, decoys [43] and inference sources can also be introduced into the problem [11]. Because we focus on the search efficiency in this article, only the simplest case is studied here.

3.1. Assumptions

In order to establish the simulation model for the problem, some assumptions are made at first, which are summarized and presented as follows.

- Environment: the entire space is vast compared with the size and perception range of each robot.

- Targets: static, small size but with wide influence scopes where fitness values decrease with increasing distance to the target. The influence of each target remains unchanged during the collecting process but will disappear once the target is collected. In the overlap area of multiple

targets, the influence equals the largest one generated by targets. The targets are distributed randomly but uniformly over the search space.

- Robots: without prior knowledge about the environment, local interaction (perception and communication), limited speed and memory.

- Robot swarm: no global leader, no central control, all individuals starting from the same region.

- Fitness value: discretized to enhance the robustness of the system considering the limitation of sensor accuracy and the influence of ambient noise.

- Iteration frequency: high enough to ensure the fitness difference between two adjacent iterations is small (to avoid missing promising positions). In each iteration, each robot gathers the information from sensors, historical records, neighbors, and makes decisions autonomously. The iteration frequency is fixed, and robots can work in an asynchronous way.

3.2. An Idealized Model

Based on the above assumptions, an idealized model for the simplest case of the problem is shown in Fig. 1, which is described as following.

- Environment: a square with size of $1000 * 1000$.

- Target: a circle with radius of r_t (10 units) in which robots can perceive the target. The targets are distributed uniformly over the environment, and the influence scopes are a series of annuli with width of 5 units whose fitness values descend linearly by 1 unit from about 40 till 1. Each target requires 10 processing steps to be fully collected, which could be done by one robot alone in 10 iterations and by ten robots in one iteration.

- Robots: a square with size of $1 * 1$, a circular range with radius of 20 units for perception and communication (i.e. robots can only communicate with the perceived neighbors), a limit of 5 units for the maximum speed (maximum moving distance in an iteration), can memorize information of 10 iterations (positions and fitness values).

- Robot swarm: typical population size is 50.

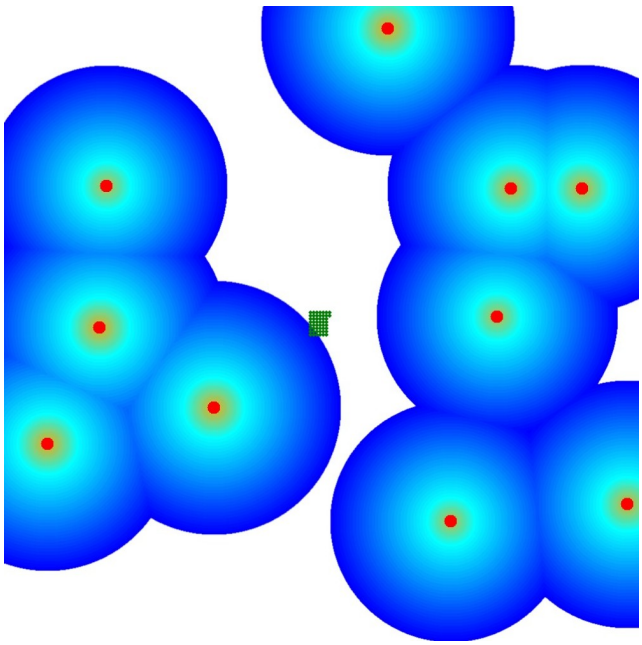

Figure 1: A screenshot of the problem at the beginning of a simulation. Red circles stand for the targets while central green dot array represents the initial robot swarm. The color of the circles around the target illustrates the fitness value of that position.

3.3. Mathematical Modeling and Analysis

In the multi-target search task, it takes the robot swarm a finite number of iterations to search and collect all targets distributed over the environment. To estimate the lower bound of the expected number of iterations, some mathematical approximations of the problem are introduced to simplify the analysis.

3.3.1. Approximations Used for the Problem

Since it is the estimation for the optimal solution of the problem, we may assume that there are more robots than targets, and the robots have been informed of the locations of targets beforehand. Another assumption is that each robot can only participate in the collection of one target, thus the original search problem is transformed into an assignment problem, in which the swarm is divided into different teams to collect different targets separately. Each team is assigned to one target, containing at least one robot.

Suppose there are N robots and T targets in the environment, and the positions of all targets are known in advance, to which the distances from the

center of the search space are successively from small to large: $d_1, d_2, ..., d_T$. Corresponding to T targets, N robots are divided into T teams, and the number of robots in each team is $k_1, k_2, ..., k_T$ respectively. The initial position of the robot swarm is the center of the map, and the maximum speed of robots is v_{max}. Generally larger speed means higher efficiency, so we let all robots run at the maximum speed. Each target needs 10 processing steps to be fully processed, i.e. one robot needs 10 iterations while 10 robots need one iteration. Let S denote the policy set for assignment (i.e. how to assign robots to targets), then the task can be formalized into an optimization problem below (Eq. 8). It should be pointed out that we do not need to actually construct the optimal strategy for the assignment problem, but only need to analyze the performance that the optimal strategy can achieve.

$$\begin{cases} \min\limits_{s\in S}\left\{\max\limits_{i}\left\{\frac{d_i}{v_{max}}+\frac{10}{k_i}\right\}\right\} \\ s.t.\sum_{i=1}^{T} k_i = N, k_i \in N^+ \end{cases} \tag{8}$$

In addition, some approximations are used to simplify the calculation. Firstly, the square of the search space is regarded as a circle with equal area for convenience, as is shown by Eq. 9, in which L is the side length of the square and R is the radius of the circle. Another thing can be simplified is the item $10/k_i$ in Eq. 8, representing the cost of target collection. It takes one robot 10 iterations to collect a target while one iteration for 10 robots, and the actual iterations should be integers, so the range of the collection item is $[1, 2, ..., 10]$, and can be represented approximately as 5.5 ± 4.5 due to its small proportion.

$$L^2 = \pi R^2 \tag{9}$$

3.3.2. An Approximate Lower Bound of the Expected Number of Iterations

Based on the approximations made above, a lower bound of the expected iteration number in search task (equivalent to a upper bound of the efficiency) can be represented approximately by utilizing the expectation of the maximum distance from the map center to targets $\mathrm{E}[d_T]$, as is illustrated by Eq. 10.

$$\min_{s\in S}\left\{\max_{i}\left\{\frac{d_i}{v_{max}}+\frac{10}{k_i}\right\}\right\} \approx E\left[\frac{d_T}{v_{max}}\right] + 5.5 \pm 4.5 \tag{10}$$

Let r denote the distance from a target to the center of the map, then its probability density function (Eq. 11) and distribution function (Eq. 12) can be drawn respectively. The probability density of the ith farthest distance d_i can be represented as Eq. 13. Then we can obtain the probability density of d_T (Eq. 14) and the corresponding expectation $E[d_T]$ (Eq. 15).

$$p(r) = \frac{2r}{R^2} \tag{11}$$

$$F(r) = \frac{r^2}{R^2} \tag{12}$$

$$p_i(r) = \frac{T!}{(i-1)!(T-i)!} F(r)^{i-1} (1 - F(r))^{T-i} p(r) \tag{13}$$

$$p_T(r) = \frac{2T}{r} \left(\frac{r^2}{R^2} \right)^T \tag{14}$$

$$E[d_T] = \frac{2T}{2T+1} R \tag{15}$$

Finally, we can obtain the lower bound with given parameters. According to the problem configuration, an approximate lower bound of the expected number of the iterations required in search task can be calculated by Eq. 10 and Eq. 16. With given parameters: side length of the square $L = 1000$, number of targets $T = 10$, maximum speed of robots $v_{max} = 5$, we can obtain the range of approximate lower bound $[108.46, 117.46]$, and 117.46, as the upper limit of the interval, means that it takes the swarm around 120 iterations to collect all the targets if the farthest one is always collected by a single robot.

$$\frac{E[d_T]}{v_{max}} = \frac{2TL}{(2T+1)v\sqrt{\pi}} \tag{16}$$

4. Proposed Methods

In this section, three independent search strategies based on random walk and a PFSM-based search strategy are proposed, the former mainly serving as a benchmark while the latter trying to approach the optimal strategy for the problem. For the former, we introduce three aspects: wide-area search,

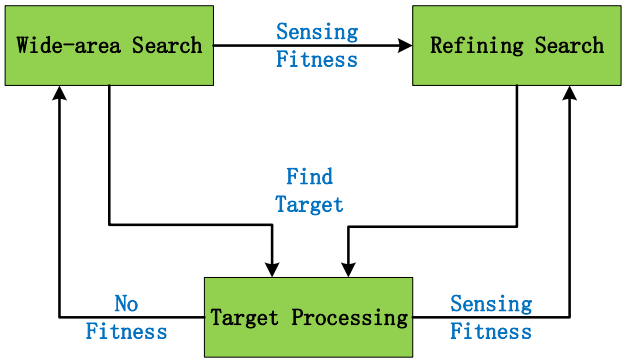

Figure 2: A three-phase search framework for the multi-target search problem. When searching in areas without fitness, robots are in the phase of wide-area search, and will get into the phase of refining search once sensing fitness. Whenever a target is found, robots will switch to collecting it.

refining search and inertia mechanism. For the latter, we first model the search strategy as a complete three-state PFSM, and then simplify it to reduce the parameters. Finally, based on the simplified PFSM, we establish the search strategy proposed in this paper.

As is stated in the assumptions, the size and perception range of each robot is small, and the targets are distributed sparsely, whose influence scopes are large but will disappear once the target is collected. In addition, a robot cannot move too far in one iteration to obtain sufficiently detailed information. Therefore, robots will spend lots of iterations searching in areas without fitness, and the whole process is shown in Fig. 2. When searching in areas without fitness, robots are in the phase of wide-area search, and they should explore unknown areas as far as possible to get information about targets. Once sensing fitness, robots in wide-area search will get into the phase of refining search, and they are supposed to use fitness information to approach the target. Whenever a target is found, robots in wide-area or refining search will switch to target processing phase. Robots that complete the target processing will again get into wide-area search or refining search according to whether new fitness information is perceived.

4.1. Independent Search Strategies

It's a natural idea to use the three-phase search framework to guide the design of search strategies, and random walk strategies can be used in the phase of wide-area search while some gradient estimation technologies can

be used in the refining search phase. "Independent Search Strategies" are the combination of random walk strategies and triangle estimation technology, in which there are no interactions among robots, so the corresponding performance can serve as a benchmark for the multi-target search problem.

4.1.1. Random Walk Strategies for Wide-area Search

Earlier in the article, three kinds of random walk strategies are introduced: Lévy Flights, ballistic motion and intermittent search.

In our problem model, the targets are non-regenerative and distributed sparsely, which means that the search performance of Lévy Flights increases with the parameter u approaching 1 and the ballistic motion strategy is a promising candidate, thus we implement these two strategies and optimize the parameter u in Lévy Flights to 1.001 (the corresponding algorithm is represented by LFS, i.e. Lévy Flight Search), and the other strategy is expressed as BMS (Ballistic Motion Search).

In intermittent search strategies, robots in phase 2 (fast motion phase) cannot detect targets, i.e. speed affecting perception, and in our implemented version (represented as IS, Intermittent Search), robots in phase 2 will just ignore the fitness information. The duration of each phase i is exponentially distributed with the mean τ_i. In experiments for parameter optimization, intermittent search strategies with smaller τ_2 and larger τ_1 perform better. In the IS strategy, τ_2 and τ_1 are set respectively as 0.3 and 3.0 times of the side length of the map, from which we can see that the fast motion phase contributes little to the search.

4.1.2. Triangle Gradient Estimation for Refining Search

When sensing fitness information, robots will step into the refining search phase and integrate current and historical data to calculate the approximate gradient direction, based on the presumption that the fitness value varies almost linearly with the distance in local area [26].

To estimate the gradient direction, three positions and their corresponding fitness values are used and the basic idea is to construct a vector perpendicular to the local contours. In LFS, BMS and IS, the three points are the current position, and two historical positions with the best and worst fitness values. Various cases are presented as follows and the details are described in Alg. 1.

- Case I: all three positions share the same fitness value, which means

that the robot is still in the area without fitness, and it can just keep the original direction.

- Case II: two positions share the same better fitness value, then the gradient vector equals the center of the two better positions minus the worse one.

- Case III: two positions share the same worse fitness value, then the gradient vector equals the better position minus the center of the two worse ones.

- Case IV: all positions have different fitness values, then the gradient vector constructed is perpendicular to the local contour line (Fig. 3). In Fig. 3, A, B and C stand for the positions, and their fitness values satisfy inequality $f(A) > f(B) > f(C)$. Based on the local linearity, $f(B') = f(B)$ and the position of B' can be calculated from Eq. 17. The line BB' serves as a contour line, so its vertical vector $B'P$ is the gradient direction (Eq. 18).

Algorithm 1 Triangle Gradient Estimation

Require: $P_i(x_i, y_i), f_i(i = 1, 2, 3)$: positions and fitness values of three points.
$\vec{v}_{last}$: the previous velocity of robot r
Ensure: $\vec{v}$: the new velocity of robot r
$(P_A, P_B, P_C) \leftarrow sort(P_1, P_2, P_3)$ by $f_i(i = 1, 2, 3)$, so that $f_A \geq f_B \geq f_C$
if $f_A = f_C$ **then**
  $\vec{v} \leftarrow \vec{v}_{last}$ {Case I}
else if $f_A = f_B$ **then**
  $\vec{v} \leftarrow (P_A + P_B)/2 - P_C$ {Case II}
else if $f_B = f_C$ **then**
  $\vec{v} \leftarrow P_A - (P_B + P_C)/2$ {Case III}
else
  $P_{B'} \leftarrow P_A + (P_C - P_A) \times (f_A - f_B)/(f_A - f_C)$ {Case IV}
  $BB' \leftarrow P_{B'} - P_B$
  Calculate $B'P$ from Eq.(18)
  $\vec{v} \leftarrow B'P$
end if

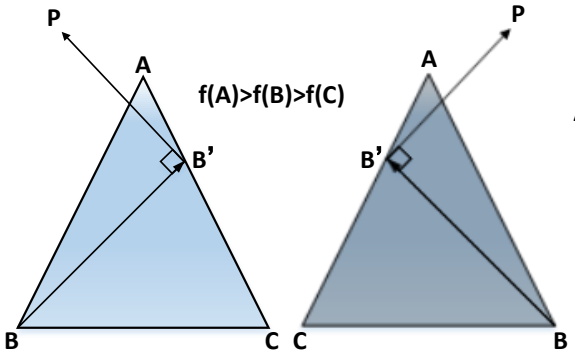

Figure 3: Calculation of the gradient direction when all positions have different fitness values.

$$\vec{BB'} = \vec{BA} + \vec{AC} \cdot \frac{\mathrm{f}(A) - \mathrm{f}(B)}{\mathrm{f}(A) - \mathrm{f}(C)} \tag{17}$$

$$\begin{cases} \vec{B'P} \cdot \vec{BB'} = 0 \\ \vec{B'P} \cdot \vec{B'A} > 0 \end{cases} \tag{18}$$

4.1.3. Inertia Mechanism

Generally, the inertia mechanism (Eq. 19) is used to stabilize the moving direction and enhance the ability to escape from local extremum in optimization algorithms, such as the inertia weight in PSO, and it is also introduced into the independent search strategies for the same consideration. As is stated above, robots sensing fitness will get into the phase of refining search, and the introduction of inertia mechanism will to some extent help robots escape from the influence scopes of nearby targets and return to the wide-area search, so the robot swarm can obtain a superior exploration ability. Besides, in 'Case II and Case III' of the triangle gradient estimation, if three positions are collinear, robots may fall into local oscillation, which can be overcome by introducing inertia mechanisms. In Eq. 19, v_t and $v_{s,t}$ are the velocity and velocity increment from strategies in the tth iteration, and w is the inertia weight, a number within the range of $[0, 1)$.

$$v_{t+1} = wv_t + (1 - w)\, v_{s,t} \tag{19}$$

4.2. A PFSM-based Search Strategy

Although following the three-phase search framework is a natural design approach, the deterministic separation of exploration and exploitation limits the performance of search strategies, for which reason the inertia mechanism is introduced to strengthen group diffusion in the influence scopes of targets. Generally, the individual robot in swarm robotics does not plan its future actions, and takes decisions only on the basis of its sensory inputs and/or its internal memory, and the probabilistic finite state machines (PFSMs) is one of the most adopted methods to design such behaviors [8].

4.2.1. A Complete Three-State PFSM

In swarm robotics, PFSMs can be used to model the behaviors of individual robots in various tasks, and the first step is to analyze the task procedure and extract typical states. As is shown in Fig. 4, three states are enough to describe the task: diffusion, search and target processing, which can be represented as s_i $(i = 1, 2, 3)$. Each state can transfer to another one (including itself) with a certain probability.

The probability values for state transition or holding depend on the information about each robot itself (historical positions and fitness values, current fitness value, current state, targets) and its neighbors (current positions and fitness values, targets), from which we can derive many decision factors, such as *'whether the robot has found a target or not', 'whether the neighbors have found targets or not', 'the number of neighbors', 'the fitness value of the robot', 'the current iteration number', 'the number of iterations kept by current state'* and so on.

4.2.2. A Simplified Three-State PFSM

Let D and d denote the set of decision factors and a set of values, then for each d, 9 probability values $P(s_i|s_j, d)$ (corresponding to the arrows in Fig. 4) are needed to be designated to obtain the probability table for the complete three-state PFSM, which will significantly increase the system complexity and make it hard to optimize parameters. According to the task characteristics, we simplify the decision-making process by introducing some deterministic decisions and select several key decision factors on the following considerations:

- Once a target is found by robots, it should be collected as soon as possible. Therefore, the decisions about targets should be deterministic,

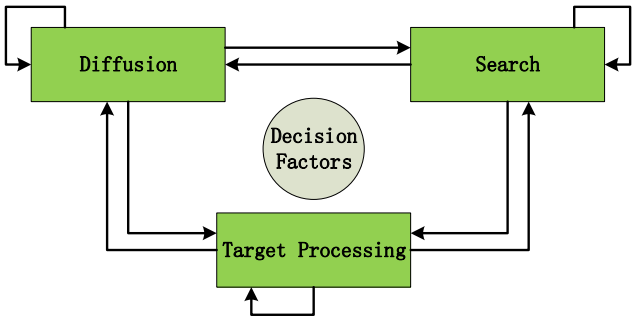

Figure 4: A three-state PFSM for the multi-target search task, including diffusion, search and target processing. Each state can transfer to another state (including itself) with a certain probability designated according to decision factors.

and the decision factors about targets should be selected (i.e. *'whether the robot has found a target or not'*, *'whether the neighbors have found targets or not'*).

- In order to make a better balance between exploration and exploitation, a smooth transition between 'Diffusion' and 'Search' is a promising idea, thus the factor *'the number of iterations kept by current state'* is selected to calculate the probability of holding current state ('Diffusion' or 'Search').

- If no target is found and the probability condition for holding current state is not satisfied, then how does a robot choose a state from 'Diffusion' and 'Search'? We can just consider the probability for 'Diffusion', and select the factor *'the number of neighbors'*, which embodies the congestion degree of the neighborhood.

Based on the above analysis, a novel corresponding strategy is put forward, which is abbreviated to PFSMS (the PFSM-based Search strategy). As is shown in Fig. 5, the simplified PFSM also shares the same three states, and a virtual state named 'Start Diffusion or Search' is introduced to help to understand the internal process, which is not a real state but a temporary step. Black solid arrows illustrate probabilistic decisions while red dashed arrows represent deterministic decisions.

- If a robot finds a target, it will switch to process the target at once, regardless of its current state.

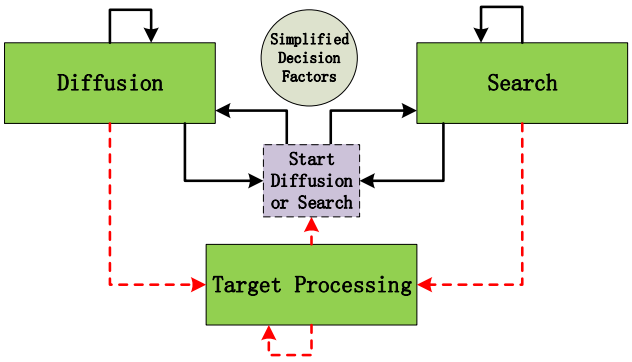

Figure 5: A simplified three-state PFSM for the multi-target search task, including diffusion, search and target processing. The central 'Start Diffusion or Search' is a virtual state to help to understand the internal process. Black solid arrows illustrate probabilistic decisions while red dashed arrows represent deterministic decisions.

- If a robot in 'Diffusion' or 'Search' satisfies the probability condition of holding state, it will keep its current state, otherwise it will get into the virtual state temporarily.

- If a robot has just finished the target processing and could find no other targets nearby, it will also get into the virtual state temporarily.

- Robots in virtual state will make a probabilistic decision according to the number of neighbors, if the probability condition of diffusion is satisfied, it will start the diffusion state, otherwise it will start to search.

In 'Search' state, the 'triangle gradient estimation' is used combining with 'inertia mechanism', the former prefers using neighbor information (neighbors sensing the best and the worst fitness values) rather than historical information, while the latter is only applied to robots having neighbors in order to stabilize the direction and avoid local oscillation (the inertia weight $w = 0.55$), and the details are described in Alg. 2. In 'Diffusion' state, robots will select a direction where the neighbors are sparse and keep moving in a straight line until the end of the state.

The boolean variable of 'whether finding targets' takes 'yes' if a robot or its neighbors find targets, and robots will deterministically process or move to targets. The 'number of neighbors' N_b is used to calculate the diffusion probability P_d in Eq. 20, where T_b is a threshold value ($T_b = 2.3$ in our

Algorithm 2 Triangle Search

Require: $P_r(x_r, y_r), f_r$: information of robot r
$P_{\{n_i\}}, f_{\{n_i\}}$: information from neighbors of robot r
$P_{\{h_i\}}, f_{\{h_i\}}$: history information of robot r
$\vec{v}_{last}$: the previous velocity of robot r
w : the inertia weight used in the $InertiaMechanism$
Note: $P_i(x_i, y_i)$ and f_i mean the robot's position and the corresponding fitness value respectively

Ensure: $\vec{v}$: the new velocity of robot r

$P_1 \leftarrow P_r, f_1 \leftarrow f_r$
if number of neighbors > 1 **then**
  $P_2, f_2 \leftarrow$ the neighbor with the best fitness value
  $P_3, f_3 \leftarrow$ the neighbor with the worst fitness value
else if number of neighbors $= 1$ **then**
  $P_2, f_2 \leftarrow$ the only neighbor
  $P_3, f_3 \leftarrow$ the history with the best fitness value
else
  $P_2, f_2 \leftarrow$ the history with the best fitness value
  $P_3, f_3 \leftarrow$ the history with the worst fitness value
end if
$\vec{v} \leftarrow TriangleGradientEstimation(P_{1\sim3}, f_{1\sim3}, \vec{v}_{last})$ {Alg. 1}
if number of neighbors > 1 **then**
  $\vec{v} \leftarrow 0.9 \times (\vec{v}/|\vec{v}|) + 0.1 \times RandomVector()$ {$RandomVector()$ is a function for generating a unit vector with a random direction}
else
  $\vec{v} \leftarrow w \times (\vec{v}_{last}/|\vec{v}_{last}|) + (1 - w) \times (\vec{v}/|\vec{v}|)$ {Eq.(19)}
end if

strategy). The 'number of iterations kept by current state' N_h is used to calculate the probability of holding current state P_h in Eq. 21, where P_{ini} is an initial probability ($P_{ini} = 0.9997$ in our strategy).

4.2.3. The PFSM-based Search strategy

In Fig. 6, we show the decision flow of the search strategy based on the simplified PFSM, and the details are described in Alg. 3. The orange diamond shaped frames represent condition judgment, if the condition is satisfied, the down branch will be performed, otherwise the right branch will be chosen. R_1 and R_2 are two random numbers uniformly distributed within the range [0,1], while P_h and P_d denote the probability of holding current state and diffusion probability respectively. As a small trick, the robot collecting targets will set its P_h to 0, thus it will restart the diffusion or search process after finishing the collection.

$$P_d = \begin{cases} 1 - \frac{T_b}{N_b} & , N_b > T_b \\ 0 & , N_b \leq T_b \end{cases} \tag{20}$$

$$P_h = P_{ini}^{N_h} \tag{21}$$

In each iteration, each robot makes a single decision (i.e. choosing one of the blue rectangles in Fig. 6) based on the information obtained and executes the corresponding operations. For example, 'Move to targets' is to keep moving to the target before the next iteration, and 'Process targets' is to process one unit of the target before the next iteration (each target has 10 units). There may be multiple robots involved in the same target processing, so the target may be processed before the next iteration, that is, the target information is no longer perceived in the next iteration, and robots will transfer to other states according to their new decisions.

5. Simulation Results and Discussions

In this section, several groups of experiments are conducted for two purposes: parameter optimization, and performance comparison. For the former, three parameters of the PFSMS strategy are tuned one by one, and for the latter, three problem settings are used to study the performance comparison of various search strategies, including different population sizes, different numbers of targets, and different collection times of targets (i.e. the number of iterations needed for one robot to collect one target).

Algorithm 3 The PFSM-based Search strategy

Require: at iteration t

$P_r^t(x_r^t, y_r^t), f_r^t, T_r^t$: information of robot r

$P_{\{n_i\}}^t, f_{\{n_i\}}^t, T_{\{n_i\}}^t$: information of neighbors of robot r

$P_{\{h_i\}}^t, f_{\{h_i\}}^t$: history information or robot r

$\vec{v}_{last}^t$: the previous velocity of robot r

w : the inertia weight used in the $InertiaMechanism$

v_{max} : the maximum speed of robot r

Note: $T_r^t, T_{\{n_i\}}^t$ contain information about target (whether robots have found targets and the targets' positions)

Ensure: $\vec{v}_r^t$: the new velocity of robot r at iteration t

Or transfer to the "Target Processing" state

for each robot r in the swarm at iteration t **do**
  if robot r has found a target **then**
    robot r will transfer to the "Target Processing" state, and collect one unit (total 10 units for each target)
  else if neighbors of robot r have found a target **then**
    $\vec{v}_r^t \leftarrow$ robot r will move towards the target
  else if $R_1 < P_h$ **then**
    $P_h \leftarrow P_h \times P_{ini}$ {Eq.(21)}
    if robot r is in the "Diffusion" state **then**
      $\vec{v}_r^t \leftarrow \vec{v}_{last}^t$, keep the "Diffusion" state
    else
      $\vec{v}_r^t \leftarrow TriangleSearch()$, keep the "Search" state {Alg. 2}
    end if
  else if $R_2 < P_d$ **then**
    $P_h \leftarrow P_{ini}$, start the "Diffusion" state
    $\vec{v}_r^t \leftarrow$ choose a direction with the fewest neighbors
  else
    $P_h \leftarrow P_{ini}$, start the "Search" state
    $\vec{v}_r^t \leftarrow TriangleSearch()$ {Alg. 2}
  end if
  $\vec{v}_r^t \leftarrow v_{\max} \times (\vec{v}_r^t / |\vec{v}_r^t|)$
end for

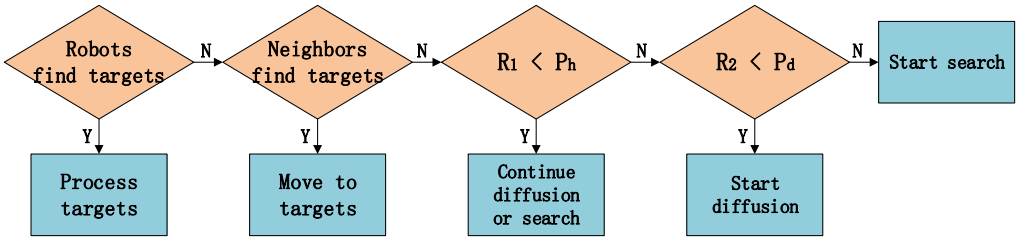

Figure 6: The decision flow of the search strategy based on the simplified PFSM. The orange diamond shaped frames represent condition judgment, if the condition is satisfied, the down branch will be performed, otherwise the right branch will be chosen. R_1 and R_2 are two random numbers uniformly distributed within the range [0,1], while P_h and P_d denote the probability of holding current state and diffusion probability respectively.

In each experiment, 40 random maps are generated and each strategy is repeated for 25 times, and the results in this section are the average value of these 1000 runs. The criteria for measuring the efficiency of searching strategies, are the mean and standard deviation of the number of iterations required to collect all targets, which are denoted by mI and dI, respectively. Criterion mI indicates the searching efficiency of strategies while dI reflects the stability. The experimental platform is built on the C# language and the Microsoft XNA framework (for game development). It mainly has two major functions: parallel test and visual simulation. The former is used for large-scale performance testing of the strategy, and the latter is used for visualizing the principles, effects and the existent problems of the strategy. A sample map showing 10 targets and 50 robots at the beginning of a simulation is displayed in Fig. 1.

5.1. Algorithms for Comparison

All parameters of the comparison algorithms are tuned under the same experimental conditions, where the map size is 1000*1000, containing 50 robots and 10 targets, and each target requires 10 processing steps to be fully collected. The optimization aims to minimize the average number of iterations required to collect all the targets (i.e. mI). All algorithms and their corresponding parameter configurations are as follows:

- RPSO: Robotic Particle Swarm Optimization, inertia weight $w = 3.0$, cognition coefficient $c_1 = 1.0$, social coefficient $c_2 = 2.0$, obstacle avoidance coefficient $c_3 = 0.0$ (obstacle-free), random coefficient $c_4 = 0.1$.

- A-RPSO: Adaptive Robotic Particle Swarm Optimization, inertia weight $w_{ini} = 1.0$, cognition coefficient $c_1 = 0.6$, social coefficient $c_2 = 0.4$, obstacle avoidance coefficient $c_3 = 0.0$ (obstacle-free), random coefficient $c_4 = 0.1$, scale factor $\alpha = 0.3$, scale factor $\beta = 0.7$.

- IGES: Improved Group Explosion Strategy, group size threshold $size = 5$. GES is not served as a comparison algorithm in experiments, for the performance of IGES is much better.

- LFS: Lévy Flight Search, exponential factor $u = 1.001$, inertia weight $w = 0.6$.

- BMS: Ballistic Motion Search, inertia weight $w = 0.7$.

- IS: Intermittent Search, expectation factor for phase 1: $\tau_1 = 3.0$ times of the side length of the map, expectation factor for phase 2: $\tau_2 = 0.3$ times of the side length of the map, inertia weight $w = 0.6$.

- TFS: Triangle Formation Search, the parameter setting is the same as that in [26].

- PFSMS: PFSM-based Search, initial probability for holding state $P_{ini} = 0.9997$, inertia weight $w = 0.55$, diffusion threshold $T_b = 2.3$, and the detailed experiments for parameter optimization are presented in the following section.

5.2. Parameter Optimization for the PFSM-based Strategy

Benefiting from the simplification for the three-state PFSM, the number of parameters of PFSMS is reduced to 3, including P_{ini}, w and T_b. In the experiments, each parameter is optimized separately and sequentially with the other two parameters fixed. Firstly, the parameter P_{ini} is optimized with $w = 0.0$ (no inertia mechanism) and $T_b = 2.0$ (2 neighbors at least). Then the parameter w is optimized with $P_{ini} = 0.9997$ and $T_b = 2.0$. Finally, the parameter T_b is optimized with $P_{ini} = 0.9997$ and $w = 0.55$. It's worth mentioning that the maps used for parameter optimization are different from those used in comparison experiments in the following parts, which to some extent guarantees the generalization of the comparison experimental results.

Firstly, the optimization for parameter P_{ini} is investigated. Under conditions of $w = 0.0$ and $T_b = 2.0$, the first-rough-then-precise method is adopted to adjust parameter P_{ini}, of which the first and second optimization intervals

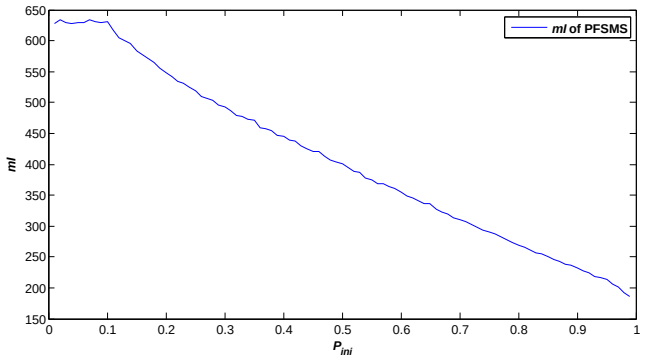

Figure 7: mI of PFSMS at different P_{ini}, which varies within $[0.01, 0.99]$ with step length 0.01. mI denotes the mean of the number of iterations.

are $[0.01, 0.99]$ (with step length 0.01) and $[0.9900, 0.9999]$ (with step length 0.0001), and the corresponding experimental results are presented in Fig. 7 and Fig. 8 respectively. The optimal value of P_{ini} in these two figures are 0.99 ($mI = 185.36$) and 0.9997 ($mI = 153.79$). Although there are some small fluctuations on curves, the overall trend is obvious. As for the rationality of the value, since the P_h goes down exponentially with the iterations kept by current state, parameter P_{ini} should be big to ensure enough time for state-keeping, meanwhile setting aside opportunities for state switching is also important to balance the exploration and exploitation of the swarm. And there is an interesting equation: $\left(0.5^{\frac{1}{50}}\right)^{\frac{1}{50}} \approx 0.9997$, which means that the whole robot swarm (containing 50 robots) can keep their current states for 50 iterations with a probability of 0.5, so the state transition is possible in the process.

Secondly, we study the optimization for parameter w. Under conditions of $P_{ini} = 0.9997$ and $T_b = 2.0$, the parameter w is optimized in the interval $[0.00, 1.00]$ with step length 0.01, and the experimental results are shown in Fig. 9. The overall trend of the curve is obvious in spite of some fluctuations, and the optimal value is roughly between 0.5 and 0.6. Accordingly, the parameter w is set to 0.55 ($mI = 146.26$), which means the weight of the old direction is roughly the same as that of the new direction.

Finally, the optimization for parameter T_b is studied. Under conditions of $P_{ini} = 0.9997$ and $w = 0.55$, the parameter T_b is optimized in the interval $[0.1, 10.0]$ with step length 0.1, as is presented in Fig. 10. According to the trend of the curve, the optimal value of parameter T_b is around 2.0, which is

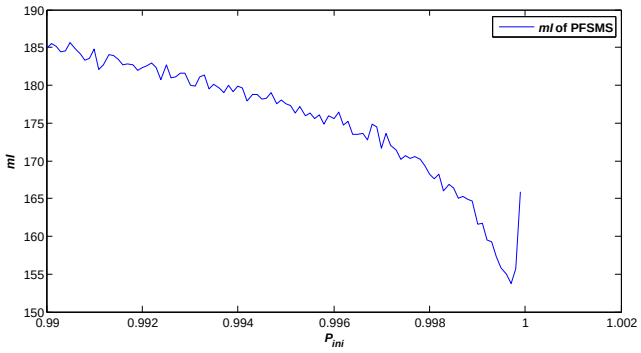

Figure 8: mI of PFSMS at different P_{ini}, which varies within $[0.9900, 0.9999]$ with step length 0.0001. mI denotes the mean of the number of iterations.

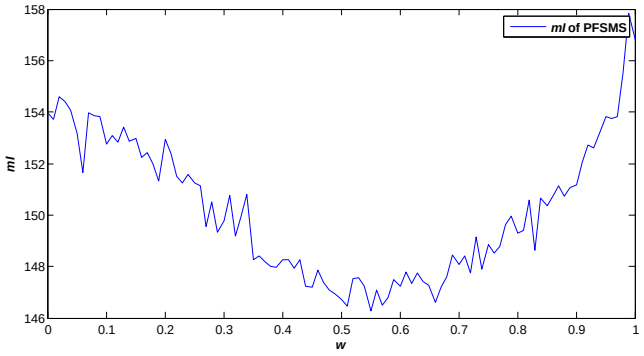

Figure 9: mI of PFSMS at different w, which varies within $[0.00, 1.00]$ with step length 0.01. mI denotes the mean of the number of iterations.

set to 2.3 ($mI = 145.67$) in the following experiments. With $T_b = 2.3$, the robot will not go into the 'Diffusion' state if the number of its neighbors is no more than 2. If the number of neighbors is larger than 2, the diffusion probability increases as the number of neighbors increases.

5.3. The Performance Comparison of Various Search Strategies under Different Problem Settings.

In this part, three problem settings are used to study the performance comparison of various search strategies, including different population sizes, different numbers of targets, and different collection times of targets. Finally, the overall performance rankings of different search strategies are given.

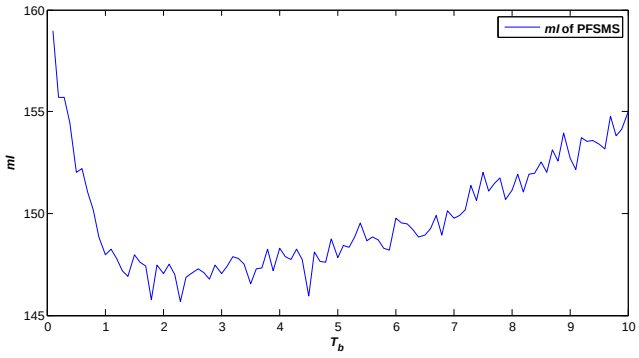

Figure 10: mI of PFSMS at different T_b, which varies within $[0.1, 10]$ with step length 0.1. mI denotes the mean of the number of iterations.

Table 1: mI and dI of search strategies with various population size and 10 targets. mI and dI denote the mean and standard deviation of the number of iterations respectively.

Population	*RPSO*		*A − RPSO*		*IGES*		*LFS*		*BMS*		*IS*		*TFS*		*PFSMS*	
	mI	*dI*	*mI*	*dI*	*mI*	*dI*	*mI*	*dI*	*mI*	*dI*	*mI*	*dI*	*mI*	*dI*	*mI*	*dI*
25	349.79	87.96	340.54	81.45	281.07	57.80	238.98	51.76	287.30	77.37	261.02	60.55	317.45	119.2	**186.43**	**38.20**
50	287.98	75.76	283.05	72.70	229.53	38.83	195.71	33.58	200.26	45.97	204.78	36.82	209.79	55.14	**147.40**	**20.05**
75	262.56	70.64	256.27	64.28	212.38	33.35	180.04	25.96	172.85	34.26	188.59	31.53	179.90	42.30	**138.24**	**17.78**
100	246.49	65.87	237.73	59.70	201.60	29.53	172.51	25.48	160.21	24.70	176.87	27.06	162.75	27.92	**132.53**	**15.95**
125	229.18	55.78	225.28	53.17	194.04	27.79	166.19	22.96	153.96	22.21	169.20	24.54	154.09	24.09	**129.34**	**15.00**
150	220.07	54.74	214.11	49.15	190.21	27.02	162.40	22.24	149.84	20.83	164.99	23.37	147.99	21.22	**126.91**	**14.79**
175	213.43	52.58	208.26	47.14	185.87	25.41	158.77	21.64	146.09	18.99	161.17	21.43	144.92	20.15	**125.27**	**14.70**
200	207.94	51.24	202.96	46.58	182.63	24.89	155.68	20.38	142.55	17.58	157.40	20.87	141.01	18.57	**123.45**	**14.39**

5.3.1. Different Population Sizes

In this section, the search efficiency of all comparison algorithms with various population sizes is investigated. Eight tests are carried out with 25, 50, 75, 100, 125, 150, 175, 200 robots in turn, and the map size is 1000*1000, containing 10 targets. The experimental results are presented in Tab. 1, Fig. 11 and Fig. 12.

The significance of the PFSMS is also tested. As is shown in Tab. 1, the mI and dI of PFSMS are obviously better than those of other strategies. In addition, for each population size, two-side Wilcoxon rank sum tests (with confidence level 99%) are conducted between the PFSMS and each of other strategies, which use the iterations of all 1000 runs. The result of the statistical test proves that, with every population size, the PFSMS is significantly better than each of other strategies.

We'd like to classify the search strategies into three groups: swarm optimization algorithms (RPSO, A-RPSO and IGES), independent search strategies (LFS, BMS and IS), and other search strategies (TFS and PFSMS).

As the results show, the efficiency of RPSO is the lowest among eight

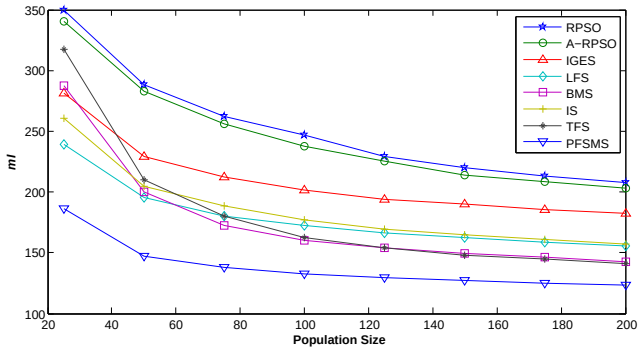

Figure 11: mI of search strategies with various population sizes and 10 targets. mI denotes the mean of the number of iterations.

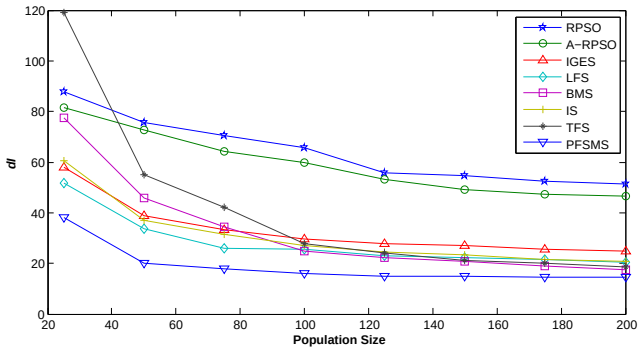

Figure 12: dI of search strategies with various population sizes and 10 targets. dI denotes the standard deviation of the number of iterations.

strategies, though we have introduced a random vector to improve its performance by avoiding local oscillation. A-RPSO also performs poor, though a simple niche technology (limiting the communication range) is introduced to divide the swarm. We can infer that traditional heuristic algorithms (such as PSO) for high dimensional optimization may not apply to the multi-target search task in swarm robotics, for the latter focuses on two or three-dimensional problem scenarios. In the problem of high dimensional optimization, the cooperation of a large number of individuals can improve the accuracy of gradient estimation while it is easy to calculate the gradient direction using a few robots in low dimensional cases, thus the attention is supposed to be paid to exploration instead of exploitation. Compared with RPSO and A-RPSO, IGES performs noticeably better, mainly resulting from its emphasis on the effect of diffusion mechanisms, which improves the ex-

ploration ability of the swarm. However, the diffusion action in IGES is a reactive decision, it can only ensure a limit distance among robots rather than further diffusion over the environment, so the exploration of the swarm is still limited.

An interesting result is that the independent search strategies perform better than RPSO, A-RPSO and IGES, which means the problem is simple enough to be tackled with random walk strategies integrated with technologies for gradient estimation. In areas with fitness values, the collaboration mechanisms of swarm optimization algorithms often lead to high degree of connectivity, for RPSO and A-RPSO tend to gather robots into a small area while robots in IGES tend to roughly maintain fixed relative positions, restricting the diffusion of the swarm. However, there is no cooperation among robots in independent search strategies, i.e. each individual makes the decision based on its own perception and history, thus there is no explicit mechanism to bring different individuals together. In addition, the introduction of the inertia mechanism to some extent improves the diffusion ability of the swarm. Therefore, independent search strategies lay emphasis on the exploration of the swarm, resulting in a relatively high efficiency.

Compared to LFS, the average step length of BMS is larger while that of IS is smaller due to the existence of phase 2 (fast motion), and it is worth pointing out that the phase 2 of IS is inefficient or even a drag in our problem model, without which IS could just behave as well as LFS. When the population size is small (such as 25), small step length will restrict significantly the exploration ability of the swarm while large ones will lead to missing nearby fitness information. As the population size becomes larger, the probability of missing nearby targets will decrease, so strategies with larger step length, such as BMS, will perform better. Another thing needed to be stated is that the 'IS' strategy in [26] is the equivalent of 'BMS' without inertia mechanism. As the BMS strategy has a good performance and easy to implement, its performance can serve as a benchmark in the multi-target search task.

As is shown in Fig. 11, the population size is an important factor in TFS. Since three-robot formation technology is adopted, the estimation for gradient direction is accurate, ensuring excellent exploitation ability, thus the main limiting factor is the exploration ability of the swarm. When the population is small, the number of three-robot teams is small, restricting the swarm exploration. Under such circumstances, some mechanisms for maintaining connectivity, such as mutual attraction or formation control,

may limit the exploration range of the swarm. As the number of robots increases, the performance of TFS improves obviously.

As for PFSMS, its performance is greatly superior to that of other strategies, of which the mI in the case of 200 robots (123.45) is close to the approximate lower bound of the problem (117.46). With a probabilistic diffusion mechanism, the PFSMS bears excellent exploration ability even with small population, and the improvement is not obvious anymore as the population is larger than 100, which means the PFSMS gives full play to the exploration ability of the swarm. Compared with IGES, the diffusion state in PFSMS could last for many iterations, giving a full play to the exploration of the swarm. With the aid of neighbor information, the triangle gradient estimation in PFSMS can obtain more accurate directions than those in independent search strategies, i.e. better exploitation ability, and the effect will be enhanced with larger population. In addition, the inertia mechanism can help robots to stabilize the moving direction and avoid some local oscillations, which mainly improved the exploitation ability of the swarm. When the population size is 50, compared with other seven strategies (RPSO, A-RPSO, IGES, LFS, BMS, IS and TFS), the efficiency of PFSMS increases 48.82%, 47.92%, 35.78%, 24.68%, 26.39%, 28.02% and 29.74% respectively, demonstrating significant performance improvement.

As is shown in Fig. 12, PFSMS also bears the best stability while RPSO and A-RPSO show the worst stability. If the individuals in a swarm tend to get together and keep strong connections with others, then it's hard to spread the effect of accidental factors, and that's why RPSO and A-RPSO are unstable. Therefore, the stability of algorithms partly reflects the exploration ability of the swarm. As the curves' tendencies of Fig. 11 and Fig. 12 are similar, it demonstrates that the exploration ability is critical in low-dimensional search. However, it is worth noting that stability is not equivalent to exploration ability, for the former can be ensured by a stable diffusion mechanism while the latter requires a stable and rapid diffusion mechanism. With the increase of population size, the stabilities of all algorithms are improved, especially for TFS and BMS. When the population is small (such as 25), the performance of TFS is extraordinarily unstable, for the three-robot formation mechanism intensifies the effect of accidental factors on the swarm. Although IGES shows similar stability to that of independent search strategies, its efficiency is still limited by the exploration rather than exploitation, because its stability mainly results from a stable diffusion instead of a rapid diffusion, which will be further illustrated in experiments with different collection times

Table 2: mI and dI of search strategies with various numbers of targets and 50 robots. mI and dI denote the mean and standard deviation of the number of iterations respectively.

Targets	$RPSO$		$A-RPSO$		$IGES$		LFS		BMS		IS		TFS		$PFSMS$	
	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI
1	96.03	36.06	98.20	39.44	105.35	42.69	89.04	31.72	92.12	37.64	94.48	35.64	94.66	46.28	**85.12**	**29.02**
5	187.12	45.79	187.97	47.15	179.70	34.61	150.77	30.72	156.19	42.51	161.16	33.52	167.54	56.61	**126.25**	**21.09**
10	292.92	76.46	281.70	71.66	228.38	38.24	196.20	32.86	200.68	46.73	205.32	36.17	212.29	56.64	**148.34**	**20.56**
15	380.26	95.80	364.65	89.08	272.52	43.66	227.43	34.00	227.65	43.32	235.46	37.81	241.80	58.71	**164.02**	**21.92**
20	437.02	94.88	414.89	85.19	301.63	50.29	245.69	36.41	249.48	44.73	254.42	40.11	261.79	62.05	**175.79**	**20.78**
30	544.09	104.16	506.65	92.64	355.33	65.66	282.03	41.43	283.73	42.50	289.45	46.31	306.15	67.58	**200.63**	**25.26**
40	627.32	109.12	560.62	93.80	408.77	63.98	309.42	40.24	313.54	45.85	316.92	46.80	348.19	76.09	**222.64**	**29.49**
50	712.58	122.77	606.38	107.38	436.35	67.35	329.20	41.70	338.02	45.91	334.33	48.41	367.50	71.15	**235.35**	**28.44**

of targets. As the swarm is getting larger to a certain size (such as 100), the dI of strategies tends to stabilize, which means the swarm is large enough to give a relatively full play to the exploration ability of various search strategies. With 50 robots, according to stability, the strategies can be sorted as $PFSMS > LFY > IS \approx IGES > BMS > TFS > A-RPSO \approx RPSO$.

5.3.2. Different Numbers of Targets

In this section, the search efficiency (i.e. mI) of all comparison algorithms with various numbers of targets is investigated. Eight tests are carried out with 1, 5, 10, 15, 20, 30, 40, 50 targets in turn, and the map size is 1000*1000, containing 50 robots. The experimental results are presented in Tab. 2, Fig. 13 and Fig. 14. It should be noted that the corresponding curve trend is expected to still be similar when the number of targets exceeds 50.

The significance of the PFSMS is also tested. As is shown in Tab. 2, the mI and dI of PFSMS are obviously better than those of other strategies. In addition, for each number of targets, two-side Wilcoxon rank sum tests (with confidence level 99%) are conducted between the PFSMS and each of other strategies, which use the iterations of all 1000 runs. The result of the statistical test proves that, with every number of targets, the PFSMS is significantly better than each of other strategies.

As is shown in the results (Tab. 2 and Fig. 13), when there is only one target, all search strategies have similar performance. One point to consider is that there may be multiple robots processing the target at the same time. To reduce the estimation error, the mean iterations for target processing can be assumed to be 5. According to the data in Tab. 2, the mean iterations (mI) of IGES, RPSO (A-RPSO/IS/TFS), LFS(BMS), PFSMS are around 105, 95, 90, 85, respectively. After subtracting 5 iterations for target processing, the mean iterations required for IGES, RPSO (A-RPSO/IS/TFS), LFS (BMS), and PFSMS are around 100, 90, 85, and 80, respectively. Calculated by

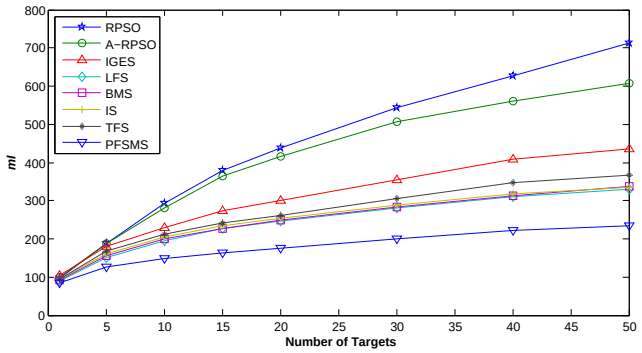

Figure 13: mI of search strategies with various numbers of targets and 50 robots. mI denotes the mean of the number of iterations.

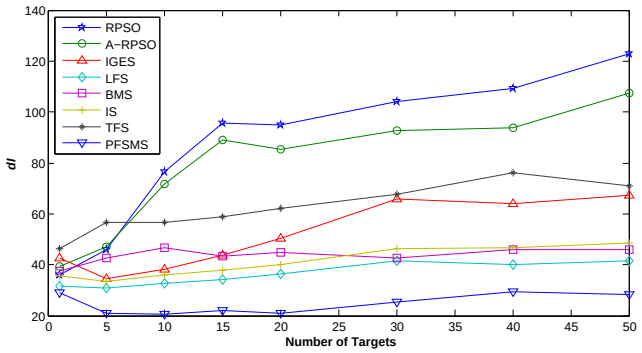

Figure 14: dI of search strategies with various numbers of targets and 50 robots. dI denotes the standard deviation of the number of iterations.

Eq. (16), when the target number is 1 (T=1), the approximate lower bound of the average number of iterations for search is 75, which shows that the PFSMS bears the closest performance, followed by LFS (BMS), RPSO (A-RPSO/IS/TFS) and IGES. In the task of finding and collecting one target, good exploration ability will help to find areas with fitness values where good exploitation ability will help to find the target. However, when there is only one target, the advantage of good exploration is not obvious, which gradually appears as the number of targets increases. Compared with RPSO and A-RPSO, the initial diffusion rate of IGES is limited by its diffusion mechanism, so its initial exploration ability is not good. All independent search strategies (LFS, BMS and IS) use the same triangle gradient estimation, so the minor performance difference mainly results from different exploration abilities (i.e. basically $LFS > BMS > IS$ with 50 robots). Although TFS has the best

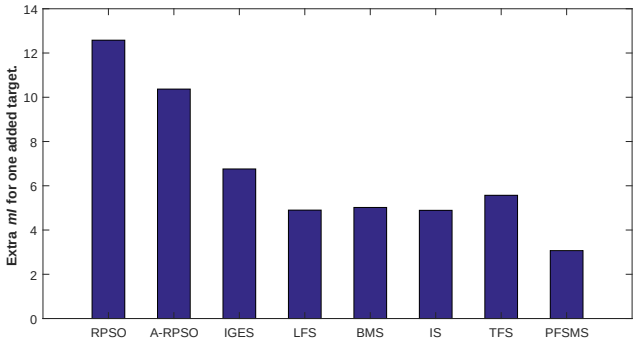

Figure 15: Extra mI required for one added target of different search strategies with 50 robots. mI denotes the mean of the number of iterations.

gradient estimation, its performance is restricted by its exploration ability. PFSMS shows the best performance, for its probabilistic diffusion mechanism contributes to a good balance between exploration and exploitation.

In Fig. 13, the slope of each curve represents the extra number of iterations required for one added target, so a smaller slope means the strategy has a better parallelism. For each strategy (RPSO, A-RPSO, IGES, LFS, BMS, IS, TFS and PFSMS), the slopes of the curves are 12.58, 10.37, 6.76, 4.90, 5.02, 4.89, 5.57 and 3.07 respectively, as is shown in Fig. 15. Compared with RPSO, the advantage of A-RPSO emerges as the number of targets becomes larger, which means that A-RPSO has better parallelism. All independent search strategies have similar parallelism performance, which are slightly better than that of TFS. According to parallelism, the strategies can be sorted as $PFSMS > IS \approx LFS \approx BMS > TFS > IGES > A-RPSO > RPSO$. Of all comparison algorithms, PFSMS shows the best parallelism, demonstrating its superior exploration ability.

The stability of strategies with various numbers of targets is shown in Fig. 14. With the number of targets increasing, the stabilities of IGES and PFSMS show a trend of fall-rise, which means a small amount of targets may help to reduce the randomness of the problem. Compared with others, the stabilities of TFS, independent search strategies and PFSMS are less susceptible to the number of targets, which can be seen from the trend of curves. On the other side, RPSO and A-RPSO are not good at handling multiple targets, for their stabilities go steeply worse as the number of targets increases.

Table 3: mI and dI of search strategies with various collection times of targets, 10 targets and 50 robots. mI and dI denote the mean and standard deviation of the number of iterations respectively.

Collection	$RPSO$		$A-RPSO$		$IGES$		LFS		BMS		IS		TFS		$PFSMS$	
Times	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI	mI	dI
1	278.26	73.79	264.94	67.55	210.64	36.68	178.35	29.75	187.05	46.06	186.57	31.78	196.83	55.73	**136.70**	**18.85**
5	286.28	78.65	275.86	71.66	220.97	37.49	186.81	31.74	193.84	45.38	197.16	35.35	206.82	55.58	**142.46**	**20.41**
10	289.05	76.14	283.17	69.65	230.61	37.84	194.82	33.69	199.97	46.53	205.24	36.71	210.67	59.39	**147.80**	**21.17**
15	294.24	76.35	287.63	70.85	236.11	40.73	203.22	35.39	208.66	47.04	212.06	37.48	215.33	54.90	**153.44**	**22.32**
20	299.69	77.26	291.80	69.82	244.18	43.32	210.55	37.59	215.39	46.42	221.17	41.32	220.66	57.12	**157.58**	**22.26**
30	306.46	75.91	304.07	72.66	256.42	43.66	223.62	39.77	224.53	45.21	231.56	41.47	227.49	57.14	**164.82**	**24.22**
40	312.79	75.37	313.91	74.53	268.03	47.28	234.03	42.43	236.79	48.40	244.35	47.19	235.67	58.00	**171.87**	**23.43**
50	321.18	78.63	321.42	74.21	277.77	47.89	243.62	42.55	247.31	49.99	254.01	43.96	244.89	56.09	**178.65**	**24.41**

5.3.3. Different Collection Times of Targets

As is stated in the problem model, it takes one robot 10 iterations to collect a target while one iteration for 10 robots, which is also used in the mathematical analysis. The 'Collection Time of Targets' is the number of iterations needed for one robot to collect one target. With different collection times of targets, we can study the cooperative collection ability of robots in different strategies, which is complementary to the parallelism ability. In this section, the search efficiency (i.e. mI) of all comparison algorithms with various collection times of targets is investigated. Eight tests are carried out with 1, 5, 10, 15, 20, 30, 40, 50 collection times of targets in turn, and the map size is 1000*1000, containing 10 targets and 50 robots. The experimental results are presented in Tab. 3, Fig. 16 and Fig. 17.

The significance of the PFSMS is also tested. As is shown in Tab. 3, the mI and dI of PFSMS are obviously better than those of other strategies. In addition, for each collection time of targets, two-side Wilcoxon rank sum tests (with confidence level 99%) are conducted between the PFSMS and each of other strategies, which use the iterations of all 1000 runs. The result of the statistical test proves that, with every collection time of targets, the PFSMS is significantly better than each of other strategies.

As is shown in Fig 16, the mI of various search strategies roughly satisfies a linear growth with the collection times increasing. With various collection times of targets, the efficiency rankings are not changed basically, except for RPSO and TFS, and the former gradually surpasses A-RPSO while the latter get superior to independent search strategies. The slope of curves is the extra number of iterations required for one added collection time, indicating the cooperative collection abilities of the swarm, so a smaller slope corresponds to a more efficient cooperative collection. For each strategy (RPSO, A-RPSO, IGES, LFS, BMS, IS, TFS and PFSMS), the average slopes of the

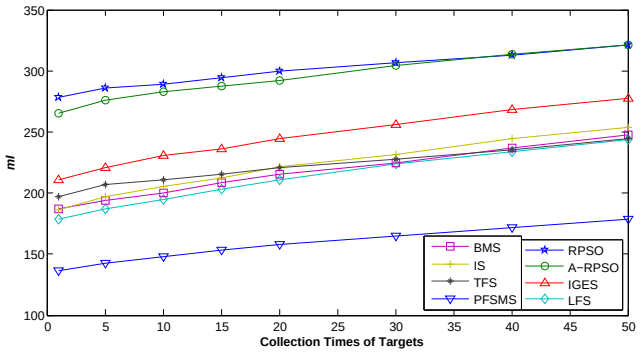

Figure 16: mI of search strategies with various collection times of targets, 10 targets and 50 robots. mI denotes the mean of the number of iterations.

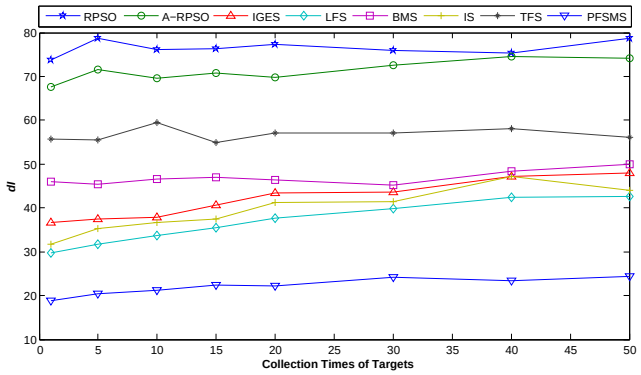

Figure 17: dI of search strategies with various collection times of targets, 10 targets and 50 robots. dI denotes the standard deviation of the number of iterations.

curves are 0.88, 1.15, 1.37, 1.33, 1.23, 1.38, 0.98 and 0.86 respectively, as is shown in Fig. 18. According to the ability of cooperation collection, the strategies can be sorted as $PFSMS \approx RPSO > TFS > A-RPSO > BMS > LFS \approx IGES \approx IS$. The cooperation collection ability of IGES is close to that of independent search strategies, which means they bear similar exploitation ability. So the search efficiency of IGES is mainly limited by its worse exploration, which has been noted previously in section 5.3.1.

Compared with the previous section, we can find that, in swarm optimization algorithms (RPSO, A-RPSO and IGES) the one with better parallelism usually has worse cooperative collection, which is the same in independent search strategies (LFS, BMS and IS). As for TFS, the three-robot formation technology leads to excellent ability of cooperative collection, and its initial

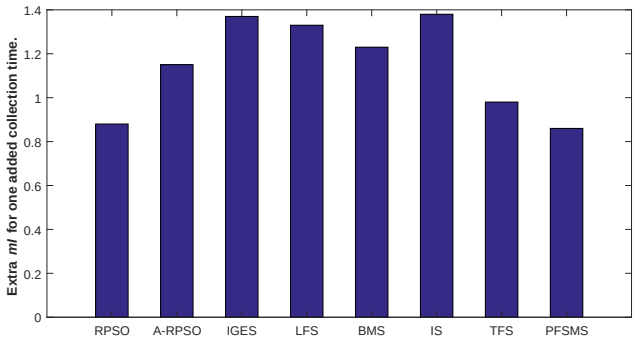

Figure 18: Extra mI required for one added collection time of different search strategies with 10 targets and 50 robots. mI denotes the mean of the number of iterations.

diffusion mechanism ensures a certain level of parallelism. Similar to TFS, the PFSMS also have a good trade-off between parallelism and cooperative collection, which is much more remarkable. Parallelism and cooperative collection could be regarded as one of the concrete forms of exploration and exploitation, so the experimental results could support the analysis and conclusions stated in the section 5.3.1 ("Different Population Size"). Since we focus on the search efficiency of strategies rather than target processing in this article, the collection times of targets is small (i.e. 10) and only linear acceleration is considered (i.e. the collection efficiency is proportional to the number of participating robots), so the efficiency improvement from good cooperative collection is not significant. However, if the iterations required for target processing is noticeably enlarged, things may work out differently.

As is shown in Fig. 17, the stabilities of various strategies are also insensitive to the collection times of targets, especially for TFS. With 50 robots, the stability rankings of search strategies are obvious, from which we can see PFSMS has the best stability, followed by independent search strategies and IGES, TFS, A-RPSO and RPSO. Compared with others, the stability of PFSMS is rather impressive, which is important to practical application.

5.3.4. The Performance Rankings of Different Search Strategies

In the previous parts, three groups of comparative experiments were carried out to study specifically the performance of search strategies under different problem settings (i.e. different population sizes, different numbers of targets, and different collection times of targets), and under each setting, the performance differences and detailed analysis and explanations are given.

Table 4: The performance rankings of search strategies under different problem settings.

	$RPSO$	$A-RPSO$	$IGES$	LFS	BMS	IS	TFS	$PFSMS$
Population	8.0	7.0	5.8	3.5	2.9	4.6	3.3	1.0
Targets	7.6	7.1	6.3	2.0	3.0	4.0	5.0	1.0
Collection Times	8.0	7.0	6.0	2.0	3.4	4.4	4.3	1.0
Total	7.9	7.0	6.0	2.5	3.1	4.3	4.2	1.0

Based on the data from Tab. 1, 2 and 3, this part aims to examine the overall performance of different search strategies from a macroscopic perspective, and to give the performance rankings of search strategies under different problem settings, as shown in Tab. 4.

In Tab. 4, the first row, the second row, and the third row are the average performance rankings of search strategies under different population sizes, different numbers of targets, and different collection times of targets, respectively; the fourth row shows the total performance rankings with all three settings considered. In all rows, the search strategies ranked first, sixth, seventh and eighth are fixed, namely PFSMS, IGES, A-RPSO, RPSO, highlighting the advantages of PFSMS and the disadvantages of swarm optimization algorithms. For the other four strategies (LFS/BMS/IS/TFS), LFS is dominant in the general population size (50 robots), but BMS and TFS are better choices when the population size is larger (as shown by the first row).

It should be pointed out that the performance ranking in Tab. 4 is a rough performance measure based on the above problem settings, and the performance in specific problems needs to be analyzed in detail, especially considering the size of the population.

6. Conclusions

In this paper, we establish an approximate mathematical model for an idealized multi-target search problem in swarm robotics, based on which we derive a lower bound of the expected number of iterations required to collect all targets. Combining random walk strategies and the triangle estimation technology, three independent search strategies (LFS, BMS and IS) are proposed, and the BMS can serve as a benchmark which is efficient and easy to implement. A novel probabilistic finite state machine based search strategy (PFSMS) is proposed, showing the highest efficiency and best stability in all comparison strategies. Experiments with different population sizes show that, with large number of robots (such as 200), the performance of PFSMS

is close to the approximate optimal value of the problem. Compared with other algorithms, the PFSMS can give a better play to the exploration of the swarm, improving greatly the search efficiency. Furthermore, experiments with various numbers of targets show that PFSMS has the best parallelism, and its excellent cooperative collection ability is also demonstrated by the experiments with different collection times of targets.

As for the future work, one research direction is the multi-target search problem in the strong interference environment, where the fitness distribution of targets may be greatly affected. Since the local exploitation ability of current search strategies depends directly or indirectly on the estimation of the fitness gradient, the performance of current strategies can be affected seriously in such environments. In addition, we'd like to study the problem with complex, especially structural obstacles (such as urban high-rise buildings, office environment, etc.), and to study some automatic approaches for strategy design, such as deep learning, reinforcement learning and evolutionary algorithms.

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. 2015CB352302.

[1] G. Beni, From swarm intelligence to swarm robotics, in: International Workshop on Swarm Robotics, Springer, 2004, pp. 1–9.

[2] G. Dudek, M. Jenkin, E. Milios, D. Wilkes, A taxonomy for swarm robots, in: Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on, Vol. 1, IEEE, 1993, pp. 441–447.

[3] L. Bayındır, A review of swarm robotics tasks, Neurocomputing 172 (2016) 292–321.

[4] M. Rubenstein, A. Cornejo, R. Nagpal, Programmable self-assembly in a thousand-robot swarm, Science 345 (6198) (2014) 795–799.

[5] E. Şahin, Swarm robotics: From sources of inspiration to domains of application, in: International workshop on swarm robotics, Springer, 2004, pp. 10–20.

[6] L. Bayindir, E. Şahin, A review of studies in swarm robotics, Turkish Journal of Electrical Engineering & Computer Sciences 15 (2) (2007) 115–147.

[7] I. Navarro, F. Matía, An introduction to swarm robotics, ISRN Robotics 2013.

[8] M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: a review from the swarm engineering perspective, Swarm Intelligence 7 (1) (2013) 1–41.

[9] Y. Tan, Handbook of research on design, control, and modeling of swarm robotics, IGI Global, 2015.

[10] M. F. Shlesinger, Mathematical physics: Search research, Nature 443 (7109) (2006) 281–282.

[11] J. Li, Y. Tan, The multi-target search problem with environmental restrictions in swarm robotics, in: Robotics and Biomimetics (ROBIO), 2014 IEEE International Conference on, IEEE, 2014, pp. 2685–2690.

[12] M. S. Couceiro, P. A. Vargas, R. P. Rocha, N. M. Ferreira, Benchmark of swarm robotics distributed techniques in a search task, Robotics and Autonomous Systems 62 (2) (2014) 200–213.

[13] M. Dadgar, S. Jafari, A. Hamzeh, A pso-based multi-robot cooperation method for target searching in unknown environments, Neurocomputing 177 (2016) 62–74.

[14] 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.

[15] 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.

[16] 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.

[17] 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.

[18] 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.

[19] 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.

[20] M. Z. Ali, N. H. Awad, P. N. Suganthan, R. G. Reynolds, A modified cultural algorithm with a balanced performance for the differential evolution frameworks, Knowledge-Based Systems 111 (2016) 73–86.

[21] M. Z. Ali, N. H. Awad, P. N. Suganthan, R. G. Reynolds, An adaptive multipopulation differential evolution with dynamic population reduction, IEEE transactions on cybernetics 47 (9) (2017) 2768–2779.

[22] G. M. Viswanathan, S. V. Buldyrev, S. Havlin, M. Da Luz, E. Raposo, H. E. Stanley, Optimizing the success of random searches, Nature 401 (6756) (1999) 911–914.

[23] F. Bartumeus, E. P. Raposo, G. M. Viswanathan, M. G. da Luz, Stochastic optimal foraging theory, in: Dispersal, individual movement and spatial ecology, Springer, 2013, pp. 3–32.

[24] G. M. Viswanathan, V. Afanasyev, S. Buldyrev, E. Murphy, P. Prince, H. E. Stanley, et al., Lévy flight search patterns of wandering albatrosses, Nature 381 (6581) (1996) 413–415.

[25] D. A. Raichlen, B. M. Wood, A. D. Gordon, A. Z. Mabulla, F. W. Marlowe, H. Pontzer, Evidence of lévy walk foraging patterns in human hunter–gatherers, Proceedings of the National Academy of Sciences 111 (2) (2014) 728–733.

[26] 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.

[27] J. Kennedy, Particle swarm optimization, in: Encyclopedia of machine learning, Springer, 2011, pp. 760–766.

[28] A. Darvishzadeh, B. Bhanu, Distributed multi-robot search in the real-world using modified particle swarm optimization, in: Proceedings of the Companion Publication of the 2014 Annual Conference on Genetic and Evolutionary Computation, ACM, 2014, pp. 169–170.

[29] K. Krishnanand, D. Ghose, Glowworm swarm based optimization algorithm for multimodal functions with collective robotics applications, Multiagent and Grid Systems 2 (3) (2006) 209–222.

[30] R. Akbari, A. Mohammadi, K. Ziarati, A novel bee swarm optimization algorithm for numerical function optimization, Communications in Nonlinear Science and Numerical Simulation 15 (10) (2010) 3142–3155.

[31] X. Yang, J. Yuan, J. Yuan, H. Mao, A modified particle swarm optimizer with dynamic adaptation, Applied Mathematics and Computation 189 (2) (2007) 1205–1213.

[32] Y. Tan, Y. Zhu, Fireworks algorithm for optimization, in: International Conference in Swarm Intelligence, Springer, 2010, pp. 355–364.

[33] 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. 246–251.

[34] F. Bartumeus, M. E. da Luz, G. Viswanathan, J. Catalan, Animal search strategies: a quantitative random-walk analysis, Ecology 86 (11) (2005) 3078–3087.

[35] G. M. Viswanathan, M. G. Da Luz, E. P. Raposo, H. E. Stanley, The physics of foraging: an introduction to random searches and biological encounters, Cambridge University Press, 2011.

[36] A. Dhariwal, G. S. Sukhatme, A. A. Requicha, Bacterium-inspired robots for environmental monitoring, in: Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on, Vol. 2, IEEE, 2004, pp. 1436–1443.

[37] D. W. Sims, E. J. Southall, N. E. Humphries, G. C. Hays, C. J. Bradshaw, J. W. Pitchford, A. James, M. Z. Ahmed, A. S. Brierley, M. A. Hindell, et al., Scaling laws of marine predator search behaviour, Nature 451 (7182) (2008) 1098–1102.

[38] O. Bénichou, C. Loverdo, M. Moreau, R. Voituriez, Intermittent search strategies, Reviews of Modern Physics 83 (1) (2011) 81.

[39] 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.

[40] U. A. Force, Air combat command manual 3-3, 1992.

[41] T. Balch, R. C. Arkin, Behavior-based formation control for multirobot teams, IEEE transactions on robotics and automation 14 (6) (1998) 926–939.

[42] T. R. Balch, R. C. Arkin, Motor schema-based formation control for multiagent robot teams., in: ICMAS, 1995, pp. 10–24.

[43] 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.