Group Explosion Strategy for Searching Multiple Targets using Swarm Robotic

Zhongyang Zheng and Ying Tan
Key Laboratory of Machine Perception and Intelligence (Peking University), Ministry of Education
Department of Machine Intelligence, School of Electronics Engineering and Computer Science,
Peking University, Beijing, China, 100871
email: gnbapsy@gmail.com, ytan@pku.edu.cn

Abstract—**In this paper, a group explosion strategy (GES) is proposed for searching multiple targets in obstructive environments using a swarm of simple robots. Considering the limited abilities of on-board sensors, fitness values detected by the robots are considered as discrete in the problem. The strategy introduces schemes from the explosion phenomenon in nature and the whole swarm is self-adaptively divided into small groups which search for targets independently. GES takes the advantages of quick convergence from intra-group cooperation as well as searching multiple targets in parallel from inter-group cooperation. The simulation results demonstrate that GES has great efficiency in energy consumption and targets collecting benefitting from cooperation among robots. GES also shows great stability in obstructive environments and large scale problems.**

Keywords*—*Swarm robotics, target search, explosion inspired, group search, obstructive environment, scalability.

I. Introduction

Swarm robotics has achieved significant progress benefiting from the development of artificial intelligent [1]. Many potential applications exist for the deployment of a swarm of robots [2], especially those require large amount of agents and time or dangerous to human being, e.g. foraging [3], surveillance, monitoring and search and rescue operations [4]. In general, these applications can be regarded as search-and-explore tasks in unknown environments. Therefore, searching strategies is an important challenge for swarm robotics researchers.

In most of the swarm robotics searching problems so far, some kinds of fitness functions are introduced for guiding the search of the swarm. These fitness functions are usually developed to measure the distance between robot and the target(s), such as adopting Euclidean distance directly [5], using olfaction measurements [6] or chemical clues [7] and some type of potential functions [8]. However, these functions are usually continuous and the robots can easily converge to the target position with gradient descent methods [9]. In real applications, on-board sensors of swarm robotics should be as cheap and simple as possible and may not detect the fitness values in such high accuracies limited abilities. Instead, sensors provide a limited number of rounded sensing results. Therefore, discrete fitness functions, rather than continuous one, are taken into consideration in this paper.

Controlling a swarm of robots is still a challenge in the robotic area despite its fast development [10]. Robots in the swarm should have as limited functions or abilities as possible, including motion ability, energy storage, sensing, communication, and computation capability due to their size, power constraints, cost and maintenance issues. Thus, cooperation plays the most important role in the swarm robotics control strategies to distribute and share resources across the swarm to complete the task [5]. PSO [11], inspired from birds flocking, is the most common swarm intelligence algorithm introduced for motivating swarm robotics for its simplicity and similarity with real robots [12]. Doctor et al. [13] are one of the first to use PSO for multi-robot searching though they mainly focus on optimizing the model parameters. Pugh and Martinoli [14] and their follow-up work [15] designed an effective searching algorithm inspired from PSO modified with various topologies. Hereford et al. [16] developed a distributed particle swarm optimization algorithm and used it for real robots in a physically-embedded version. Xue et al. [17] presented their PSO application for robots in target searching with a parallel asynchronous control strategy.

However, there still remain many problems when adopting PSO in swarm robotics searching applications, such as disadvantages of PSO as well as differences between optimization problems and searching applications which cannot be neglected. These problems can be named: large amount of random movements, trapping in local minimal, speed limitation [18] and others. To solve these problems, some researchers divide the swarm into sub groups for better cooperation to accelerate the searching progress. Xue et al. [19] introduced a mechanism for predicating target positions using information from at least three neighbours. Couceiro et al. [20] proposed a RDPSO that involves dynamic sub-grouping of the whole swarm. However, the main problem of this research is that the robots in the swarm require global communication for arranging subgroups, which is normally unavailable for large-scale outdoor applications. Therefore, new strategies capable of solving these problems should be proposed.

In recent years, many researches focused on introducing new schemes into swarm intelligence from the nature, including biological such as bacteria colonies [21] and whales [22]; non-biological such as improvisation process of musicians [23] and firework explosions [24]. Such schemes can also be adopted into swarm robotics.

Inspired from the explosion phenomenon in nature, a swarm robotics searching strategy, referred as “group explosion strategy”, is proposed in this paper. In the proposed method, the entire swarm is divided into sub groups in a self-organizing way and each group searches for the targets independently. Robots maintain the group structure and may split into smaller groups during the search. The strategy provides both intra-group cooperation and inter-group cooperation within the swarm. The sub-grouping strategy can overcome the problems mentioned above under limited sensing constraints. The searching problem considered in this paper is introduced in Section II. Section III describes the proposed method in detail, the experimental results and discussions are presented in IV. Finally, Section V concludes work in this paper.

II. Problem Statement

In this paper, a swarm of robots is applied to solve the problem of searching multiple targets in obstructive environment. The swarm has no prior information about the environment and targets. The problem is simulated in a computer program and time is divided into discrete iterations. Every iteration, each individual first retrieve information from environment and their neighbours, then move according to the sensing results. Maximum speed of robots is restricted so that robots' movements are guaranteed to complete before next iteration which is not too long. In real-life applications, all robots in the swarm are not restricted to share the same iteration cycle.

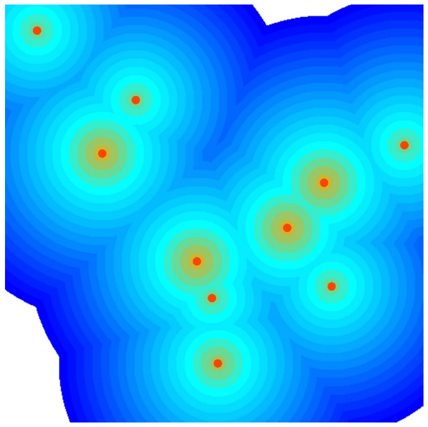


Fig. 1: A screenshot of the problem at the beginning of the simulation. Red circles stand for the targets. The background color illustrates fitness of that position. Robots and obstacles are not illustrated in this figure.

The problem is quite similar with the searching problems in other researches, such as [10], [19], [20], except the fitness values detected by the robots are discrete. The details of the problem are defined as follows. There exists m targets in the environment and information of a target can only be sensed as fitness values, which are discrete values inversely proportional to the distance from the target, as shown in Figure 1. The aim of this problem is to search and collect the targets. It takes 10 iterations for an individual to collect a target and the cooperation of multiple individuals collecting one target in the same time can accelerate this progress.

Each target has a randomly generated fitness ranges from F_{Max}-2 to F_{Max}, where F_{Max} is a predefined constant and is set to 20 in our experiments. Considering sensitivity and errors in real-time application, fitness values sensed by the robots are discrete values ranges from F_{Max} to 0. A target disappears when it is collected by the swarm and its fitness can be sensed no more.

Each target is shaped as a circle with a radius of $Size_t$. The target is identified as found if a robot overlaps with the ring and the robot can start collect the target at the position. A ring with a same radius of $Size_t$ outside the target has the same fitness as the target. The fitness reduces by 1 when the distance from the target is increased by $2Size_t$ until the fitness drops to 0. This indicates that fitness values are shaped as a ring with width of $2Size_t$ with increased radius and reduced fitness value. When fitness of several targets overlap, the largest value is adopted. Value of constant $Size_t$ is set to 10 in this paper.

A swarm of n autonomous mobile robots is used to solve this problem. The robots are designed to be as simple as possible are modeled as squares able to move freely in environment. The swarm has no leader or unique IDs and share no common coordinate systems nor global position systems. Each robot can sense the fitness of its current position and has a limited sensing range to detect the relative positions of their neighbour robots. They have a limited memory of past states of themselves. Each robot normally does not communicate explicitly with other robots except when sharing current fitness to their neighbours. Each individual executes the same algorithm but acts independently and asynchronously from others.

Each robot has the ability to sense neighbour robots within the range of $4Size_t$. Since no global positioning system is available, the positions are relative which can be detected without direct communications with the help of an infrared sensor and angle transducer. If F_{Max} is quite small, the robots can detect neighbors' fitness values without direct communication through colored lights equipped on the robot. Otherwise, just like the situation in this paper, robots share their fitness values to all their neighbors through direct communications or other strategies which is not focused in this paper.

The robots have a maximum speed of $2Size_t$ per iteration so that they can past a fitness grade in one iteration at the maximum speed and react quickly to the fitness changes if they are searching at the right directions. The individuals also have the ability to maintain last 10 history states including past position and the corresponding fitness values. Positions in the history are relative positions updated according to the local coordinating system of the robot. Past states cannot be shared among the swarm since complex communications and localizations are required for sharing the past states.

Static obstacles are also introduced in the problem. Each obstacle is regarded as a square with different sizes. Collisions are detected every iteration in the simulation and any robots that run into an obstacle are considered as broken and will be removed from the swarm permanently while the obstacle remains still in the environment.

III. Group Explosion Strategy

A. *Introducing Explosion Schemes into Swarm Robotics*

In this section, the group explosion strategy (GES) designed for searching multiple targets is explained in detail. In GES model, the whole swarm is divided into several groups. Robots in a group are spatially adjacent, i.e. within the sensing range of each other. Different groups do not intersect directly and their search for targets is parallel and independent. However, through certain strategies, groups that

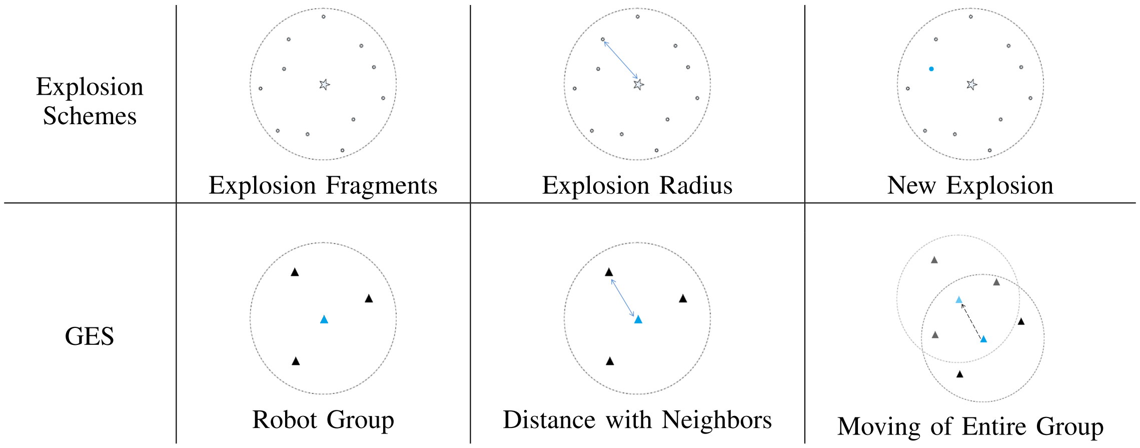


Fig. 2: Introducing explosion schemes into swarm robotics

run into each other will be re-arranged into new groups with possibly different members and search directions. In this way, inter-group cooperation can emerge in the swarm.

Explosion schemes are introduced into GES for searching multiple targets. The comparison of an explosion process and the corresponding actions in GES is shown in Figure 2 which has some similarities with the firework algorithm for optimization problems in [24], since both algorithms are inspired from explosion phenomenon. An explosion starts from a point and generates several fragments with different distances around the initial point. The center point can be regarded as a robot and the fragments exploded are the neighbour robots. These robots as a whole aggregates into a group, randomly distributed around the center robot just like the fragments. Each iteration, robots process sensing data and decide their movements. In this way a new explosion center is selected and the group explodes again in the next iteration to search near the new center.

B. Overview of Group Explosion Strategy

A group of robots can converge to the target much more quickly than a single robot with the help of intra-group cooperation, since the trends of the fitness in the environment are clearer within the group than a single individual. The more robots in a group, the quicker the group can converge to a target yet resulting in fewer targets the swarm is searching simultaneously since the number of groups is reduced. If the sizes of groups become too large, searching efficiency of the entire swarm declines instead. Therefore, a balanced group size should be adopted and the swarm can thus take the advantage of quick convergence from intra-group cooperation as well as searching several targets in parallel as inter-group cooperation. With a carefully designed strategy, the swarm can search and collect the targets more efficiently.

In GES model, robots first retrieve information from the environment and their neighbors, then calculate their new movements of this iteration (referred as velocity) according to the sensing data and carry out the move before next iteration. The state of this iteration is stored in history before robots actually move towards the new positions. In this way, the history of each robot contains 10 past states of the robots excluding their current states.

Velocity of robot i at every iteration consists two components: grouping component $G(i)$ and history component $H(i)$. $G(i)$ controls the robots' behavior relevant to grouping and is calculated according to the current fitness of the robot and states of its neighbour including their relative positions and fitness values. $H(i)$ is computed from the past history states stored in the robot.

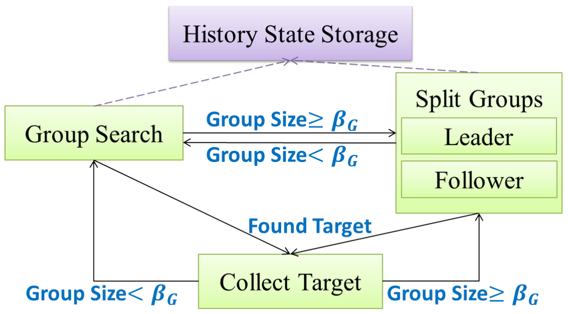


Fig. 3: Flow chart of GES

The grouping behavior of a robot differs with the size of its current group, i.e. the number of robots in its neighbourhood. Group size is controlled by a pre-defined threshold β_G. When the size exceeds the threshold, robots in the group try to split this group into two smaller ones to balance the search efficiency. Otherwise, robots try to maintain the group and take advantage of the information shared in the group to benefit searching progress.

A brief flow chart of the GES strategy is shown in Figure 3. Each robot is regarded as a finite state machine with three states: group search, split groups and collect target. Expressions of $G(i)$ for these two situations are explained in Section III-C and III-D respectively, Section III-E gives the expression of $H(i)$ and the final velocity update equation is presented in Section III-F. In collect state, robot stays still at its position until the target is collected and goes back to the two searching state according to the size of the group. Several robots collecting same target can accelerate this process as mentioned in previous section.

C. Group Search

When the size of group is beyond the threshold β_G, the group tries to search for a target through intra-group cooperation. Following the schemes from explosion, searching in next iteration should take place in the area around the current best position found in the group. Therefore the searching strategy of the group is to move the group center towards the best position within the group. The best position is selected as the current position of the robot with highest fitness value among all the robots in the group. Historical positions of any robots are not taken into account, since the communication overloads for exchanging historical states is too large. In this way, the group should steadily converge to a target much quicker than only one robot.

The group center is calculated as the centroid of all the robots in the group. The group center $C(i)$ for robot i is calculated as following:

$$C(i) = \frac{\sum_{j \in N(i)} P(j) + P(i)}{|N(i)| + 1} \tag{1}$$

where $P(i)$ is the current position of robot i and $N(i)$ is the collection consists of all neighbour robots of robot i, i.e. all other robots within the group.

It can be easily seen from (1) that although robots do not exchange their positions with each other directly, they can calculate the same centroid of the group if no noise and error is considered. Even with little noise and error from the sensors, the result could be quite similar and should not affect the cooperative scheme.

Given the group center, the robot can compute the grouping component $G(i)$ used in the velocity update equation. Before calculating, the robots in the group should exchange the fitness information among the group with the help of direct communication or colored lights due to the hardware design of the robots. $G(i)$ is computed using (2).

$$G(i) = (P(b) - C(i)) * R_S \tag{2}$$

where robot b is the robot with the best fitness in the group and R_S is a scaling factor randomly selected from three values with equal possibilities: $1 - \beta_S$, 1 and $1 + \beta_S$. β_S is used to adjust the shape of the group by controlling the distance of robots from the group center. Robots with various distances from the center can provide the group with great diversity and meaningful feedbacks in different scales for choosing the searching direction.

It should be noted in GES that, robots do not explicitly tell their neighbours which robot has the best fitness. Instead, they exchange their fitness with all their neighbours. Each robot computes the best robot according to the fitness values it receives independently. Therefore, robots in the group may choose different best robot if noise or error occurs. However, the sensing distance is twice the time of the maximum speed of the robot, so it could be possible that the robot still remains in the group. If a robot goes out of the group, it will start searching the targets itself and rejoin a group if encounters one. The most possible place for rejoining a group is near a target, since robots always try to converge into a target even if a single robot without group.

D. Split Groups

When group size exceeds the threshold β_G, the strategy for the robots in the group is splitting the large group into two smaller ones. Robots with the best two fitness, denoted as $L1$ and $L2$, are selected as two leaders for the new groups. Without loss of generality, it is assumed that fitness of $L1$ is not worse than $L2$. To separate the two groups apart, two opposite directions are selected for the new groups. Two leaders repulse with each other and try to be as far away as possible. The repulsive vector is calculated in (3).

$$\begin{aligned} R(L1) &= (P(L1) - P(L2)) * \beta_R \\ R(L2) &= (P(L2) - P(L1)) * \beta_R \end{aligned} \tag{3}$$

where β_R is a pre-defined coefficient for repulsing the two leaders $L1$ and $L2$.

As for other robots in the group, each of them selects a leader to follow independently and randomly. Each of $L1$ and $L2$ is given a weight for selecting, the higher the weight, the more change a leader is selected since a leader with higher fitness is more possible to find a target. Besides, the leaders are repulse to each other and thus the robot $L1$ is quite possibly being repulsed towards a target. The weights of two leaders are assigned based on their fitness in (4) and the possibility for selecting leaders are calculated in (5).

$$w(L1) = F(L1) - F(L2) + \beta_P,\ w(L2) = \beta_P \tag{4}$$

$$P_{L1} = \frac{w(L1)}{w(L1) + w(L2)},\ P_{L2} = \frac{w(L2)}{w(L1) + w(L2)} \tag{5}$$

where function F indicates the current fitness of the robot, $w(L1)$, $w(L2)$, P_{L1} and P_{L2} are the weights and possibilities of two leaders respectively. β_P is a constant to balance the weights and fixed to 1 in this paper.

Since the sensing range of a robot is double of size of a fitness value can occupy, so the value of $F(L1) - F(L2)$ is restricted to be zero or one. This guarantees the difference between the two weights is not too large that makes the new group sizes very unbalanced.

The grouping components of the leaders and other robots can now calculated using (6).

$$G(i) = \begin{cases} R(i) & , i = L1, L2 \\ R(i) + (P(l) - P(i)) * R_S, & \text{otherwise} \end{cases} \tag{6}$$

where l is the leader robot i has selected. The leaders just repulse each other to separate the groups. Other robots use the same repulse vector with the leader and the new groups move apart as a whole to separate with each other. Besides separating, distances between robots and their leaders are also changed by the factor R_S which is the same as mentioned in previous section.

E. Utilize History States

History component $H(i)$ is independent with the grouping situations and is computed based only on the stored states in robot's history. In GES, only ten latest history states (position and fitness) are maintained. When robots search in a wrong direction and depart the targets (fitness observed is decreasing), they can get back to route if making full use of the history components as computed in (7).

$$H(i) = (P(i) - h(i)) * r \tag{7}$$

where $h(i)$ is the position of the history state with the best fitness and r is a random number uniformly distributed within the range $[0.4, 0.8]$.

If several history states share the same fitness, the most recent position is selected. Note that when counting $h(i)$, current position is also taken into account to make sure the robot is not attracted by a worse position. Thus, the history component is set to be 0 if fitness of current position is better than all the states in history and will not contribute to the velocity update of the robot as the history state can provide no positive information to guide the search.

F. Velocity Update Equation

After calculating the two components $G(i)$ and $H(i)$, the velocity of robot i at iteration t, denoted as $V_t(i)$, is updated as following.

$$V_t(i) = \begin{cases} G(i) + H(i), \ \|G(i)\| > 0 \\ H(i) + R_p \ , \ \|G(i)\| = 0 \land \|H(i)\| > 0 \\ V_{t-1}(i) \quad , \ \|G(i)\| = 0 \land \|H(i)\| = 0 \end{cases} \tag{8}$$

where R_P is a randomly generated unit vector and $V_{t-1}(i)$ is the velocity of the last iteration.

In the equation, velocity update strategy is different if $\|G(i)\|$ is 0. Since no grouping action is taking place when $\|G(i)\| = 0$, only the history component remains in the equation. This will lead to a vibration around the history best position if the robot gets stuck in an area with same fitness value. A small random vector R_P is introduced to avoid such situation. R_P is an unit vector while $H(i)$ is normally quite large so the movement of the robot is not too stochastic.

The situation that both $G(i)$ and $H(i)$ are 0 usually occurs when the robot is the best of the group and fitness is improving in recent iterations, so the robot just remains its searching direction as the previous iteration.

G. Obstacle Avoidance

Obstacles are also considered in GES. Obstacle avoidance is computed stand alone after the velocity update and is not used in the $V_{t-1}(i)$ in (8). Since obstacle avoidance is not the main concern of this paper, a simple avoiding scheme is used for both GES and the baseline algorithm (introduced in Section IV-A). Each robot checks if it will run into any obstacles with the velocity $V_t(i)$. If so, it adds a small repulsive force perpendicular to $V_t(i)$ from the obstacle to make sure a collision will not happen. This simple scheme can provide acceptable performance for obstacle avoidance from the simulation results in Section IV-E.

IV. Simulation Results and Discussions

In this section, comparison between GES and a searching algorithm inspired from standard PSO is presented. We first introduce the baseline algorithm briefly and present the simulation results of several experiments. Two methods are simulated in a self-built simulation platform [25] and tested under different situations. The first experiment is to validate the GES to see if it's capable for solving the problem considered in this paper. Various population sizes and number of targets are considered in this experiment. In the second experiment, the swarms are supposed to collect as much targets as possible under a time constraint. The third experiment is the scalable experiment which has the same condition with the first one except the scales of swarm size, number of target and map sizes are enlarged. Obstacles are introduced in the last experiment and results are compared with the same situation without obstacles.

In most of the experiments, the stop criteria are to collect a certain percent of the targets and the essential standard for judging the performance is the total iteration used. For the problem definition, it can be easily seen that fitness becomes inadequate as large areas with 0 fitness appear in the environment as targets are collected or in large maps. This means the search of a swarm becomes harder as the experiment goes.

All the experiments in this section use the same environment setup: 20 randomly generated maps are used and each method is repeated for 20 times in the map of 500*500 sizes. Average results of these 400 runs are presented in results. Parameters of the two methods are tuned in advance with 10 robots and 20 targets.

A. Baseline Algorithms

In the experiment of this paper, the baseline algorithm uses the same strategy of RPSO in [20]. Each robot acts as a particle and the spacial-based topology of the robots for calculating gbest is adopted. However, RPSO requires a large communication range for the swarm which is restricted in this paper, thus a little modification is introduced. In case the robot vibrates in an area, the small random vector R_P in GES is introduced if both pbest and gbest are the current position. Except the velocity update strategy, baseline algorithm shares the same scheme with GES for better comparison, such as obstacle avoiding and history state updating.

B. Validation Experiment

The first experiment validates the GES method and compares the iterations used for two methods to collect half and all the targets in the environment respectively. The experiment is simulated in a small scale setup: map size is 500*500, population n ranges from 5 to 25 and number of targets m varies from 10 to 40.

The results are shown in Table I. In the table, results for collecting half targets and all targets are presented in different columns. "Iteration" stands for the iterations used to collect the targets and is the most important criterion to evaluate the performance. "Collect" stands for the targets collected in the "Half Target" situation as several targets may be collected in the same iteration. "Distance" stands for the averaged moving distance for each robot in the entire searching process and "Time" indicates the average simulation time on PC in milliseconds which is used to evaluate the computation overload. The last two columns indicate the ratio of iterations between GES and RPSO.

TABLE I: Validation Results

m	n	GES					RPSO					GES / RPSO	
		Half Target		All Target			Half Target		All Target			Iteration	
		Iteration	Collect	Iteration	Distance	Time	Iteration	Collect	Iteration	Distance	Time	Half	Full
	5	**162.62**	5.00	**369.660**	3285.164	**238.24**	178.71	5.00	763.670	**3184.458**	243.58	**91.00%**	**48.41%**
	10	**133.35**	**5.02**	**271.088**	**2414.354**	**242.39**	145.05	5.01	558.915	2456.718	250.85	**91.93%**	**48.50%**
10	15	**124.10**	**5.02**	**233.105**	**2066.172**	**246.37**	132.19	5.01	487.398	2200.033	259.21	**93.88%**	**47.83%**
	20	**123.36**	**5.04**	**215.855**	**1905.821**	**252.74**	124.11	5.01	457.593	2094.180	266.80	**99.39%**	**47.17%**
	25	125.66	**5.03**	**206.813**	**1819.752**	**259.88**	**122.33**	5.01	424.483	1973.483	275.49	102.72%	**48.72%**
	5	**273.80**	10.01	**574.055**	4951.138	**992.89**	277.13	**10.02**	826.823	**4151.130**	997.10	**98.80%**	**69.43%**
	10	**191.51**	**10.03**	**388.220**	3372.749	**1000.11**	226.72	10.02	600.575	**3335.970**	1000.80	**84.47%**	**64.64%**
20	15	**159.62**	**10.05**	**318.800**	**2774.251**	**1001.23**	201.78	10.01	529.613	3054.101	1013.95	**79.11%**	**60.19%**
	20	**152.44**	**10.06**	**285.258**	**2481.513**	**1010.11**	190.34	10.01	487.583	2885.941	1024.92	**80.09%**	**58.50%**
	25	**155.09**	**10.06**	**273.293**	**2378.512**	**1016.96**	185.64	10.01	471.033	2822.598	1037.37	**83.54%**	**58.02%**
	5	339.77	15.02	**728.570**	6167.176	2264.98	**336.67**	**15.03**	972.588	**5095.470**	**2256.03**	100.92%	**74.91%**
	10	**217.53**	**15.02**	**457.145**	**3891.138**	**2267.21**	255.58	15.01	676.360	3934.549	2273.14	**85.11%**	**67.59%**
30	15	**188.08**	**15.04**	**373.935**	**3202.253**	**2268.14**	237.07	15.02	595.553	3657.545	2275.55	**79.33%**	**62.79%**
	20	**171.23**	**15.05**	**327.325**	**2811.040**	**2272.69**	221.35	15.02	550.285	3479.020	2293.55	**77.36%**	**59.48%**
	25	**175.31**	**15.06**	**315.625**	**2723.570**	**2291.36**	203.65	15.05	507.245	3260.953	2307.78	**86.08%**	**62.22%**
	5	412.04	20.02	**887.593**	7417.298	4038.25	**381.80**	**20.02**	1132.458	**5993.424**	**4033.30**	107.92%	**78.38%**
	10	**254.63**	**20.04**	**544.280**	**4571.844**	**4035.46**	299.73	20.03	813.088	4775.093	4054.15	**84.95%**	**66.94%**
40	15	**207.65**	**20.03**	**430.508**	**3636.355**	**4054.05**	276.99	20.02	696.380	4378.265	4073.41	**74.97%**	**61.82%**
	20	**195.73**	**20.06**	**380.403**	**3232.400**	**4061.64**	259.55	20.02	667.423	4306.407	4082.21	**75.41%**	**57.00%**
	25	**192.42**	**20.08**	**354.113**	**3021.856**	**4075.64**	245.51	20.03	618.000	4020.028	4088.93	**78.37%**	**57.30%**

From the table, it can be easily seen that GES dominates RPSO in the performance regardless of population and number of targets, especially when collecting all targets. GES can complete most of the missions with only 70-90% and 50-70% of the iterations than that of RPSO in "Half" and "All" situations respectively. It also shows that GES has a smaller total moving distance which means GES is more energy efficient than RPSO. Considering the smaller iterations, GES utilizes the maximum speed much better than RPSO which is one of the important reasons why GES has a better efficiency. In the table, GES also has a shorter time which indicates the GES strategy requires less computation resources than RPSO. This is important for swarm robotics applications since the robots normally do not have powerful computation abilities.

The results also show that GES outperforms RPSO more when population size is larger. This indicates the strategy introduced in GES shows great ability in cooperation among robots and can accelerate the searching process when there are more robots in the environment. When number of targets increases, the average iteration GES used to collect one target decreases quicker than RPSO which also indicates the strategy is taking full advantage of cooperation among robots.

C. Experiment Under Time Constraint

The second experiment is simulated in this section which requires the swarm to collect as many targets in a fixed number of iterations. The purpose of this experiment is to see how fast the swarm can converge when fitness information is adequate. Simulation stops after $5m + 100$ iterations, where m is the number of target. This number is set based on the results in previous section to make sure most of the searching is still in progress before reaching the predefined iteration. The results is shown in Table II, "Collect" in the table indicates the total number of collected targets in the simulation and is the main concern of this experiment.

Similar with the results of "Half Target" situation in previous section, GES outperforms RPSO in this test as well

TABLE II: Results Under Time Constraint

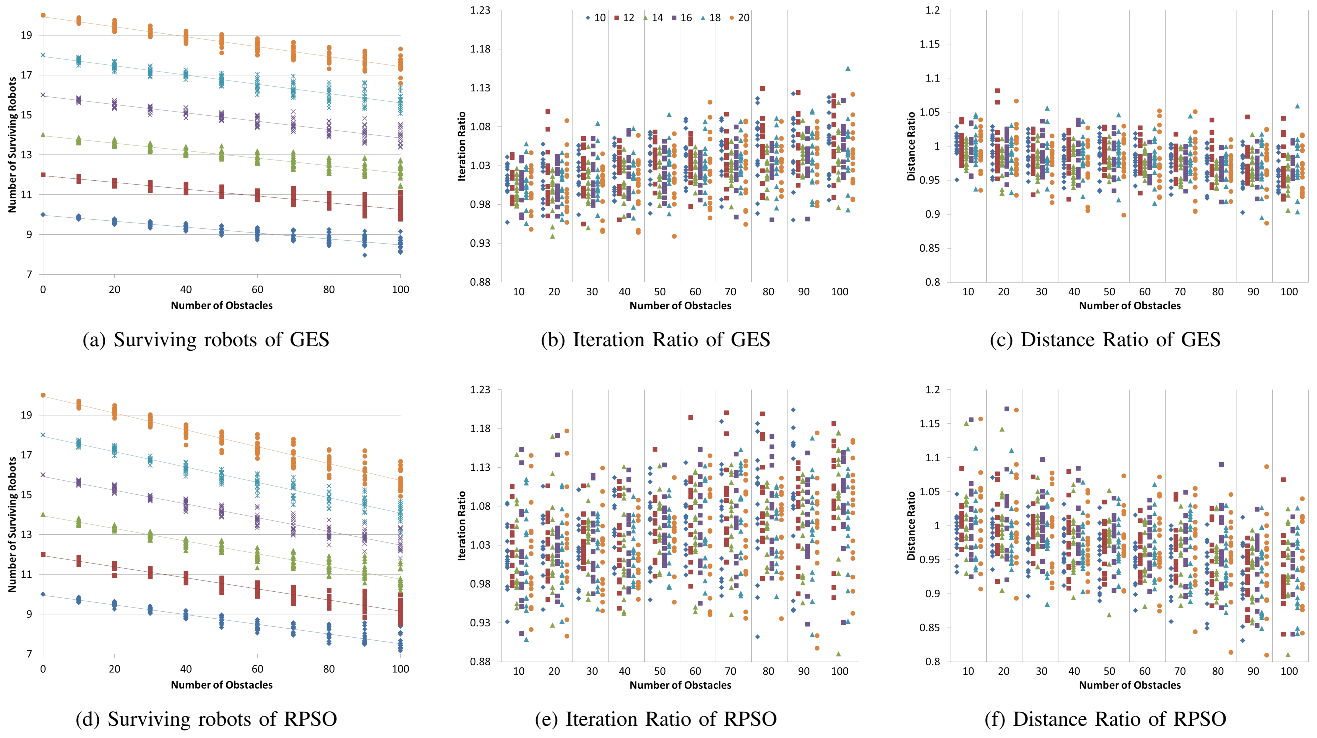
m	n	GES		RPSO		GES / RPSO
		Collect	Iterations	Collect	Iterations	Collect
	5	4.385	150.00	**4.823**	150.00	90.93%
	10	5.540	150.00	**5.585**	**149.98**	99.19%
10	15	**6.238**	**149.94**	5.843	149.97	**106.76%**
	20	**6.545**	**149.66**	6.175	149.90	**105.99%**
	25	**6.650**	**149.45**	6.260	149.94	**106.23%**
	5	7.808	200.00	**8.378**	200.00	93.20%
	10	**11.043**	200.00	10.640	200.00	**103.78%**
20	15	**12.905**	**199.90**	11.245	200.00	**114.76%**
	20	**13.363**	**199.87**	11.835	200.00	**112.91%**
	25	**13.630**	**199.66**	12.010	199.97	**113.49%**
	5	11.460	250.00	**12.448**	250.00	92.07%
	10	**17.538**	250.00	15.845	250.00	**110.68%**
30	15	**21.130**	250.00	17.225	250.00	**122.67%**
	20	**22.880**	**249.88**	18.303	249.96	**125.01%**
	25	**23.603**	**249.13**	19.095	249.69	**123.61%**
	5	15.005	300.00	**16.250**	300.00	92.34%
	10	**23.988**	300.00	21.635	300.00	**110.87%**
40	15	**29.858**	**299.99**	23.490	300.00	**127.11%**
	20	**32.750**	**299.38**	25.145	299.93	**130.24%**
	25	**33.225**	**298.09**	26.688	299.59	**124.50%**

and has the same trends when population and number of targets change. GES collects at least 10% more targets than RPSO within the same iterations. The advantage even becomes more than 20% when number of targets is large.

Comparing the results in this section with the previous section, it's obvious that GES shows more advantage in performance than RPSO especially when there are more targets or robots in the environment. In such situation, cooperation between robots is playing a more important role and GES achieved excellent performance. More detailed comparison will be shown in next section in the scalable experiment.

D. Scalable Experiment

The main aim of this experiment is to see if the proposed GES strategy can scale to problems with larger number of robots, targets or map sizes. In this experiment, environment


(a) Surviving robots of GES (b) Iteration Ratio of GES (c) Distance Ratio of GES

(d) Surviving robots of RPSO (e) Iteration Ratio of RPSO (f) Distance Ratio of RPSO

Fig. 5: The trends of three main criteria under obstructive environments with various setups of targets and populations. In the last two criteria, the ratio is regarded to the division by the result of the same strategy and setup in non-obstructive environment. Points with different colors indicate the results of different swarm populations, as shown in the legend.

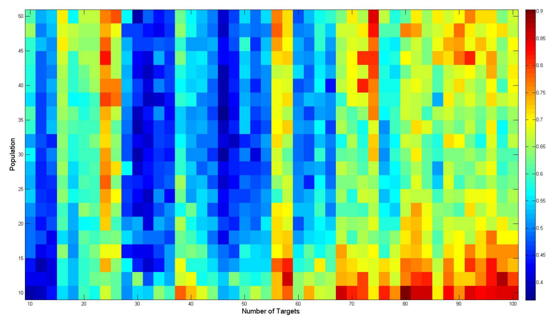


Fig. 4: Scalability results. The values indicate the iteration ratio for collecting 75% of the targets of GES and RPSO. GES outperforms more when the value is smaller.

is setup much bigger than previous experiments: map size is 1000*1000, n ranges from 10 to 50 and m varies from 10 to 100. Both methods use the same parameter with the previous experiments. The results are shown in Figure 4.

It can be easily seen that the trend of two algorithms are quite the same than in previous sections in Figure 4. GES has great advantages when population is large. It can also be induced that the advantages is not that large when number of targets is increased. The main reason is that the map size is not large enough so the environment is full of fitness values even when only 25% of targets remains. As shown in previous section, advantage of GES is smaller when fitness is adequate.

E. Obstacle Avoidance Experiment

In this experiment, we test the performance of the proposed algorithm in environments with small static obstacles. Obstacles are randomly distributed in the environment and the number ranges from 0 to 100. Performances in obstacle environments are compared with that of non-obstacle environment. The results are shown in Figure 5. In this experiment, the changes of the two numbers: population m and number of targets n are tested in a more conscientious way, with a step of 2 instead of 5 and 10 in previous experiments. Three main criteria are considered in this experiment: number of surviving robots, "Iteration", "Distance". The last one indicates the remaining robots after the simulation and is used to judge how many collisions were taking place. The stop criteria are set as collecting 75% of the targets.

From Figure 5a and 5d, it can be easily deduced that GES shows advantages in avoiding the obstacles than RPSO even with the same obstacle avoidance strategy. There are possibly two reasons for such results. The first one is that GES takes much less iterations than RPSO and thus has a lower possibility of encountering obstacles. The second one may be the cooperation schemes in GES. Although the strategy does not include a direct cooperation of obstacle avoidance, the swarm do benefit from the cooperation. The robots that successfully avoided any obstacles usually have lower fitness values in the group since they took a longer way round. Therefore, the groups will possibly move towards other directions and thus avoid the obstacles indirectly.

In Figure 5, the trends of “Iteration” and “Distance” are very similar for both GES and RPSO strategy. Robots take shorter moves per iteration to avoid more obstacles which leads to larger iterations. However, it’s obvious that GES has a much better stability than RPSO when the environment setups change, as the points in the figure of GES are much more compact. Stability can be also indicated from the trend of the two algorithms as RPSO is more sloped. If comparing the mean value of all the results with same number of obstacles, two strategies are quite the same when few obstacles exist, but GES becomes better when there are number of obstacles increases. Considering the big differences of “Iteration” and “Distance” from the results of previous experiments, GES definitely outperforms RPSO in environments with obstacles.

V. Conclusion

A grouped explosion strategy (GES) for searching multiple targets is proposed in this paper. GES method is applied to the multiple targets searching problem on a self-built simulation platform. The swarm searches and collects targets in the environment without prior knowledge. Several tests are run to evaluate how GES performs in various aspects including stability, robustness and flexibility. Simulation results demonstrate that GES shows great efficiency when fitness is either adequate or inadequate in the environment. GES also shows good stability in obstructive and large-scale environments. These results indicate that the GES strategy has great ability in cooperating robots to accomplish the tasks.

As for future work, we plan to improve the strategy through introducing direct inter-group cooperation. We will also try to apply GES on more complicated searching problems, such as dynamic environment with various kinds of targets which have different values to collect. Through modeling and analyzing the intra-group behaviors in GES, improved strategy may be proposed for more complicated problems.

Acknowledgements

This work is supported by the National Natural Science Foundation of China under grants number 61170057 and 60875080. Professor Ying Tan is the corresponding author.

References

[1] Q. Tang and P. Eberhard, “Cooperative motion of swarm mobile robots based on particle swarm optimization and multibody system dynamics,” *Mechanics based design of structures and machines*, vol. 39, no. 2, pp. 179–193, 2011.

[2] R. Grabowski, L. Navarro-Serment, and P. Khosla, “An army of small robots,” *Scientific American*, vol. 18, pp. 34–39, 2008.

[3] V. Gazi, B. Fidan, Y. Hanay, and M. Köksal, “Aggregation, foraging, and formation control of swarms with non-holonomic agents using potential functions and sliding mode techniques,” *Turk J Elec Engin*, vol. 15, no. 2, pp. 149–168, 2007.

[4] G. Kantor, S. Singh, R. Peterson, D. Rus, A. Das, V. Kumar, G. Pereira, and J. Spletzer, “Distributed search and rescue with robot and sensor teams,” in *Field and Service Robotics*. Springer, 2006, pp. 529–538.

[5] K. Derr and M. Manic, “Multi-robot, multi-target particle swarm optimization search in noisy wireless environments,” in *Human System Interactions, 2009. HSI’09. 2nd Conference on*. IEEE, 2009, pp. 81–86.

[6] A. Marjovi, J. Nunes, P. Sousa, R. Faria, and L. Marques, “An olfactory-based robot swarm navigation method,” in *Robotics and Automation (ICRA), 2010 IEEE International Conference on*. IEEE, 2010, pp. 4958–4963.

[7] W. Li, J. Farrell, S. Pang, and R. Arrieta, “Moth-inspired chemical plume tracing on an autonomous underwater vehicle,” *Robotics, IEEE Transactions on*, vol. 22, no. 2, pp. 292–307, 2006.

[8] H. Espitia and J. Sofrony, “Path planning of mobile robots using potential fields and swarms of brownian particles,” in *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 2011, pp. 123–129.

[9] T. Schmickl and K. Crailsheim, “Trophallaxis within a robotic swarm: bio-inspired communication among robots in a swarm,” *Autonomous Robots*, vol. 25, no. 1, pp. 171–188, 2008.

[10] Q. Tang and P. Eberhard, “A pso-based algorithm designed for a swarm of mobile robots,” *Structural and Multidisciplinary Optimization*, vol. 44, no. 4, pp. 483–498, 2011.

[11] R. Eberhart and J. Kennedy, “A new optimizer using particle swarm theory,” in *Micro Machine and Human Science, 1995. MHS’95., Proceedings of the Sixth International Symposium on*. IEEE, 1995, pp. 39–43.

[12] D. Gong, C. Qi, Y. Zhang, and M. Li, “Modified particle swarm optimization for odor source localization of multi-robot,” in *Evolutionary Computation (CEC), 2011 IEEE Congress on*. IEEE, 2011, pp. 130–136.

[13] S. Doctor, G. Venayagamoorthy, and V. Gudise, “Optimal pso for collective robotic search applications,” in *Evolutionary Computation, 2004. CEC2004. Congress on*, vol. 2. IEEE, 2004, pp. 1390–1395.

[14] J. Pugh and A. Martinoli, “Multi-robot learning with particle swarm optimization,” in *Proceedings of the fifth international joint conference on Autonomous agents and multiagent systems*. ACM, 2006, pp. 441–448.

[15] ——, “Inspiring and modeling multi-robot search with particle swarm optimization,” in *Swarm Intelligence Symposium, 2007. SIS 2007. IEEE*. IEEE, 2007, pp. 332–339.

[16] J. Hereford and M. Siebold, “Multi-robot search using a physically-embedded particle swarm optimization,” *International Journal of Computational Intelligence Research*, vol. 4, no. 2, pp. 197–209, 2008.

[17] S. Xue, J. Zhang, and J. Zeng, “Parallel asynchronous control strategy for target search with swarm robots,” *International Journal of Bio-Inspired Computation*, vol. 1, no. 3, pp. 151–163, 2009.

[18] J. Li, J. Yang, S. Cui, and L. Geng, “Speed limitation of a mobile robot and methodology of tracing odor plume in airflow environments,” *Procedia Engineering*, vol. 15, pp. 1041–1045, 2011.

[19] X. Songdong, Z. Yunlong, Z. Jianchao, X. Zhibin, and D. Jing, “Group decision making aided pso-type swarm robotic search,” in *Computer, Consumer and Control (IS3C), 2012 International Symposium on*. IEEE, 2012, pp. 785–788.

[20] M. Couceiro, R. Rocha, and N. Ferreira, “A novel multi-robot exploration approach based on particle swarm optimization algorithms,” in *Safety, Security, and Rescue Robotics (SSRR), 2011 IEEE International Symposium on*, nov. 2011, pp. 327 –332.

[21] A. Gasparri and M. Prosperi, “A bacterial colony growth algorithm for mobile robot localization,” *Autonomous Robots*, vol. 24, no. 4, pp. 349–364, 2008.

[22] S. Camazine, J. Deneubourg, N. Franks, J. Sneyd, G. Theraula, and E. Bonabeau, *Self-organization in biological systems*. Princeton University Press, 2003.

[23] A. Jati, G. Singh, P. Rakshit, A. Konar, E. Kim, and A. Nagar, “A hybridisation of improved harmony search and bacterial foraging for multi-robot motion planning,” in *Evolutionary Computation (CEC), 2012 IEEE Congress on*. IEEE, 2012, pp. 1–8.

[24] Y. Tan and Y. Zhu, “Fireworks algorithm for optimization,” *Advances in Swarm Intelligence*, pp. 355–364, 2010.

[25] Z. Zheng and Y. Tan, “An indexed k-d tree for neighborhood generation in swarm robotics simulation,” in *Advances in Swarm Intelligence*, ser. Lecture Notes in Computer Science, Y. Tan, Y. Shi, and H. Mo, Eds. Springer Berlin Heidelberg, 2013, vol. 7929, pp. 53–62.