2014 IEEE Congress on Evolutionary Computation (CEC)
July 6-11, 2014, Beijing, China

Avoiding Decoys in Multiple Targets Searching Problems using Swarm Robotics

Zhongyang Zheng, Junzhi Li, Jie Li and Ying Tan

Abstract— **In this paper, we consider the target searching problems with a new type of the object: decoys which can be sensed exactly as targets but cannot be collected by the robots. In real-life applications, decoys are very common especially for swarm robots whose hardware should be designed as simple and cheap as possible. This inevitably brings errors and mistakes in the sensing results and the swarm may mistakenly sense certain kinds of environment objects as the target they are looking for. We proposed a simple cooperative strategy to solve this problem, comparing with a non-cooperative strategy as the baseline. The strategies work with other searching algorithms and provide schemes for avoiding decoys. Simulation results demonstrate that the cooperative strategy shares almost the same computation overload yet has better performance in iterations and especially visited times of decoys. The strategy shows great adaptiveness to large scale problems and performs better when more decoys or robots exist in the simulation.**

I. Introduction

Swarm robotics has achieved significant progress benefiting from the development of artificial intelligent [1]. Swarm robotics can be used in many applications, especially those require large amount of robots and time as well as difficult or dangerous for human beings, e.g. foraging [2], surveillance [3], monitoring [4] and search-and-rescue [5]. These applications can be abstracted as a multiple target searching problem with restrictions in the environment, such as obstacles. Searching strategies for solving this abstracted problem can be adapted to many applications and remains an important task for swarm robotics researchers.

Inspired from the swarm intelligence, most of swarm robotics searching problems use fitness values to guide the robots in the swarm. In both simulation and entity researches, these values can determine the distance of the targets from the robot. Fitness values usually have corresponding meanings in physical world, such as Euclidean distance [6], olfaction measurements [7] or chemical clues [8] and potential functions [9]. Fitness values are continuous and as regular as contours. Such problems can be solved with gradient descent methods [10] or other local searching schemes [11]. However, hardware designs in swarm robotics should be as simple as possible which may leads to low quality on-board sensors and fault sensing results and errors [12].

Due to the reasons mentioned above, discrete fitness values are used in this paper. Continuous sensing results are rounded into discrete values which reduces the sensing errors significantly. The discrete fitness makes the problem a more little difficult but local searching strategies can still solve the problem. Considering hardware of the robots, they inevitable missense some objects in the environment as the target. Thus, we introduce another type of objects in the problem, decoys which have exactly the same behavior as targets in the robots' point of view except they are uncollectible by the swarm. The decoys disturb the searching of targets in a great extent, similar to local minimal in optimization problems. Two decoy avoiding strategies are proposed in this paper, one utilizes cooperation among robots and the other not. Both strategies are simple yet simulation results show that even simple cooperation among robots can improve the performance by at least 5%. Both two strategies are designed to avoid decoys not searching for targets and need to work together with algorithms proposed for target searching problem.

Thanks to the similarity in problem, many swarm intelligence algorithms inspired from nature are adopted directly into swarm robotics. Such nature scheme includes biological ones such as PSO [13], ACO [14] and others such as firework explosions [15]. However, these algorithms may not be completely suitable for swarm robotics, as robots should have as limited abilities as possible including, but not limited to, motion, storage, sensing, communication, energy consumption and computation. Swarm intelligence algorithms cannot meet the requirement of swarm robotics, such as limited sensing and communication ranges, dynamic adding or removing the robots and continuous movement. Therefore several variants of these algorithms are proposed for swarm robotics. We choose two underlaying algorithms in this paper to cope with the proposed decoy avoiding strategy. The two algorithms are in different type, one is RPSO [16] inspired from the biological scheme of PSO and the other is GES, proposed in our previous work [17] inspired from explosion phenomenon. These two algorithms are from different types and outperform each other in certain environment setups. Both strategies can take effect on both algorithms, showing the adaptiveness of the strategies.

In this paper, searching problem with decoys and discrete fitness is first introduced in Section II. Then section III describes the two decoy avoiding strategies in detail. Experimental results and discussions are presented in IV. Finally, Section V concludes work in this paper.

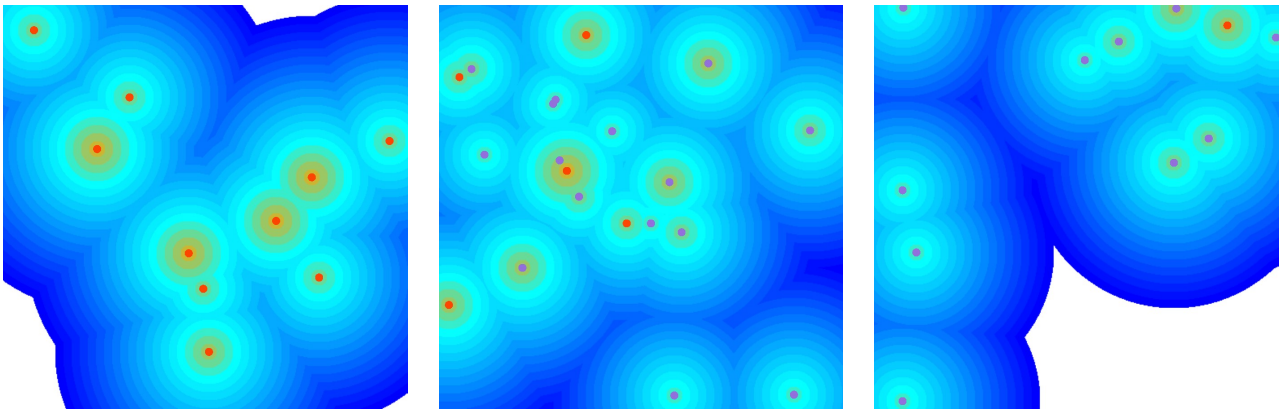
II. Problem Statement

In our previous work, we introduced the problem for searching multiple targets. In the problem, a swarm of robots

All authors are with 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.

This work was supported by the National Natural Science Foundation of China with grants no. 61375119, 61170057 and 60875080. Professor Ying Tan is the corresponding author.

978-1-4799-1488-3/14/$31.00 ©2014 IEEE



(a) Original Searching Problem (b) Problem with Decoys (c) Problem with one Target and Decoys

Fig. 1: A screenshot of the searching problem at the beginning of the simulation. Red circles stand for the targets and orange circles stand for interferences. Black squares are obstacles. The background color illustrates fitness of that position. Position of targets, interferences and obstacles are generated randomly. Robots are not illustrated.

searches for multiple targets in obstructive environment as shown in Figure 1a. All the targets are detectable and collectable for the robots. The simulation stops after the swarm collects all the targets or the iteration limit exceeds. However, in real-life applications, we cannot guarantee that the only targets can be sensed by the sensors. Some objects may share the same characteristics with targets in the view of the limited sensors on-board. In military applications, false targets may also exist to interfere the searching process of the swarm. The swarm cannot deal with these objects, yet the robots can still sense these objects which may interfere the search for real targets. Therefore, we introduce a new type of object "decoy" into the searching problem.

A. Outline of Problem

The multiple target searching problem in this paper is quite similar with other searching problems in [16], [18], except the fitness values in the environment are discrete. The reason for discrete fitness values is that robots in the swarm robotic researches should be as simple as possible. The on-board sensors should be simple and cheap, and are thus inevitably inadequate for sensing the targets. The direct result from the sensors may be continuous. However, considering the errors, we divide the whole sensing range into several intervals and approximate the result to the nearest interval. The results become discrete yet more reliable than the raw ones.

The swarm searches in an environment with targets and decoys. Both targets and decoys can be sensed with the help of fitness. Fitness is relevant to the distance to the targets and decoys and larger fitness indicates a closer distance. The fitness value of targets and decoys are the same so that robots cannot distinguish the fitness of a target of decoy until it reaches the position of the target or decoy. If the robot finds a target, it can collect the target and continue to search for other targets. The aim of the swarm is to collect all the targets in the environment as quickly as possible.

The swarm has no prior information about the environment. The problem is simulated in our self-built computer simulation program. Though time is divided into discrete iterations in simulation, all robots in the swarm are not restricted to share the same iteration cycle in real applications. Every iteration, sensing information of each robot is first updated from the environment and their neighbours, then the algorithm calculates robots' movements according to the sensing results. In the simulation, maximum speed of the robots is restricted so that their movements are guaranteed to complete within one iteration.

The main objective and challenge for this problem is to try to avoid the decoy while some targets may be very close to the decoys as shown in Figure 1. These two tasks are conflicting and the strategy should make a balance between them as the most important criteria is the total iterations used for collecting all the targets. Although in real-life applications, positions of targets and decoys may have different distributions which makes the problem easier, we consider the more general and difficult situation in this paper. If the swarm find all other targets very quickly and avoids the decoys well, but got stuck for the last target which is close to a decoy, we can hardly say the algorithm is suitable for solving this problem.

B. Robots

The n robots in this problem are autonomous and mobile, designed to be as simple and cheap as possible. They are modeled as squares or cubes with the ability to move freely in environment. The swarm has no leader nor unique IDs without any common coordinate systems or global position systems. Each robot can sense the fitness at its current position and detect the relative positions of their neighbour robots within limited sensing range. They have a limited memory of past states of themselves. Each robot communicates explicitly with other robots only for sharing their current fitness to neighbours. Each individual executes the

same algorithm but acts independently and asynchronously from all others. Although the robots are very simple, through carefully designed strategies they can emerge great intelligent behaviors through cooperation.

The sensing range of neighbour robots is set as $4Size_t$ so that robots can possibly detect nearby neighbour with different fitness values so as to accelerate the search. All the detected positions are relative which can be done through infrared sensors and angle transducers. Therefore, direct communications are not required nor the need to synchronize the position system in the swarm. If F_{Max} is quite small, the robots can detect neighbors' fitness values 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. They can past a fitness level in one iteration at the maximum speed and react quickly to the fitness when searching in 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 in the local coordinating system which is updated according to its movement. Past states cannot be shared among the swarm since complex communications and localizations are required. This conflicts with the principle of the swarm robotics: simple and elegant [19].

C. Targets and Decoys

An illustration of the problem is shown in Figure 1b. There exists m targets and d decoys in the environment and the swarm searches for the targets with the aid of fitness values. The fitness are generated by the targets and decoys exactly in the same way and it's impossible for the robots to distinguish them through fitness. Fitness values are discrete and inversely proportional to the distance from the target, as shown in Figure 1a. The swarm should search and collect the targets as quickly as possible. A target cannot be collected immediately and requires 10 iteration*robot to do it, i.e. it takes one robot 10 iterations and 10 robots one iteration to collect the target. This means the cooperation of multiple individuals collecting the same target in the same time can accelerate this progress. A target disappears after it has been collected and its fitness can be sensed no more.

A target or a decoy is found only if a robot runs into the target. The robot can start collect the target at its position or just leave the decoy which cannot be collected. Fitness value of each target and decoy is generated randomly and the fitness of the decoy is lower by 2 tan the target on average. Fitness values of the targets range from $F_{Max} - 2$ to F_{Max} and fitness values of decoys range from $F_{Max} - 4$ to $F_{Max} - 2$. We can notice here that it's possible a target and a decoy shares the same fitness. F_{Max} is a predefined constant set as 20 in this paper. Therefore, the fitness values that robots can sense range from F_{Max} to 0. A 0 fitness indicates no targets or decoys nearby.

Targets and decoys shape as circles or spheres with the radius of $Size_t$. $Size_t$ is a predefined constants set as 10. The fitness values nearby the target or decoy shapes as several rings placed one by one outside the target with decreasing fitness values. The first ring outside has the same radius ($Size_t$) and fitness value as the target or decoy. Although share the same fitness, the robots cannot find the target or decoy when they are in this ring. Other rings are placed outside its previous ring with twice the radius ($2Size_t$) and fitness values descending by 1 until 0. When fitness of several targets and decoys overlap, the largest value is adopted.

Decoys and targets are almost the same except decoys cannot be collected by robots. This makes the problem much more difficult than original problem without decoys as shown in Figure 1. A target may be very close to another decoy or surrounded by several decoys. This makes these targets hard to find for the swarm. In an extreme situation, there are only one target and many decoys in the environment (Figure 1c). It's quite challenging for the swarm to search for the only target especially in our situation that positions of targets and decoys share the same distribution.

Although robots can have limited storage of its own history (positions and fitness values), they do not maintain a list of visited decoys, i.e. decoy history. Considering distribution of targets and decoys positions in the environment as well as the strategies proposed in this paper, decoy history may not always accelerate the searching progress. This will be explained in detail in Section III-C.

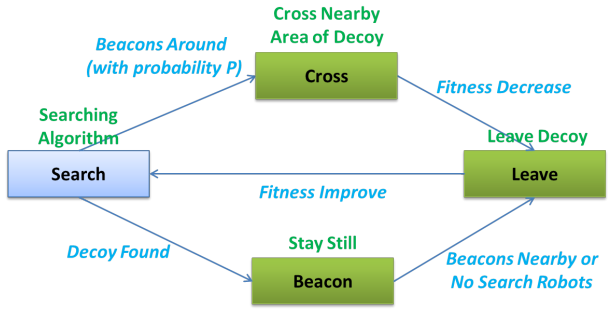
III. Decoys Avoiding Strategies

In this section, we propose two strategies to solve the multiple target searching problem with decoys. Flow charts of the two strategies are shown in Figure 2. The main difference of the two strategies is that one strategy makes use of cooperation among robots and the other one does not. Robots share their information of decoys they found to nearby neighbours in the cooperative strategy which can be done without direct communication. By comparing the performance of these two strategies, we can show that the cooperative strategy can faster the searching progress and reduce the total visits of decoys with very simple cooperating and interacting behaviors.

It should be noted that the strategy we propose here are not algorithms used for searching targets, but strategies for avoiding decoys the robots encounter during the search. Therefore, the strategy should be used together with other searching algorithms so as to complete the task. The decoy avoiding strategy acts like a layer on top of the searching algorithm. Every iteration, robots would first see if they have found a decoy or a decoy is nearby (in cooperative strategy) then it triggers the decoy avoiding strategy this iteration and will not call the searching algorithm underneath. Otherwise, the robots search normally following the algorithm as if there is no decoy avoiding strategies.

In our simulation in Section IV, we test our two strategies on two algorithms: GES [17] and RPSO [16]. In our previous work, GES is proposed and RPSO is used as the

comparison algorithm. The algorithms are of different types: GES is a heuristic algorithm inspired from explosion schemes in nature, and RPSO utilizes cooperation strategy in PSO into swarm robotics. The two algorithms performs better in different environment setups and tests on both algorithms can show our strategies proposed in this paper can adapt to different algorithms to a certain extent.



(a) Flow chart of the Cooperative Strategy

(b) Flow chart of the Non-Cooperative Strategy

Fig. 2: Flow charts of the two strategies. Green texts indicate the behaviors of robots in current states, blue italic texts and arrows indicate the conditions for state transitions.

A. Cooperative Strategy

Flow chart of the cooperative strategy is shown in Figure 2a. Robots may stay in four states in this strategy: search, beacon, cross and leave. The cross state is a pre-state for leave state, which will be explained later. Robots who found a decoy would stay at the position and act as beacons. Beacon robots will give out signals so that other robots nearby can sense it and start to leave the decoy. This can be done without any direct communication with the help of colored lights on board, which makes both the software and hardware parts of the robots simple and cheap.

Robots start in the search state running the searching algorithms to find targets or decoys. Once a robot finds a decoy, it goes into beacon state. The beacon robot stays so that every robot nearby can sense its signal. Searching robots go into cross state (and then leave state) when a signal is sensed from a beacon. The distance for a robot to detect the beacon is the same as the robot's sensing range, just as a robot detects its neighbours. Beacon robots will go into leave state when every robot around gets the signal (i.e. every robot around is in the cross or leave state) or another beacon is very close to it. Multiple beacons in the same place is a waste, so only one beacon will remain.

1) Leave and Cross State: The idea of the leave state is to leave the decoy. Each robot in the two states selects a random direction to leave and go straight at the maximum speed allowed until the fitness value increases. When leaving the decoy, the fitness normally drops since the robots are getting further to the decoy. When the fitness value increases, this means the robot encounters the fitness area of another target or decoy. Therefore, the robot should leave the leave state and go into search state.

The robots can go into the leave state in two ways, and this makes a little different for selecting the random direction for the robot. If robots come from the beacon state, we assume they are at the same position with the decoy. They just leave in a random direction selected uniformly from angle 0 to 2π. However, if the robots are from the cross state, things becomes different. Targets and decoys may stay very close as shown in Figure 1b and 1c. If robots leave in the direction away from the decoy immediately, the area around the decoy can hardly be searched and this makes it very hard to find those targets very close to the decoy.

When robots in the search state receives a signal from the beacon, they go into the cross state first and then the leave state. In cross state, they will try to cross the area around the decoy in the hope of finding targets in this area, i.e. select a direction neither too close nor too far from the decoy. The robots will first encounter the increase of the fitness as they are getting closer to the decoy. After certain position, the fitness starts to drop as they are leaving the decoy. The robots then goes into the leave state and go straight in this direction until fitness improves again and go back to search state.

2) Direction Selection in Cross State: Since the maximum distance of the robot and the decoy in this situation is the sensing range of the robot, the angle from the selected direction in cross state to the line between the robot and the decoy should be within the range of $\left[-\frac{\pi}{3}, -\frac{\pi}{6}\right] \cup \left[\frac{\pi}{6}, \frac{\pi}{3}\right]$. The reason for choosing $\frac{\pi}{3}$ and $\frac{\pi}{6}$ as the border is that directions with angles inside the range have the best efficiency for the hope of searching the target in this area, if there exists any. Considering fitness of target is larger than decoy most of the time, we assume robot will experience a boost in fitness if a target exists near the decoy.

At the angle $\frac{\pi}{3}$ which is the largest one available for the robot, the robot, decoy and another position P in the selected direction can form a equilateral triangle. The position P shares the same fitness with the current robot position and the line between them may have a higher fitness. With any angle larger than $\frac{\pi}{3}$, the robot may not have the chance to experience the increase of the fitness. Thus, it's not helpful for searching targets near the decoy when angle is large.

If the robot takes the direction with angle $\frac{\pi}{6}$, the closest distance to the decoy would be half of the robot's sensing range at most. At this position, the fitness value is the same as the decoy which means this closeness is enough for the robot. It would be a waste for the robot to get any closer to the decoy, since it can not provide more information for finding targets near a decoy.

B. Non-Cooperative Strategy

Compared with the cooperative strategy, the non-cooperative strategy is simpler as shown in Figure 2b. There

are only two states in this strategy: search and leave. As no cooperative schemes is included, robots who found a decoy simple enters the leave state and select a random direction to leave until the fitness increases again. Since robots go into the leave state at the position of the decoy, the cross state is not necessary and robots can select the direction uniformly.

No cooperation occurs among the robots in this strategy. This strategy is used as the baseline for the decoy avoiding problem and shows how the simple cooperation in the previous strategy can improve the performance of the swarm.

C. Why not Storing Decoy History

In the original multiple targets problem in previous work, the robots have the ability to store limited past history states (position and fitness) to aid their searching process. However, in the decoy avoiding problem introduced in this paper, robots do not store the decoys they found. Actually, this scheme can be added to the strategy easily, as the history can act exactly as a beacon and the robot can go into cross and leave state. The main reason for not introducing such scheme is make the strategy simple and give a good balance for the swarm to find those targets nearby the decoys.

As the convergence speed of the swarm toward a target or decoy is decided by the underlaying algorithm rather than the strategy itself, the strategy should provide more opportunity for the swarm to get close to the decoy. If in other problems where targets and decoys may have different distributions in position and the swarm will not encounter the problem of closeness of the target and decoy, then storing decoy history can definitely accelerate the searching progress which may not be the case in the problem in this paper.

IV. Simulation Results and Discussions

In this section, the two decoy avoiding strategies are compared based on two underlaying algorithms. We first briefly introduce the underlaying algorithm and present the simulation results of several experiments conducted to test the performance of both strategies in various aspects. The algorithms are simulated in our self-built simulation platform [20] and tested under various environment setups to see its adaptability. First, parameter tuning and analysis of the cooperation strategy is presented. In the second validation experiment, the two strategies are compared in a default setup with various numbers of decoys to see if they are capable for solving the problem. In the last scalability experiment, results of two strategies in larger scale environment are presented. The adaptability to various numbers of targets, robots and decoys of the two strategies can be observed. The combination of strategy and algorithm in this section is named after the syntax like "GES-C" or "RPSO-N" indicating cooperative strategy with GES or non-cooperative strategy with RPSO.

In all the experiments, the simulations stop when all the targets are collected. The two most important criteria are the iterations used and times the swarm visited decoys, which are denoted as "iteration" and "decoy visit" in the results. Iteration determines how fast the swarm can collect the targets and shorter iteration indicates the better performance. Decoy visit shows how often the swarm finds a decoy, which is useful for real-life applications especially in situations when certain punishment will happen when a robot arrives at a decoy. In this paper however, the robots can just leave without any delay.

All the experiments in this section use the same environment setup. The map size is 1000*1000 while the size of a robot is 1 and its sensing range is 20 which means the environment is very difficult for the swarm to cover. In each test, 20 random maps are generated for different setups and each method is repeated 20 times. The results shown in this section are the average results of these 400 runs.

The non-cooperative strategy does not have any parameters and can be used directly. For the cooperative strategy which has only one parameter, parameter tuning and analysis is presented in section IV-B in a fixed environment setup. Other experiments use the tuned parameter for comparison.

A. Underlaying Algorithms

To verify the adaptiveness of the two strategies on different underlaying algorithm, the strategies are tested on two algorithms of different types. The two algorithms are GES [17] and RPSO [16]. Parameters of both algorithms are tuned in our previous work in the environments without decoys and are used directly in experiments in this paper.

GES is inspired from the explosion behavior of fireworks and makes use of both inter and intra group cooperation. The swarm is self-organized as several groups and searches in parallel for targets. In RPSO, each robot acts as a particle from the PSO and the spacial-based topology of the robots for calculating gbest is adopted. Except the basic searching behaviors, both underlaying algorithms share the same scheme for better comparison, such as history state updating.

The two algorithms show advantage in performance at different situations. GES performances better when the fitness is not adequate in the environment or the population is not very large. When environment is full of targets or the swarm size is very crowded in the environment, RPSO can find the targets more quickly. Comparison based on these two algorithms can show the adaptiveness the strategies to certain extend, since the two algorithms differs greatly in both schemes and performances.

B. Parameter analysis

When a searching robot receives signal from the beacon in the cooperative strategy, it has the probability P to leave the decoy and probability $1-P$ to remain searching nearby the decoy. The effect of P can be deduced from this strategy. A larger P makes the robot leave the decoy more easily and quickly and the robots can spend more time in searching other targets. A smaller P makes the robot stay and search nearby the decoy in case a target sits within this area.

The parameter is tuned in a fixed environment with the middle scale of number of targets and robots, compared with the values used in the scalability experiment in section IV-D. In this environment, 50 robots are used to collect 20 targets mixed among 80 decoys. Values of P are within the range

TABLE I: Validation results of two strategies.

Number of Decoy	GES-N Iteration	GES-N Decoy Visit	GES-N CPU Time	GES-C Iteration	GES-C Decoy Visit	GES-C CPU Time	RPSO-N Iteration	RPSO-N Decoy Visit	RPSO-N CPU Time	RPSO-C Iteration	RPSO-C Decoy Visit	RPSO-C CPU Time
0	**308.65**	0.00	**53.33**	313.70	0.00	60.53	**270.11**	0.00	**32.89**	271.66	0.00	34.70
20	564.48	199.19	**101.39**	**549.45**	**125.68**	105.55	456.21	256.62	**43.67**	**434.67**	**148.97**	43.73
40	726.86	346.00	**125.74**	**687.46**	**239.26**	134.27	614.94	407.38	**63.81**	**576.42**	**258.98**	64.46
60	915.16	473.16	187.84	**870.74**	**351.08**	**187.26**	785.44	549.11	**84.63**	**753.96**	**383.51**	86.21
80	1191.00	608.76	274.45	**1074.72**	**463.82**	**266.91**	912.74	673.22	104.58	**841.65**	**471.75**	**101.95**
100	1234.93	730.48	**274.37**	**1172.33**	**573.03**	290.49	1090.30	816.67	131.42	**1019.71**	**606.87**	**126.27**
120	1417.97	894.96	**319.48**	**1366.52**	**705.44**	347.21	1231.41	948.65	**151.22**	**1156.48**	**723.07**	156.52
140	1505.23	997.94	**327.97**	**1500.86**	**842.00**	370.19	1355.90	1072.74	173.67	**1279.65**	**837.92**	**172.55**
160	1569.09	1091.82	**352.27**	**1492.68**	**909.01**	373.71	1447.06	1174.49	**181.24**	**1415.99**	**965.62**	198.11
180	1676.17	1250.53	**384.63**	**1605.80**	**1098.01**	394.40	1582.72	1319.69	215.37	**1474.42**	**1048.11**	**202.89**
200	2012.64	1394.47	526.18	**1794.55**	**1185.32**	**473.98**	1615.38	1382.11	**223.67**	**1563.54**	**1149.92**	227.01

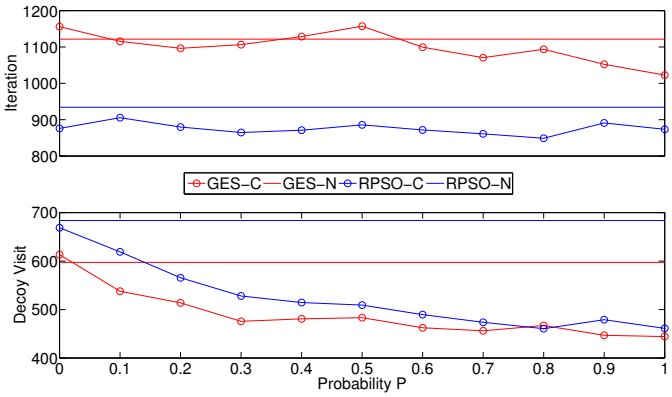


Fig. 3: Parameter analysis result of probability P.

$[0, 1]$ with a step size of 0.1. The result of non-cooperative strategy is also shown for comparison. The results are shown in Figure 3 with both iteration and decoy visit results.

Both iteration and decoy visit trends of GES and RPSO are quite similar except the turbulence of GES is a little larger than that of RPSO in iteration. This is simply because of the difference in basic idea of two algorithms, and GES may be a little sensitive to the decoys in this environment.

It can be seen from the figure that in almost all P values, cooperative strategy has a greater performance than non-cooperative strategy. This shows the cooperative strategy is taking effect with such few modifications than the non-cooperative strategy. The cooperation of the swarm can take effect even when P is very small.

The trend of decoy visit is almost linearly decreasing as it can be easily seen from the definition of P. A larger P reduces the possibility of a robot staying near the decoy. The robot leaves directly without visiting the decoy which makes this criterion small. However, the trend of iteration is more complex although the overall trend is still decreasing. This may have something to do with the distribution of decoys and targets as well as the randomness when a robot is making the decision to stay or leave.

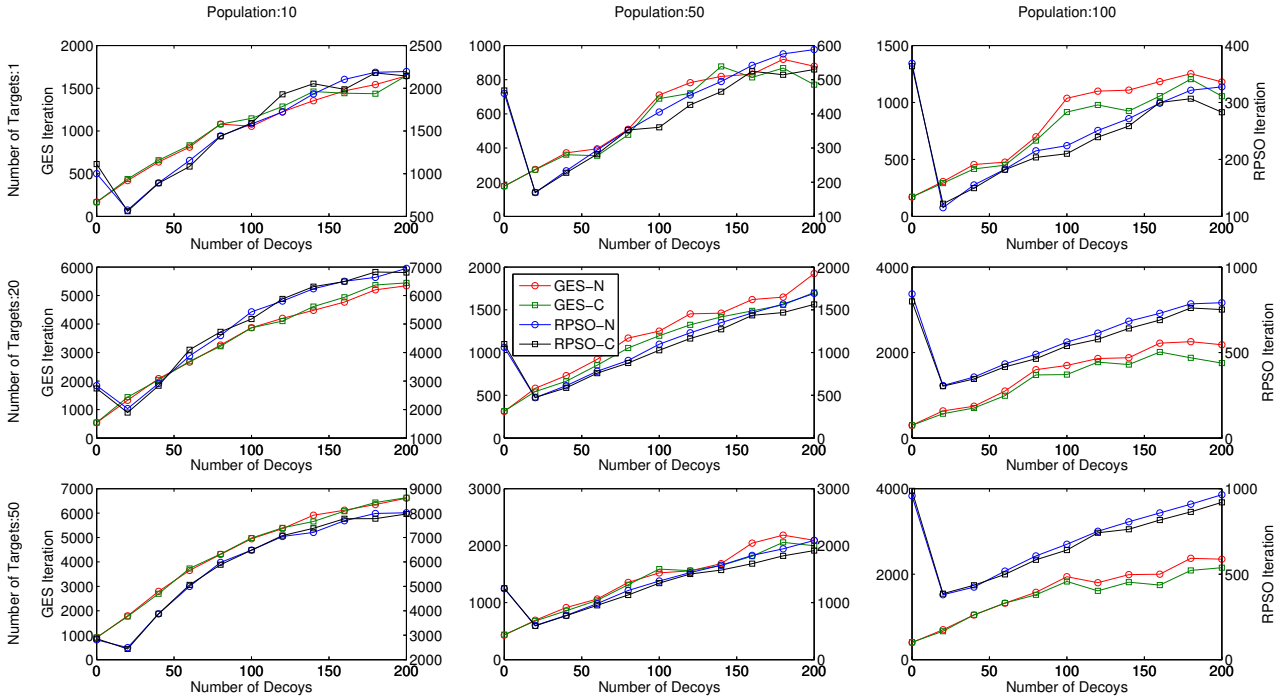
C. Validation Experiment

Validation results are shown in Table I. Results of two strategies combined with two algorithms are presented only in the same environment as we tune the parameters except that various numbers of decoys are selected from the range of 0 and 200. The full results of different setups are shown in scalability experiment. The column "CPU Time" indicates the CPU time used for algorithm calculation itself in the simulation excluding environment updates and result outputs. Calculation is an important energy consuming activity in real robots; therefore a shorter CPU time saves swarm for a smaller battery which is critical for designs of the robots. Results of 0 decoys are shown in the first row to give a baseline of the searching ability of two algorithms.

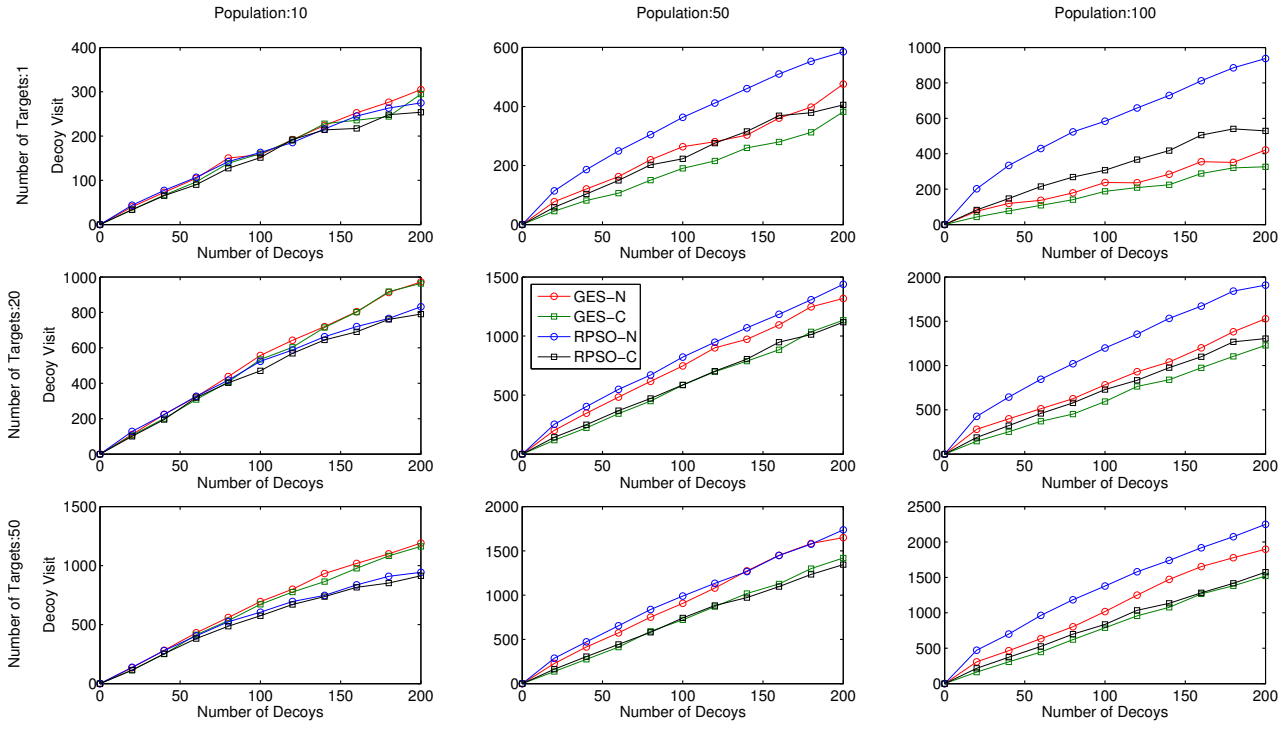
From the table, the iteration of cooperative strategy is about 5% better than that of non-cooperative strategy in both algorithms. The advantage seems not very large especially when the total iteration is only for hundreds of iterations. The main reason is that the cooperative strategy has to balance between the searching of targets near decoys against leaving the decoys directly. If the distribution of targets and decoys are different, or the scale of the experiment is even larger, the advantage of the cooperative strategy will be even more significant.

Meanwhile, cooperative strategy has much significant advantage in the decoy visit criteria. The advantage can reach 15-25% for GES and a slightly better 20-30% for RPSO. This indicates the cooperative strategy can do avoid the decoys with great performance. This means cooperative strategy is very useful for many real-life situations when visiting decoys should be avoided which may damage the robots.

The cooperative strategy shows better performance yet shares almost the same CPU time with non-cooperative one. This means that the proposed cooperative strategy is very simple and really effective with such improvement in performance without any loss of computational overloads. This proves that simple and fast cooperative schemes at robot level can really show great performance at swarm level.



(a) Iteration Results



(b) Decoy Visit Results

Fig. 4: Scalability results. The legend of the whole figure is shown in the center sub-figure. In sub-figure (a), GES iterations are with the left Y axis and RPSO iterations are with the right Y axis. Two Y axes may have different value range. The comparison occurs within two groups: 1) GES-N (red) and GES-C (green) 2) RPSO-N (blue) and RPSO-C (black). The frontier ones are supposed be higher (have larger value) than the latter one in the same group.

D. Scalable Experiment

In the validation experiment, only a fixed number of targets and robots is considered which can hardly show the effect and adaptiveness of the strategies. In scalability experiment, we present the a series of results under several numbers

of environment setups in both small and large scales. As mentioned before, an extreme situation where only 1 target exists in the environment is also taken into consideration. Such situation can occur in some real-life applications and the performance in this situation can show the adaptiveness of the proposed strategies.

Results are shown in Figure 4 in a three by three table where the rows represent the results of different numbers of targets (1, 20 and 50) and the columns represent the swarm populations (10, 50 and 100). X and Y axis of each sub-figure stands for the number of decoys and iteration or decoy visit.

From the figure, we can see that the overall advantages of cooperative strategy in iteration and decoy visit are quite similar than in previous experiment. The advantage in decoy visit is much more obvious than that of iteration results.

Advantage of iterations grows bigger as the population increases. There's hardly any advantage when population is 10, but it becomes quite obvious when population is 50 and 100. This is easy to understand as the cooperative strategy is hard to take effect when population is small. The beacon has to stay at the decoy for a few iterations and this is a large cost when population is small.

The performance is also affected by the ratio of number of decoys among targets. If the density of decoys is large, say in the first row in the figure where only one target exist, the cooperative strategy shows great advantage against the non-cooperative strategies. As the target grows or when decoys are small, the advantage is not that obvious in the figure. It's not surprise to see such results from the definition of the cooperative strategy as it requires more decoys to take effect.

V. Conclusion

Due to the restriction and designs of hardware of swarm robots which should be as simple as possible, some objects may be mistakenly sensed as targets in real-life applications. In this paper, a new type of object: decoys, are introduced into the swarm robotics multiple target searching problem. Decoys behave exactly the same as the targets from robots' view, except that decoys cannot be collected and targets can. This makes the searching problem more difficult and realistic.

We proposed two strategies for solving the problem with decoys, one cooperative and the other not. The strategies are used to avoid decoys while trying to searching the targets using the underlaying algorithms. Through our experiment, the cooperative one shows better performance and good adaptiveness with almost the same computational overload. The cooperative strategy is especially effective when there are more decoys in the environment or the population is large, demonstrating the cooperative mechanism is very useful.

As for future work, we plan to improve the strategy with more complex yet effective schemes, such as changing the value of P adaptively in the search. We also plan to introduce more types of objects that make the problem more realistic and possibly difficult, such as objects that can damage the robots. This will make the swarm robotics searching problem more applicable in real-life.

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] J.-H. Lee, C. W. Ahn, and J. An, "A honey bee swarm-inspired cooperation algorithm for foraging swarm robots: An empirical analysis," in *Advanced Intelligent Mechatronics (AIM), 2013 IEEE/ASME International Conference on*. IEEE, 2013, pp. 489–493.

[3] M. Masár, "A biologically inspired swarm robot coordination algorithm for exploration and surveillance," in *Intelligent Engineering Systems (INES), 2013 IEEE 17th International Conference on*. IEEE, 2013, pp. 271–275.

[4] A. Khapalov, "Source localization and sensor placement in environmental monitoring," *International Journal of Applied Mathematics and Computer Science*, vol. 20, no. 3, pp. 445–458, 2010.

[5] 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.

[6] 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.

[7] 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.

[8] G. Cabrita and L. Marques, "Divergence-based odor source declaration," in *Control Conference (ASCC), 2013 9th Asian*. IEEE, 2013, pp. 1–6.

[9] 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.

[10] 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.

[11] J.-G. Li, Q.-H. Meng, Y. Wang, and M. Zeng, "Odor source localization using a mobile robot in outdoor airflow environments with a particle filter algorithm," *Autonomous Robots*, vol. 30, no. 3, pp. 281–292, 2011.

[12] A. Amory, B. Meyer, C. Osterloh, T. Tosik, and E. Maehle, "Towards fault-tolerant and energy-efficient swarms of underwater robots," in *Parallel and Distributed Processing Symposium Workshops & PhD Forum (IPDPSW), 2013 IEEE 27th International*. IEEE, 2013, pp. 1550–1553.

[13] 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.

[14] W. Li and W. Shen, "Swarm behavior control of mobile multi-robots with wireless sensor networks," *Journal of Network and Computer Applications*, vol. 34, no. 4, pp. 1398–1407, 2011.

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

[16] 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.

[17] Z. Zheng and Y. Tan, "Group explosion strategy for searching multiple targets using swarm robotic," in *IEEE Congress on Evolutionary Computation*, 2013, pp. 821–828.

[18] 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.

[19] J. McLurkin, A. J. Lynch, S. Rixner, T. W. Barr, A. Chou, K. Foster, and S. Bilstein, "A low-cost multi-robot system for research, teaching, and outreach," in *Distributed Autonomous Robotic Systems*. Springer Berlin Heidelberg, 2013, pp. 597–609.

[20] 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.