Communication-Aware Distributed PSO for Dynamic Robotic Search

Logan Perreault
Montana State University
Bozeman, Montana 59715
logan.perreault@cs.montana.edu

Mike P. Wittie
Montana State University
Bozeman, Montana 59715
mwittie@cs.montana.edu

John Sheppard
Montana State University
Bozeman, Montana 59715
john.sheppard@cs.montana.edu

***Abstract*—The use of swarm robotics in search tasks is an active area of research. A variety of algorithms have been developed that effectively direct robots toward a desired target by leveraging their collaborative sensing capabilities. Unfortunately, these algorithms often neglect the task of communicating possible task solutions outside of the swarm. Many scenarios require a monitoring station that must receive updates from robots within the swarm. This task is trivial in constrained locations, but becomes difficult as the search area increases and communication between nodes is not always possible. A second shortcoming of existing algorithms is the inability to find and track mobile targets. We propose an extension to the distributed Particle Swarm Optimization algorithm that is both communication-aware and capable of tracking mobile targets within a search space. Simulated experiments show that our algorithm returns more accurate solutions to a monitoring station than existing algorithms, especially in scenarios, where the target value or location changes over time.**

I. Introduction

Many tasks are more suitable for robots than humans. Search problems such as detection of explosives and radioactive sources can be made safer by using robots operating under a well-performing algorithm. Consider a scenario where a radioactive object is lost or stolen and must be recovered quickly to protect the health and safety of the public. Airborne robots equipped with radiation sensors could be deployed to search for the object. These robots must locate the object and relay its location to a monitoring station. Search reporting is difficult when the search area is much larger than the communication range of the robots. Additionally, the target may be mobile during the search process, which means that robots must not only report the target's position, but also track it.

Emerging technology has revealed that swarms of small, inexpensive robots may be more effective at performing search tasks than their larger counterparts [1]. These swarms utilize concepts from many different fields and may be applied to a wide variety of applications [2]. Unfortunately, these robotic swarms are only as effective as the distributed algorithms that enable their emergent functionality. This dependency introduces a need for practical algorithms capable of controlling a swarm's behavior to solve a variety of tasks. In regards to the search problem, robots must effectively find a solution as well as transmit this solution back to a monitoring station. The communication component of the problem can be difficult in large areas, especially when working with inexpensive, power constrained robots that may have a relatively short transmission range. When the target solution is mobile throughout the search process, the problem of tracking becomes more difficult still. Finding a target and transmitting the solution to a server are distinct goals that may have conflicting optimal solutions, and effective distributed algorithms must balance both objectives.

An existing algorithm called *distributed Particle Swarm Optimization* (dPSO) has been shown to perform reasonably well in simulated environments [3–7]. However, previous work does not address the problem of communicating a solution back to a central server. Once a swarm of autonomous robots has found a solution to a problem, that solution must be relayed back to a monitoring node that is capable of processing the information or initiating an external action.

Another potential shortcoming of the current dPSO algorithm is the inability to track mobile targets. The original PSO algorithm cannot be applied directly to mobile target search, as this typically requires reevaluation of a particular location's fitness at a later timestep, or posting sentinels dedicated to detecting change [8]. Existing studies have focused strictly on static targets within a search space. This is acceptable for scenarios where optima are static, but may not be sufficient for robotic search, where a target may move during the search process.

The combination of search and connectivity objectives is a novel problem addressed by this paper. We propose an improved dPSO algorithm capable of tracking mobile targets as well as transmitting solutions to a server throughout the course of the search process. This new communication-oriented dPSO algorithm (C-dPSO) uses a modified velocity update equation to maintain consistent communication with the server. We also propose decaying fitness values to track targets whose location, or emitted fitness value changes over time. The result is a distributed algorithm for robotic swarms that maintains communication with a central server, while solving the problem of search with dynamic optima. The computation placed on each robot is relatively lightweight and well-suited to inexpensive, power constrained robots. The contributions made in this paper provide the means for such robots to solve difficult search problems over large areas.

The remainder of this paper is organized as follows. In

Section II, we discuss related work and identify areas for improvement within the swarm robotic literature. In Section III we introduce our C-dPSO and in Section IV we discuss the concept of decaying target values. Section V states the hypotheses of this study. Our experiments are described in Section VI and the results are presented in Section VII. Finally, we discuss the implication of these findings in Section VIII and conclude with future work in Section IX.

II. Related Work

The related work for this research can be separated into three main groups. The first, described in Section II-A, deals with PSO distributed across a swarm of robots. In this case, the algorithms view each robot as a separate particle in the PSO algorithm. Our C-dPSO algorithm behaves in this fashion, and therefore is most closely related to these works. Section II-B discusses several algorithms that use PSO to perform particular tasks for robots but do not necessarily distribute the algorithm onto individual robots. Although somewhat different from our approach, these algorithms solve related problems. Finally, Section II-C discusses some of the methods being used to perform message routing that were influential to our work.

A. Distributed PSO

This work focuses primarily on an extension of Kennedy and Eberhart's PSO algorithm into a physical environment [9]. Rather than using virtual particles that model physical movement, dPSO attempts to perform the same tasks using swarms of robots with actual parameters. The principles are similar to the original algorithm, but the replacement of particles with robots introduces a new set of problems Although dPSO still uses the same velocity update equation, the performance of such an update may be unreliable given a noisy or variable performance of hardware components. Communication between agents in a robotic swarm is also not guaranteed as they are in PSO, which makes it difficult to transmit the global best position to other agents or a central monitoring node. A relatively recent study provides an excellent review of PSO applications, and shows that publications involving robotics account for approximately 3.4% of the literature [10]. Many of these works deal with robotic control, and only a small subset are dedicated to the search problem.

Several studies have been conducted that focus on applying PSO to the robot search problem. Hereford first proposed the concept of dPSO as a solution to the robotic search problem [3]. The primary contribution was the notion that robots can represent particles when working in a physical search space.[1] The proposed algorithm is decentralized by pushing computation from a supervising node onto the individual robots. These robots calculate new target locations and update personal best values without any interaction with an outside controller/coordinator. A broadcast of position and value are sent only when a robot finds a new global best, which reduces the number of transmissions sent among nodes. Experiments with simulated robots were used within a 2D space to track 10 different static targets that each produced a detectable emission. An extension of Hereford's work implements this algorithm on three physical robots that attempted to find light sources within a room [4]. In both the simulated and physical environment, robots were able to locate targets by working cooperatively.

[1]Throughout the course of this paper, we use the terms robot and particle interchangeably.

Another study was conducted in parallel to, but independent from Hereford. Here, the authors developed a PSO inspired robotic olfactory search algorithm [5]. This study focused mainly on the specific case of odor detection and search. Although their algorithm was proposed for use with existing hardware, all experiments were conducted using simulations. Odor sources remained stationary throughout the numerous experiments.

A paper by Pugh and Martinoli [7] extended their previous work [6] and introduces another implementation of dPSO. Similar to Hereford's work, this study also focused on the problem of finding multiple static targets within a search space. The algorithm was implemented on a simulated environment using e-puck robots [11]. Unlike Hereford's work, this implementation had robots communicate with one another at every timestep. This paper is related to their previous work, in which the authors use PSO as a noise resistant robot learning algorithm [12]. While similar to our own work, this paper did not deal with robotic search and did not distribute the PSO algorithm among the various nodes.

Our implementation of dPSO differs from all previous work in that it incorporates a communication scheme that will allow transmission of a solution back to a monitoring node. Although some of these studies assumed a limited transmission range between robots themselves, all experiments extracted the final solution by using an oracle that only existed within the simulation. In addition, previous studies attempted to solve problems where the target solution was static. Scenarios exist in which robots are required to track mobile targets. The original dPSO algorithm [3] cannot be modified trivially to perform mobile target search. It does seem possible to adapt Pugh and Martinoli's algorithm [7] to work with dynamic targets, but the reporting problem would remain unaddressed.

B. PSO for Robotic Control

A related paper [13] uses PSO to learn optimal parameters for the search problem, but not as a solution to the search problem itself. Other biologically inspired models include Digital Hormone Models [14], Pheromone Models [15], [16], and Stochastic Cellular Automata (SCA) [17]. These models, while similar, use different communication mechanisms for cooperation and in most cases have not been applied to the problem of search. A number of statistical models have also been developed [18–21] for use in robotic swarms. These probabilistic models are different from biologically inspired models (including PSO) in that they are used to plan robot behavior in an offline setting prior to runtime. Behavior for each robot is planned according to rules discovered prior to the search itself, and communication between robots is unnecessary during the modeling process.

Several papers have been published focusing on the robot path planning problem using PSO [22–25]. This is a different problem than dPSO. In path planning, a PSO algorithm is used to determine the optimal next location for a robot at any given timestep. The PSO algorithm itself still uses virtual particles to accomplish this, and the result is simply passed to a single robot as a plan of action. Distributed PSO aims to use robots as a replacement for particles with the logic of the PSO algorithm itself being distributed among them. Work by Nasrollahy and Javadi is relevant to robotic search because it addresses mobile targets [22]. Their approach, however, is different from our problem in that PSO is run at each iteration. Although the target may move over time, the PSO algorithm is solving for a static target at each timestep. We seek to deal with instances where a target may move during the course of the PSO algorithm.

C. Routing Protocols

Our work uses concepts likes opportunistic forwarding and delay tolerant networking to achieve communication within the network. These notions are prominent in wireless sensor networks (WSNs), and we adopt some of the techniques described in this literature for communication between robots. PSO has been used in WSNs for positioning mobile nodes for optimal coverage and quality of service [26]. Two publications use WSNs to find and track a moving target but use WSNs that have been previously deployed with a known topology [27], [28]. The communication model our system uses was inspired by Boudriga *et. al.* [29]. Our system assumes that robots are able to know their position accurately in physical space. The problem of finding node locality in WSNs is challenging in practice and much work has been done to attempt to solve it [30].

Finally, Haberman and Sheppard introduced a routing protocol for power-constrained sensor networks [31]. This work uses a distributed form of PSO that maintains a series of overlapping swarms, one for each node in the network. This is similar to our approach in that each node represents a single particle and the final goal is to provide communication with a data collection point. This algorithm was shown to reduce the energy consumption in sensor networks. Our algorithm differs from this research in that we use C-dPSO to direct the movement of robots such that communication with a server is available.

III. Communication-oriented dPSO

We propose a dPSO algorithm that incorporates a communication objective into the velocity update equation. Rather than simply pursuing solutions to a specific problem, nodes strive to remain in contact with the server. PSO traditionally operates by moving particles using a velocity update equation. The direction in which particles move corresponds to local and global best solutions as determined by a fitness function, which returns higher values for better solutions. In the case of robotic search, we assume all nodes are equipped with a sensor that detects the value of a position. A logical extension for dPSO is to augment the velocity update equation with information relating to server connection. This will have the effect of pulling individuals toward either the target solution or a location with server connection as necessary. Messages may be passed from robot to robot in an ad-hoc fashion, so direct connection to the server is not necessary to attain information exchange. This is related to work in delay tolerant networks, which attempt to maintain data flow even though a contiguous path may not exist [32].

The original PSO algorithm operates by moving particles based on a current velocity. Let p_i be the local best position seen by particle i and p_g be the global best position seen by any particle. The following is the velocity update rule for particle i.

$$v_i = \omega v_i + U(0, \phi_1) \otimes (p_i - x_i) + U(0, \phi_2) \otimes (p_g - x_i)$$

where $\otimes$ is a point-wise multiplication of elements. Here, ω acts as inertia and ϕ_1 and ϕ_2 denote the amount of influence the personal and global best positions have. Randomly sampling from a uniform distribution U introduces additional exploration that may otherwise be dominated by the exploitation of a particular solution. The term that makes use of the personal best location is often called the cognitive component, while the term containing the global best position is called the social component. For simplicity, we can rewrite the velocity update equation as follows. We use Φ to represent the random variables for readability.

$$v_i = \omega v_i + \Phi_1 Cognitive() + \Phi_2 Social()$$

This update occurs during each iteration of the algorithm prior to a particle's movement. In the context of distributed PSO, time is discretized into slices with each timestep serving as an iteration of the algorithm.

Ideally we would like communication with the server at all timesteps, but this is unrealistic for problems where the search space is larger than the communication range of a robot. As an alternative, we set a target number c of maximum timesteps before communication with the server is restored, thereby limiting the amount of time where no communication is available. To achieve this additional goal, some of the searching power of the system needs to be diverted to the task of restoring connection. This means that c is a tunable parameter, where lower values allow for more frequent communication with the server and higher values correspond to a more robust search of the available space.

Better performance can be achieved by varying the target number of timesteps for each particle. We can achieve this by adding a unique offset θ_i to each c that is different for each particle so that robots attempt to communicate with the server at different times. By introducing unique offsets, robots will travel back to the server one after another. Ideally this maximizes the effectiveness of the adhoc communication by creating a chain of robots back to the server. Assuming that the size of the swarm is known to be N, and each robot is assigned a unique identifier R_{id} that ranges from 0 to $N-1$, we can determine the timestep offset for each robot i as $\theta_i = R_{id} \times c/N$. By using this offset, each robot will start searching for the server at evenly spaced intervals at the beginning of the algorithm. As time continues, particles switch to searching for the server based on when communication with the server was last established.

This communication-oriented behavior can be achieved by modifying the velocity update equation to include a communication component.

$$\begin{aligned} v_i = \omega v_i &+ \left(1 - \min\left(1, \left\lfloor \frac{t-t_c}{c+\theta_i} \right\rfloor\right)\right) \Phi_1 Cognitive() \\ &+ \left(1 - \min\left(1, \left\lfloor \frac{t-t_c}{c+\theta_i} \right\rfloor\right)\right) \Phi_2 Social() \\ &+ \min\left(1, \left\lfloor \frac{t-t_c}{c+\theta_i} \right\rfloor\right) \Phi_3 Communication() \end{aligned}$$

Both the cognitive and social components are pursuing a solution within the space, and as such we may simplify the equation even further by combining these terms into a Goal() component.

$$\begin{aligned} v_i = \omega v_i &+ \left(1 - \min\left(1, \left\lfloor \frac{t-t_c}{c+\theta_i} \right\rfloor\right)\right) \Phi_x Goal() \\ &+ \min\left(1, \left\lfloor \frac{t-t_c}{c+\theta_i} \right\rfloor\right) \Phi_3 Communication() \end{aligned}$$

Here, t is the current timestep and t_c is the last timestep in which successful communication with the server occurred. This new velocity update now relates the tradeoff between search and communication objectives. When $t - t_c < c + \theta_i$ (communication with the server has been established within the threshold number of timesteps), the velocity update reverts back to the original PSO version where only the goal component is used. When $t - t_c \geq c + \theta_i$ (the target number of timesteps have occurred since the last successful communication with the server), the goal objective is entirely ignored and the robot focuses all efforts on regaining communication with the server.

The additional communication component requires a new fitness function that denotes how good a location is in terms of communication. A value of 0.0 is awarded to a position that is unable to communicate with the server. If a robot is capable of communicating with the server directly, it is given a value of 1.0. It is possible to establish a connection with the server via intermediary nodes, but each additional hop detracts from the communication value. The fitness value reduction can be represented using the following decay formula,

$$fitness_c = \frac{e}{e^H}$$

where e is the mathematical constant approximately equal to 2.718 and H denotes the number of hops required for communication with the server. When H = 1 (direct communication), a value of 1.0 is received as expected. As H increases there is an exponential drop-off such that as H approaches infinity the value goes to zero. In reality, there are a finite number of robots, meaning that a large enough increase in the number of hops will eventually result in failed communication with the server. This will also produce a zero value for the given position. When any connection is made to the server, t_c is set to t. If the connection value is the best seen throughout the course of the algorithm, the new communication best location is stored as s_i. This tracking of the last known position where communication was available is analogous to target tracking, in that each robot has its own opinion of where it believes it should go. For this reason, future work may investigate the C-dPSO's ability to track mobile servers as well as targets. The final, un-simplified velocity update equation with the incorporated communication component is as follows.

$$\begin{aligned} V_i &= \omega V_i \\ &+ \left(1 - \min\left(1, \left\lfloor \frac{t-t_c}{c+\theta_i} \right\rfloor\right)\right) U(0,\phi_1) \otimes (p_i - x_i) \\ &+ \left(1 - \min\left(1, \left\lfloor \frac{t-t_c}{c+\theta_i} \right\rfloor\right)\right) U(0,\phi_2) \otimes (p_g - x_i) \\ &+ \min\left(1, \left\lfloor \frac{t-t_c}{c+\theta_i} \right\rfloor\right) U(0,\phi_3) \otimes (s_i - x_i) \end{aligned}$$

It should be noted that communication to the server is only necessary when a new global best is found. There is no need to report back with no new information. This is accomplished by setting t_c to t for every timestep until a new best location is found. Essentially this means the timestep count does not begin until a valid solution is found. This allows a robot $c + \theta_i$ timesteps to exploit the new area in search of a better solution before defaulting back to the communication goal. Communication between nodes may void the need to connect to the server if a path through other robots can reach the server or if a better solution is found by another node prior to making contact with the server. To accomplish this, robots communicate their last known connection time with the server as well as what fitness value the server had stored at that point. Using this information, robots may determine if better information has already been relayed to the server. If so, t_c is set to t and the robot resumes the task of searching.

When a new global best is found by a robot, the robot broadcasts a message with the new global best value and its position to all the peer robots within range. Upon receiving a broadcast message that a new global best was found, the robot will rebroadcast the message. All duplicate messages are ignored. If the new global best message manages to propagate to the server, an acknowledgment will be sent to the original robot. Upon receiving the acknowledgment the robot resets t_c to the timestep when the new global best was discovered. Ties for the best fitness value use the one most recently observed. This is useful in dynamic scenarios, and is especially important for the communication fitness value where frequent ties occur.

IV. Extension to Dynamic Search

In addition to being communication aware, it is also desirable for dPSO to extend to problems with non-static targets. A mobile target creates a dynamic search problem with respect to position. It is also possible for targets to change their emitted fitness throughout time, which introduces another dynamic element to the task.

In traditional PSO, a best location (p_i or p_g) is stored based on the highest fitness value experienced by a particle (f_i or f_g respectively). If the target is in motion, the particles may remain fixed on a worthless location believing it still to have value. Typically this problem is solved by reevaluating stored best positions to verify its fitness at later timesteps. This solution applies to targets with changing fitness values as well. Unfortunately this is not possible in a physical environment because a particle is unable to evaluate the fitness of a location it is not currently at.

To avoid this issue, we decay the f_i and f_g values throughout the execution of the algorithm. Eventually, the worth of a location dissipates over time and nodes become interested in more valuable locations. If the target remains in the general vicinity, additional best locations will be found in the area and the values will be boosted back to their original starting positions. This change to the fitness values does not affect the velocity update equation directly and can be applied to either the original dPSO or the communication-oriented version. During each timestep, fitness values are decayed as follows,

$$\begin{aligned} f_i &\leftarrow \beta_1 f_i \\ f_g &\leftarrow \beta_2 f_g \\ f_c &\leftarrow \beta_3 f_c \end{aligned}$$

where $\beta_k \in (0, 1]$ is a decay factor. If the target solution is known to be stationary, $\beta_1 = \beta_2 = 1$. Similarly if the communication server is known to be stationary, $\beta_3 = 1$. By reducing as a percentage of the previous value at each timestep, the fitness values are modeled by a nonlinear decay function. The reduction in fitness can be thought of as a reduction in the confidence we have about the original observed value. As time continues, the information becomes less "fresh", and a more recently observed position may be more valuable despite having a lower fitness. This desired effect is achieved by decaying the values as described.

The values for β_k become new hyperparameters for the equation, and should be tuned based on how quickly the search task is changing in the system. For systems where the target is moving quickly, β_k must be set to a low value for the robots to value the freshness of information more. In situations where the fitness value is decreasing, β_k must be set so that the perceived fitness decays at least as quickly as the true fitness' rate of change. This is not an issue with increasing fitness values, as the believed fitness would be lower than the true fitness at a given point and the newer information would be stored. If values are decayed too slowly, particles will still face the issue of tracking outdated information. Alternatively, if values are decayed too quickly, particles will lose possibly valid information and essentially will be starting over with a random search when it may not be necessary. As targets move more quickly or their fitness values degrade at a higher rate, the β_k values must be set lower and lower, making it difficult to reuse collected information.

V. Hypotheses

The purpose of our evaluation is to determine the effectiveness of the proposed communication component and dPSO's ability to track mobile targets. A number of hypotheses have been created to test these elements individually. Solutions for all hypotheses are in reference to the robotic search problem. Error is determined using the Euclidean distance between the returned solution and actual target position.

First, we hypothesize that the addition of decay improves the performance of dPSO for mobile problems. We predict that dPSO with a tuned decay parameter will perform better than dPSO with no decay when tracking a mobile target, a target with a slowly decreasing fitness value, a target with a quickly decreasing fitness value, and a mobile target with a quickly decreasing fitness value. These problems represent a wide range of possible dynamic search problems, and the results form the foundation for our claims.

Next, we hypothesize that dPSO with a communication mechanism will perform better than dPSO without one. Performance is measured in terms of the server's knowledge of the target location, so communication is a valid concern. Five tests are run on the same scenarios as before, plus an additional static scenario where the target is stationary with a fixed fitness value.

VI. Experimental Approach

Experiments are conducted in a simulated environment. Rather than targeting a specific robot, additional constraints have been added to the classic PSO algorithm to which real-world robots must adhere. Specifically, only particles within a specified range of each other are allowed to exchange information, and the notion of a global best must be propagated through the network rather than being immediately available to all nodes. This simulates the potentially limited communication range that robots must adhere to, but this local best topology is often used in other PSO implementations as well to improve performance. Communication between nodes may fail with some small random chance, but message transfers are assumed to be successful after the start of the transmission. Also, the velocity for each particle is clamped to simulate a top speed for the robot. An additional node is constructed that acts as the server. The simulator does not consider movement constraints that may be imposed by specific robots (such as turn radius), and ignores the problem of collision as this can typically be handled by dedicated collision detection algorithms.

For all experiments, the robots operate in a two-dimensional space in search of a single target. Future work may investigate an extension into a three-dimensional space or problems with multiple targets or servers. Swarms are composed of a fixed number of eight robots. These robots, along with the server, have a communication range that is 25% of the total width/height of the search space. For each experiment, the search problem is solved 1000 times, where the target is randomly assigned to a new location for each iteration. These targets have a fixed fitness value that degrades non-linearly with distance from the source. In addition to these targets, noise is added to the search space by assigning a large number of randomly placed nodes that produce a small fitness value.

We seed the random placement of these nodes such that each experiment will be solving the same 1000 problems each time. We record several measurements during the course of the experiments, including solution error in terms of Euclidean distance between the correct target location and the server's belief of the solution location. For fairness, we allot 2000 timesteps to every algorithm, which should be sufficient for convergence in all cases. This is because there is a tradeoff between the quality of the solution and the time available, and we opt to provide sufficient time to better study the quality of the solution. To properly evaluate the dynamic scenarios, the error is measured during every timestep from timestep 1800 to 2000 and then averaged. This allows us to determine how well each algorithm is tracking a mobile target after an initial search period. A Student's t-test was used to test statistical significance in all cases.

Four experiments were conducted to test claims about the decay component. Each experiment uses the newly proposed communication-aware dPSO. The performance of C-dPSO with a tuned decay parameter is compared to C-dPSO without any decay when solving dynamic problems. β_k parameters were tuned by starting with a value of 1.0 (no decay), and decreased by 0.05 until performance on a training set of simulations achieved the highest value. We note that similar improvements are seen with any value smaller than 1.0, and that these hyperparameters are relatively insensitive to change. Experiment *Dynamic 1* attempts to find a mobile target whose top velocity is 20% of the top speed of the robots. Experiment *Dynamic 2* attempts to track a target whose fitness value is degrading linearly with time. In this case, the fitness value will be $\frac{1}{3}$ of the original value at the end of the 2000 timsteps. Similarly, experiment *Dynamic 3* will also have a linearly degrading fitness value. In this case however, the fitness value will reach zero by the time 1500 timesteps have occurred. Finally, experiment *Dynamic 4* combines the difficulties introduced by *Dynamic 1* and *Dynamic 3*. This will attempt to track a target moving at 20% of the speed of the robots whose fitness value is degrading and will have completely depleted by 1500 timesteps.

Additionally, five experiments were performed to test the claims about the communication component. Each experiment compares dPSO with the new communication-aware velocity update equation with the existing standard dPSO algorithm. Each algorithm uses the same optimally tuned decay parameters to observe the effects of the communication component on its own. Experiment *Comm 0* deals with a static problem

TABLE I: Summary of Experiment Attributes

Experiment	**Communication**	**Dynamic**	**Target**
Dynamic 1	✗	✓	Mobile
Dynamic 2	✗	✓	Decay
Dynamic 3	✗	✓	Fast Decay
Dynamic 4	✗	✓	Mobile Fast Decay
Comm 0	✓	✗	Static
Comm 1	✓	✗	Mobile
Comm 2	✓	✗	Decay
Comm 3	✓	✗	Fast Decay
Comm 4	✓	✗	Mobile Fast Decay

where the target is stationary and has a fixed fitness value. Experiments *Comm 1*, *Comm 2*, *Comm 3* and *Comm 4* operate on identical problems as *Dynamic 1*, *Dynamic 2*, *Dynamic 3*, and *Dynamic 4* respectively. The difference is that instead of analyzing the effects of decay, the communication aspect of the algorithm is now being tested. These nine experiments are summarized in Table I, where the *Communication* column indicates that the communication mechanism is being varied and *Dynamic* indicates that the decay component is being tested. The last column describes the behavior of the target during the search process.

VII. Experimental Results

Results for the dynamic experiments are shown in Table II. Recall that each experiment tests the performance of the C-dPSO algorithm with and without decaying the fitness values. At a significance level of 0.05, C-dPSO using decay outperformed (by reporting a more accurate solution) the version that did not for the mobile target experiment (*Dynamic 1*), the fast decay experiment (*Dynamic 3*) and the mobile fast decay experiment (*Dynamic 4*). In the slow decay experiment (*Dynamic 2*), C-dPSO still produces a smaller mean error, although this value is not significant until we consider a 0.1 significance level. Recall that experiment *Dynamic 2* was the case where the target's fitness value degrades slowly. It is likely the difference between the algorithms was smaller in this case, because the problem was close enough to the static case. This observation is supported by the fact that the fast decay experiment (*Dynamic 3*) did achieve statistical significance, which is simply the case where the target's fitness value degrades faster. These results show that decayed fitness is especially important when the target is in motion.

Table III shows the communication results for the static target (*Comm 0*), mobile target (*Comm 1*), slow decay (*Comm 2*), fast decay (*Comm 3*) and mobile fast decay (*Comm 4*) experiments. In this case, dPSO augmented with the communication mechanism (C-dPSO) performed better than the original dPSO algorithm for all experiments at a 0.05 significance level. This is not surprising, as the error is measured with respect to the server's belief of the target position. Without the communication component in the dPSO velocity update equation, the server is only notified if a robot enters communication range by chance. This means that the original dPSO algorithm may

TABLE II: Dynamic Results in Euclidean Distance

Experiment	C-dPSO	C-dPSO Decayed	p-value
Dynamic 1	53.435	**1.263**	$< 2.2e\text{-}16$
Dynamic 2	8.823	6.879	0.09863
Dynamic 3	12.071	**9.419**	0.04702
Dynamic 4	53.586	**27.936**	$< 2.2e\text{-}16$

TABLE III: Communication Results in Euclidean Distance

Experiment	dPSO	C-dPSO	p-value
Comm 0	8.308	**4.338**	8.591e-05
Comm 1	5.351	**1.263**	7.562e-12
Comm 2	10.159	**6.879**	4.994e-03
Comm 3	15.160	**9.419**	5.012e-05
Comm 4	30.072	**27.936**	3.093e-03

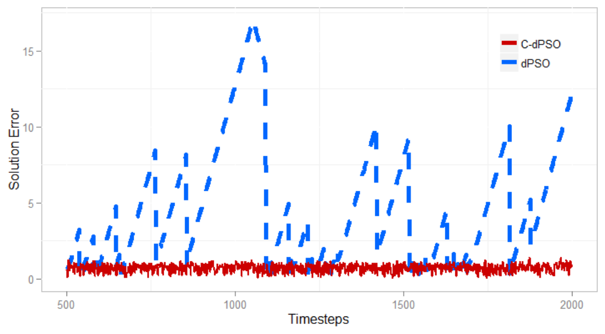

Fig. 1: Time series graph for mobile target search problem.

be capable of finding a solution but has no effective method of relaying this information to a useful location.

Figure 1 is a time series graph for a single simulation run on the mobile target experiment with decay. This figure illustrates the original dPSO algorithm's inability to communicate consistently with the server, as depicted by the spikes in solution error over time. While C-dPSO is required to report back to the server at routine intervals, standard dPSO focuses solely on finding the target. When measuring error from the server's point-of-view, this results in a consistently low error for C-dPSO and peculiar spikes in error for dPSO. These spikes occur as the target moves away from the location the server had stored previously. On occasion, a particle coincidentally comes in contact with the server and is able to communicate an updated position. Although this reduces the error to a near-zero value, it begins to rise again immediately once the particle moves away from the server. This behavior is consistent with other runs for all experiments. For this reason, Figure 1 serves as a representative for all other runs in this scenario.

The results shown in both tables strongly support the hypotheses presented in section V. Specifally, the addition of decay improves the performance of C-dPSO when faced with sufficiently dynamic search problems. Also, C-dPSO outperforms standard dPSO, when algorithm error is measured based on a servers' belief of the target location, or value.

VIII. Discussion

The proposed communication scheme will be useful if implemented on smart robotic swarms where the solution space is larger than the connection range to the server. The proposed approach for relaying information is distributed among the robots and dynamically adapts to changing conditions. In particular, this algorithm is agnostic with respect to the number of robots, servers, and targets. The fitness evaluation for the target can also be modified easily without changing any other parts of the algorithm. The coupling between solution and communication goals will direct robots toward necessary locations in the search space at all times. The distributed nature of the algorithm indicates that it should function despite a varying number of robots. In addition, the design of the algorithm does not place any restrictions on the size of the search space and will even allow for moving optima. Assuming the number of available robots is sufficient to provide reasonable coverage, the algorithm will be able to function in increasingly large search spaces.

Unattended search drones are a particularly useful application for robots capable of these tasks [33–35]. These drones are intended for use over a wide area and are often assigned to the task of finding a mobile target. This introduces situations where communication range is limited [36] and will necessarily require the continual tracking of a target. A specific, applicable event occured on March 3, 2014 when Malaysia Airlines Flight 370 went missing [37]. The aircraft presumably went down in the Indian Ocean, but no crash site was found. Automated underwater vehicles (AUVs) were used to scan the ocean floor for the signal emitted by the blackbox, but unfortunately the batteries powering the blackbox eventually depleted. In a scenario such as this, a form of dPSO with decaying fitness values may be an effective method of directing these autonomous vehicles to find the black box and export its location to a search ship.

IX. Conclusion

In this paper, we presented a version of distributed particle swarm optimization that incorporates a communication component into the velocity update equation. This additional term uses a separate fitness evaluation that promotes communication with the server. Experiments showed that for a variety of problems, our method provided a superior server-stored solution to that of standard dPSO.

We also introduced the concept of decaying fitness values to adapt dPSO to track moving targets. In this case, we found that decay is necessary to track effectively mobile targets or targets whose fitness is degrading over time. In scenarios where a solution is located outside of the communication range of a monitoring server, the discovered solutions are only as good as the ability to relay this information back to the server. This is particularly true of mobile tracking problems, where the server will ideally receive consistent updates on the whereabouts of the target.

In future work, we will investigate problems involving one or several mobile communication servers. Our algorithm currently supports the tracking of a server in the same way target tracking is accomplished. This may be useful in situations where a piloted vehicle is attempting to find a target with the assistance of several autonomous vehicles. Another interesting extension is to examine the effect of increased communication range for the server alone. It is feasible that the server node may have a larger communication range than the robots themselves, which would allow server information to be propagated more easily through the swarm.

Finally, we plan to extend these tests into 3-dimensional search problems. This extension is trivial as it simply requires an additional term in each particle's position and velocity vectors. Identical experiments would then be run to verify the usefulness of our algorithm for applications involving flight. We also hope to implement this algorithm on a set of real robots to eliminate any bias introduced by the simulator itself.

Acknowledgments

The authors would like to thank John McIntosh for his ideas during the formative stages of this research as well as his contributions to the related work section.

References

[1] J. Werfel, K. Petersen, and R. Nagpal, "Designing collective behavior in a termite-inspired robot construction team," *Science*, vol. 343, no. 6172, pp. 754–758, 2014.

[2] E. Şahin, "Swarm robotics: From sources of inspiration to domains of application," in *Swarm Robotics*. Springer, 2005, pp. 10–20.

[3] J. M. Hereford, "A distributed particle swarm optimization algorithm for swarm robotic applications," in *IEEE Congress on Evolutionary Computation*. IEEE, 2006, pp. 1678–1685.

[4] J. M. Hereford, M. Siebold, and S. Nichols, "Using the particle swarm optimization algorithm for robotic search applications," in *Proceedings of IEEE swarm intelligence symposium (SIS)*. IEEE, 2007, pp. 53–59.

[5] L. Marques, U. Nunes, and A. T. de Almeida, "Particle swarm-based olfactory guided search," *Autonomous Robots*, vol. 20, no. 3, pp. 277–287, 2006.

[6] J. Pugh and A. Martinoli, "Inspiring and modeling multi-robot search with particle swarm optimization," in *Proceedings of IEEE swarm intelligence symposium (SIS)*. IEEE, 2007, pp. 332–339.

[7] ——, "Distributed adaptation in multi-robot search using particle swarm optimization," in *From Animals to Animats 10*. Springer, 2008, pp. 393–402.

[8] D. Yazdani, B. Nasiri, R. Azizi, A. Sepas-Moghaddam, and M. R. Meybodi, "Optimization in dynamic environments utilizing a novel method based on particle swarm optimization," *Intl Journal of Artificial Intelligence*, vol. 11, no. 13, pp. 170–192, 2013.

[9] J. Kennedy, R. Eberhart *et al.*, "Particle swarm optimization," in *Proceedings of IEEE international conference on neural networks*, vol. 4, no. 2. Perth, Australia, 1995, pp. 1942–1948.

[10] R. Poli, "Analysis of the publications on the applications of particle swarm optimisation," *Journal of Artificial Evolution and Applications*, vol. 2008, p. 3, 2008.

[11] C. M. Cianci, X. Raemy, J. Pugh, and A. Martinoli, "Communication in a swarm of miniature robots: The e-puck as an educational tool for swarm robotics," in *Swarm Robotics*. Springer, 2007, pp. 103–115.

[12] J. Pugh, A. Martinoli, and Y. Zhang, "Particle swarm optimization for unsupervised robotic learning," in *Proceedings of IEEE swarm intelligence symposium (SIS)*, 2005, pp. 92–99.

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

[14] W.-M. Shen, P. Will, A. Galstyan, and C.-M. Chuong, "Hormone-inspired self-organization and distributed control of robotic swarms," *Autonomous Robots*, vol. 17, no. 1, pp. 93–105, 2004.

[15] D. Payton, R. Estkowski, and M. Howard, "Compound behaviors in pheromone robotics," *Robotics and Autonomous Systems*, vol. 44, no. 3, pp. 229–240, 2003.

[16] ——, "Pheromone robotics and the logic of virtual pheromones," in *Swarm Robotics*. Springer, 2005, pp. 45–57.

[17] G. Beni and J. Wang, "Swarm intelligence in cellular robotic systems," in *Robots and Biological Systems: Towards a New Bionics?* Springer, 1993, pp. 703–712.

[18] K. Lerman, A. Martinoli, and A. Galstyan, "A review of probabilistic macroscopic models for swarm robotic systems," in *Swarm Robotics*. Springer, 2005, pp. 143–152.

[19] A. Martinoli, K. Easton, and W. Agassounon, "Modeling swarm robotic systems: A case study in collaborative distributed manipulation," *The International Journal of Robotics Research*, vol. 23, no. 4-5, pp. 415–436, 2004.

[20] K. Lerman, A. Galstyan, A. Martinoli, and A. Ijspeert, "A macroscopic analytical model of collaboration in distributed robotic systems," *Artificial Life*, vol. 7, no. 4, pp. 375–393, 2001.

[21] K. Lerman and A. Galstyan, "Mathematical model of foraging in a group of robots: Effect of interference," *Autonomous Robots*, vol. 13, no. 2, pp. 127–141, 2002.

[22] A. Z. Nasrollahy and H. Javadi, "Using particle swarm optimization for robot path planning in dynamic environments with moving obstacles and target," in *Third UKSim European Symposium on Computer Modeling and Simulation*. IEEE, 2009, pp. 60–65.

[23] X. Chen and Y. Li, "Smooth path planning of a mobile robot using stochastic particle swarm optimization," in *Proceedings of the IEEE International Conference on Mechatronics and Automation*. IEEE, 2006, pp. 1722–1727.

[24] L. Wang, Y. Liu, H. Deng, and Y. Xu, "Obstacle-avoidance path planning for soccer robots using particle swarm optimization," in *IEEE International Conference on Robotics and Biomimetics*. IEEE, 2006, pp. 1233–1238.

[25] Y.-Q. Qin, D.-B. Sun, N. Li, and Y.-G. Cen, "Path planning for mobile robot using the particle swarm optimization with mutation operator," in *Proceedings of International Conference on Machine Learning and Cybernetics*, vol. 4. IEEE, 2004, pp. 2473–2478.

[26] R. V. Kulkarni and G. K. Venayagamoorthy, "Particle swarm optimization in wireless-sensor networks: A brief survey," *Trans. Sys. Man Cyber Part C*, vol. 41, no. 2, pp. 262–267, Mar. 2011.

[27] J. Lee, K. Cho, S. Lee, T. Kwon, and Y. Choi, "Distributed and energy-efficient target localization and tracking in wireless sensor networks," *Comput. Commun.*, vol. 29, no. 13-14, pp. 2494–2505, Aug. 2006.

[28] S. Misra and S. Singh, "Localized policy-based target tracking using wireless sensor networks," *ACM Trans. Sen. Netw.*, vol. 8, no. 3, pp. 27:1–27:30, Aug. 2012.

[29] N. Boudriga, M. Baghdadi, and M. Obaidat, "A new scheme for mobility, sensing, and security management in wireless ad hoc sensor networks," in *Simulation Symposium, 2006. 39th Annual*, April 2006, pp. 7 pp.–.

[30] G. Mao and B. Fidan, *Introduction to Wireless Sensor Network Localization*, 1st ed., ser. 1. USA: IGI Global, May 2009, vol. 1, p. 526.

[31] B. K. Haberman and J. W. Sheppard, "Overlapping particle swarms for energy-efficient routing in sensor networks," *Wireless Networks*, vol. 18, no. 4, pp. 351–363, 2012.

[32] M. P. Wittie, K. A. Harras, K. C. Almeroth, and E. M. Belding, "On the implications of routing metric staleness in delay tolerant networks," *Computer Communications*, vol. 32, no. 16, pp. 1699–1709, 2009.

[33] M. Flint, E. Fernandez, and M. Polycarpou, "Stochastic models of a cooperative autonomous uav search problem," *Military Operations Research*, vol. 8, no. 4, pp. 13–32, 2003.

[34] P. Sujit and D. Ghose, "Multiple uav search using agent based negotiation scheme," in *Proceedings of the American Control Conference*. IEEE, 2005, pp. 2995–3000.

[35] D. Graham-Rowe, "Cheap drones could replace search-and-rescue helicopters," *New Scientist*, vol. 207, no. 2769, p. 20, 2010.

[36] R. W. Beard and T. W. McLain, "Multiple uav cooperative search under collision avoidance and limited range communication constraints," in *42nd IEEE Conference on Decision and Control*, vol. 1. IEEE, 2003, pp. 25–30.

[37] C. Staff. (2014) Mh370 lost in indian ocean, malaysian pm announces. [Online]. Available: http://www.cnn.com/2014/03/24/world/asia/malaysia-airlines-prime-minister-statement/index.html