Robotics and Autonomous Systems 75 (2016) 422–434

Contents lists available at ScienceDirect

Robotics and Autonomous Systems

journal homepage: www.elsevier.com/locate/robot

Search and tracking algorithms for swarms of robots: A survey

Madhubhashi Senanayake [a,*], Ilankaikone Senthooran [a], Jan Carlo Barca [a], Hoam Chung [b], Joarder Kamruzzaman [c], Manzur Murshed [c]

[a] Faculty of Information Technology, Monash University, Clayton, VIC 3800, Australia
[b] Department of Mechanical and Aerospace Engineering, Monash University, Clayton, VIC 3800, Australia
[c] School of Engineering and Information Technology, Federation University Australia, Churchill, VIC 3842, Australia

HIGHLIGHTS

- Surveys algorithms applicable to swarm robotic systems for target search and tracking.
- Identifies variations of the search and tracking problem addressed in the literature.
- Discusses desired capabilities of search and tracking algorithms for robot swarms.

ARTICLE INFO

Article history:
Received 17 August 2014
Received in revised form
21 April 2015
Accepted 29 August 2015
Available online 7 September 2015

Keywords:
Swarm
Robotics
Search
Tracking
Review
Multi-robot systems

ABSTRACT

Target search and tracking is a classical but difficult problem in many research domains, including computer vision, wireless sensor networks and robotics. We review the seminal works that addressed this problem in the area of *swarm robotics*, which is the application of *swarm intelligence* principles to the control of multi-robot systems. *Robustness*, *scalability* and *flexibility*, as well as distributed sensing, make swarm robotic systems well suited for the problem of target search and tracking in real-world applications. We classify the works we review according to the variations and aspects of the search and tracking problems they addressed. As this is a particularly application-driven research area, the adopted taxonomy makes this review serve as a quick reference guide to our readers in identifying related works and approaches according to their problem at hand. By no means is this an exhaustive review, but an overview for researchers who are new to the swarm robotics field, to help them easily start off their research.

1. Introduction

The problem of target search and tracking has a very long history, and in recent times, more and more civilian applications have emerged, which include a wide variety of high-impact application areas, for example, search and rescue operations in disaster scenarios, exploration for natural resources, environmental monitoring, air traffic control and surveillance [1]. Target search and tracking is an important application of wireless sensor networks [2] and one of the oldest problems in computer vision [3]. This problem has also been tackled by the use of multi-robot systems (MRSs) [4], making use of the advantage mobile robots have with being capable of dynamically adapting to the target movements by changing their spatial distribution accordingly [5].

MRSs also have several advantages over their single-robot counterparts [6]. Their ability to execute tasks in parallel enables them to perform tasks more efficiently than a single robot or accomplish tasks that are impossible for a single robot to carry out. Furthermore, due to distributed sensing, MRSs have the advantage of a wider range of sensing than that of a single robot. Distributed actuations enable them to perform actions at different places at the same time. Compared to a single robot, an MRS can also be more fault tolerant under certain conditions, as the failure of a single robot within the group does not always result in mission failure.

In this paper, we discuss search and tracking algorithms for swarm robotic systems (SRSs), which are basically MRSs with some special properties, mainly related to achieving scalability by means of distributed control and local communication [7]

* Corresponding author.
E-mail addresses: madhubhashi.senanayake@monash.edu (M. Senanayake), ilankaikone.senthooran@monash.edu (I. Senthooran), jan.barca@monash.edu (J.C. Barca), hoam.chung@monash.edu (H. Chung), joarder.kamruzzaman@federation.edu.au (J. Kamruzzaman), manzur.murshed@federation.edu.au (M. Murshed).

http://dx.doi.org/10.1016/j.robot.2015.08.010
0921-8890/

and emergent global behaviours. Besides the advantages common to all MRSs, SRSs have additional merits to them due to their particular characteristics. Robot swarms have high redundancy and are comprised of relatively simple robots compared to the complexity of the task at hand. Furthermore, they use distributed control with the use of local rules and local communication. These characteristics of robot swarms make them highly robust, scalable and flexible [8,9].

Even though there are several recent review and survey papers on swarm robotics (SR) [10–15], all of these broadly review all different works related to SR rather than go into detail on one particular task or application area. This paper is aimed at researchers new to the field of SR who have decided to solve the specific problem of search and tracking. Although not an exhaustive review, this paper will serve as an overview and quick reference guide to help researchers easily locate some seminal work related to their problem at hand, to start off their research.

In Section 2, we give an overview SR and explain features, highlighting the suitability of SRSs for search and tracking applications. In Section 3, we introduce certain characteristics upon which the search and tracking problem can be defined. In the following sections, we refer back to these characteristics in order to explain the different problem formulations used in each study. In Section 4 we explain and discuss algorithms for target search and tracking in SRSs under two categories. Section 4.1 details algorithms based on swarm intelligence and Section 4.2 describes algorithms based on other methods. Comparison of the algorithms based on the criteria relevant to SRSs are given in Section 5 and finally concluding remarks are presented in Section 6.

2. Swarm robotics—an overview

SR is the application of swarm intelligence (SI) principles to the control of groups of robots. Although there is some debate about this definition of the term, there is a consensus that this is how the field of SR came into existence [16,7]. The term 'swarm intelligence' was introduced by Beni and Wang in the context of cellular robotic systems in 1989 [17]. It was before this that Fukuda introduced the first cellular robotic system CEBOT [18, 19], the earliest attempt at developing a collaborative team of robots. CEBOT is a distributed system made of many indiscriminate autonomous robots which have limited intelligence on their own, and worthy of being regarded as one of the earliest SRSs.

The essential characteristics of SI consist of emphasis on decentralised local control and local communication, and on the emergence of global behaviour as the result of self-organisation [7]. The SI approach to mobile robot control was motivated by three main advantages, namely, scalability, flexibility and robustness [8].

- **Scalability:** Scalability of an SRS is defined as the ability to operate with larger or smaller numbers of individuals without impacting performance considerably [15,9]. The use of local sensing and communication is the main reason for scalability in SRSs [11].
- **Flexibility:** Flexibility refers to the ability of an SRS to adapt to new, different, or changing requirements of the environment [15]. When the problem changes, the system has to be flexible enough to promptly respond to the new problem. In swarms, flexibility is promoted by redundancy, simplicity of the behaviours and mechanisms such as task allocation [11] and stochasticity.
- **Robustness:** Robustness can be defined as the degree to which a system can continue to function in the presence of partial failures or other abnormal conditions [15]. SRSs are more reliable due to high redundancy. Any individual is expendable as others can compensate for their loss. Due to their simple and minimalist design, the individual robots are less prone to failures. Because of decentralised control with either no leaders or intermittent/replaceable leaders, losing an individual robot or even a group of them will not drastically affect the overall operation of the swarm. Furthermore, the distributed sensing makes the system less susceptible to noise [9].

It is often rather difficult to define a clear classification of MRSs, especially the difference between SRSs and other distributed MRSs. This is because they share several common features, for example, the use of autonomous robots with distributed control, fault-tolerance and robustness [8,20]. The main dividing point between the two is that achieving scalability remains the core focus of SR [7]. But as the field of SR has evolved, biological inspiration and stigmergic communication are no longer common to all SRSs [7].

In [9], Şahin defines SR as:

> *the study of how large number of relatively simple physically embodied agents can be designed such that a desired collective behaviour emerges from the local interactions among agents and between the agents and the environment.*

Even though this definition does cover some of the special characteristics of swarm robotics, [9] also provides a set of supplementary criteria that can be used for measuring the degree to which an MRS falls under SR.

1. A swarm should consist of autonomous robots able to sense and actuate in a physical world. This excludes software agents that exist purely in a virtual world, and even sensor networks that do not have physical actuation abilities.
2. The study should be relevant for the coordination of large numbers of robots. It should be noted that this does not exclude the studies that are carried out with smaller number of robots as long as they aim for scalability.
3. The system should consist of relatively few homogeneous groups of robots with a large number of robots in each group to preserve redundancy.
4. The robots should be relatively incapable or inefficient with respect to the task at hand. Thus their performance should improve when they cooperate.
5. The robots should only have local and limited-range sensing and communication abilities. This constraint ensures that the team's coordination is distributed and promotes scalability.

The two-dimensional modular robot 'Slimebot' [21] is a good example of an SRS and an example of how a swarm of robots can adapt as a response to its environment. Slimebot has been designed by taking the slime mould as a model living system and modelled it as an 'embodied' coupled non-linear oscillator system. It consists of many identical modules with simple motile functions. It is equipped with a fully decentralised algorithm able to control the morphology of the modular robot in real-time according to the encountered environment, such as obstacles. One of the most significant features of their approach is that they exploit 'emergent phenomena' stemming from the interplay between control and mechanical systems. Their results show high adaptability, high scalability and high fault tolerance [22].

SR algorithms must fit and make full use of the features of SR [10]. The algorithms should scale well with the number of robots as well as targets, and should neither use a highly heterogeneous population nor use operations that are impossible for real robots (e.g., spawn). Also the algorithms should not rely on global information. As the individual robots in a robotic swarm should be simple with limited resources, the algorithms should not involve heavy computations, nor require a lot of memory and communication.

SRSs have abilities that are particularly beneficial in application domains of target search and tracking. For example, their distributed sensing ability is useful in environmental monitoring

and surveillance, their ability to operate in environments that are hazardous or inaccessible to humans is useful in search & rescue and disaster situations and their scalability is useful since these tasks can scale up or down [9].

In spite of these potential advantages of SRSs, there are also several serious challenges involved with their design. A centralised system can be realised in a much simpler structure than a distributed one [23]. Moreover, in the centralised version, an optimised plan is always guaranteed due to the presence of a single leader who can make an optimal decision by evaluating all the relevant information gathered from the team members [24]. When this central decision-maker is taken away, it is extremely difficult to achieve even the most basic properties that are expected from an MRS, such as stability, coordination and coverage [23]. The difficulty in predicting the emergent swarm behaviour makes research in the area of SR immensely challenging. However, the potential advantages of using such systems in the real-world make it a worthwhile effort.

3. Search and tracking problem variants

In the literature, researchers have considered various problem setups when addressing the problem of target search and tracking. These vary in certain parameters and assumptions used, which in turn may also narrow down the focus of the study to certain sub-problems. In this section, we discuss the differences among these problem variants using a taxonomy adopted from [25].

3.1. Number of targets

The problem of target search and tracking may be divided into two main scenarios depending on the number of targets to be searched or tracked: single target and multiple targets. When tracking a single target with any MRS, the main focus would be on sensor data fusion from multiple trackers in order to improve the target *state estimation* accuracy [25].

The multiple targets scenario can be viewed as an extension of the single target case, where many other uncertainties come into play. For example, the number of targets may be unknown, or may even vary with time. But even when the number of targets is known and constant, there is still uncertainty in sensor measurements, as it could be coming from any one of the targets. This is the problem of *data association*. And unlike in the single target scenario, robots need to spread themselves appropriately among the multiple targets, which calls for a *task allocation* method.

The ratio between the number of targets and trackers is another important characteristic that influences the solution approach [25]. For example, when the targets significantly outnumber the trackers, it may not be possible to track all the targets all the time, and the objective may be maximising the average number of targets that are being observed by at least one robot throughout the mission [4]. And another possible approach would be to group targets into clusters and track those clusters instead of tracking them individually [26]. When it comes to tracking a large number of moving targets as in a crowd of people or a herd of animals, observing each individual separately is neither realistic nor necessary. People in a crowd or animals in a herd tend to move towards common destinations collaboratively as they can navigate more accurately than when alone [27,28]. Therefore it would be more efficient to track a cluster of targets in such a scenario. On the other hand, when the number of targets are significantly less than the trackers, it would be possible to track all the targets all the time, and small subgroups of robots may be formed and assigned to each target [29,30]. When the targets and trackers are equal in number, each target may be assigned to one robot, either dynamically or during initialisation in a static manner [31].

3.2. Mobility of targets

According to the mobility of the targets, the problem becomes either searching for stationary targets or tracking moving targets. While the stationary target case has been studied extensively in the SR community [32–34], less work has been done considering mobile targets [35,36]. In the case of stationary targets, the only uncertainty is of noisy observations, i.e. there might be false alarms or missing measurements. But for moving targets, there is additional uncertainty in target motion.

The mobility mode of the targets should also be considered, for example, the target may be moving on the ground, swimming under water or flying in the air. As SR is a relatively new research area, most of the works in target search and tracking so far has been tested under controlled conditions, and has focused on ground moving targets moving on a 2D plane.

3.3. Mobility of trackers

Although trackers in wireless sensor networks may be mostly stationary, in the case of a robotic swarm, they are always mobile. But the mobility mode of the trackers highly influences the problem solution. It governs the trackers' view of the world as well as speed and agility of motion. The trackers may be the same as the targets, for example ground moving trackers tracking ground moving targets, flying trackers tracking flying targets, etc. Alternatively, the mobility mode of the targets and trackers might be different, for example flying trackers can be used to track ground moving targets.

3.4. Complexity of environment

Complexity of the environment is an important factor governing the design of an MRS, because a robot's interactions with other robots and with the environment play crucial roles. In the case of an *open space*, the only interactions to take into account are those among the trackers and targets. In *structured environments*, such as office-type indoor environments, the structure of the environment can be exploited for target detection or the motion planning of the robots. However, in *unstructured environments*, i.e. *cluttered environments*, occlusion caused by the environment structure should be taken into account as uncertainties in sensor measurements. The environment may also affect the mobility of the trackers and targets, due to uneven terrains, obstacles and in dynamic environments, environmental changes such as wind forces.

3.5. Prior knowledge of target motion

When tracking mobile targets using mobile robots, any prior knowledge on target motion would greatly help in predicting the next position of the target so that the robots' movements can be controlled accordingly. When the motion of the target is completely known, its motion model is said to be 'deterministic' [25]. The most classic example is in missile tracking applications, where the missile is known to follow a trajectory governed by the laws of physics [37]. In [38], simulation experiments were run to demonstrate how the integration of target dynamics into sensor planning improves tracking performance. The experiments involved a group of ground robots tracking an aerial target, whose motion was deterministic. The target motion is called 'probabilistic', when the prior knowledge of targets' motion can be modelled with random variables [25]. For real-world applications however, there is often no priori information about target movements. In these cases, the alternatives are to either assume a simple motion model [39,38] or a random motion model such as Brownian motion [40].

Table 1
MR decision making as optimisation problems.

	Objective	Metric to optimise
Task allocation [43,33,34]	Map a set of robots to a set of tasks	Optimise overall system utility. Here, "utility" refers to a combination of the quality at which a robot can execute a given task, and the cost it incurs in executing that task (e.g., power consumption) [43].
Path planning [44–47]	Generate paths for multiple robots	Minimise a performance metric e.g., combined robot path lengths [45], combined travel times for robots to reach their respective goals [47], combined energy use [46].
Formations [35]	Enable robots to move into a desired formation, or to maintain a specific formation, while moving through the environment	Minimise the error between each current robot position and that robot's assigned position in the formation.
Target tracking or observation [4,35]	Control cooperative robot motions to ensure that a group of targets remains under observation by the robots	Optimise a combination of the time in which targets are under observation and a robot cost function [4].

3.6. Type of cooperation

The cooperation among the swarm members is essential to achieving the desired team performance in an SRS, which is superior to the mere sum of individual performances. The robots are able to make two different types of improvements on their overall performance via cooperation: (1) uncertainty reduction and (2) target allocation. The first kind of cooperation is used in the single target scenario, by combining measurements from multiple sensors for a more accurate estimate of the target position than is possible by a single sensor alone. In the case of target search and tracking using an SRS, sensor observations from different robots can be combined in order to estimate the current target location and velocity. Powers et al. [41] approached the problem of cooperative multi-robot (MR) tracking of multiple moving targets, focusing on sensor fusion. Target allocation is used in the multiple target scenario, for improving tracking performance by allocating targets to the trackers who are in the best position for tracking them. This can be seen as a Multi Robot Task Allocation (MRTA) problem, where the goal is to assign tasks to the robots in a way that the global objective is achieved more efficiently through cooperation [34]. For the problem at hand, the 'tasks' would be individual targets or clusters of them and the goal objective would be to track them reliably and efficiently.

3.7. Coordination among multiple trackers

In order to reap the maximum benefits from the cooperation among the robots, a good coordination strategy is essential. Robot coordination strategies can be broadly divided into two main categories, namely, explicit coordination and implicit coordination [25]. In explicit coordination, the behaviour of one robot can be influenced by another robot via explicit communication [25]. In implicit coordination, the individual robots make independent decisions on how to behave, based on the information it gathers through its own observations and communication with others [25]. When using explicit communication, the accuracy of the exchange of information between robots is guaranteed. However, the communication load of a system will increase with the number of robots, possibly deteriorating system performance [42]. When using implicit communication, although the information obtained by a robot is not completely reliable, the stability, reliability and fault tolerance of the overall system are improved [42].

4. Search and tracking algorithms for swarm robotic systems

As mentioned earlier in Section 2, features of SRSs make them very well suited to target search and tracking. In this section, we discuss search and tracking algorithms that have been or potentially can be used in SRSs. We classify these algorithms into two main types: one inspired from SI algorithms and the other inspired from other methods. These two categories of algorithms have been presented separately in the following subsections.

4.1. Search and tracking algorithms based on swarm intelligence

It is easy to see the analogy between swarm optimisation algorithms and SR search algorithms. They both search for 'best locations' within some search space using swarms. In fact, as Parker [48] explains, all fundamental MR interaction skills, including target tracking, involves a decision-making process that can be formulated as optimisation problems as given in Table 1. Thus it is evident that SI algorithms can be applied to finding near-optimal solutions for these problems.

Parker [48] also notes that these optimisation problems are typically not treated as *global* optimisation problems as they are known to be NP-complete. The necessity of real-time responses by the robots leaves insufficient time to calculate globally optimal solutions, unless the problem is very small-scale. Therefore distributed methods that incorporate only local cost/utility metrics are normally used even though they can only approximate the global solution. These sub-optimal solutions often are sufficient for practical applications.

As SI algorithms essentially emphasise on decentralised local control, local communication, and on the emergence of global behaviour as the result of self-organisation [7], they naturally fit and make use of the main features of SRSs. This can be seen as the main reason why many of the existing SR search and tracking algorithms have been based on prominent SI algorithms.

Search and tracking algorithms presented in the remainder of this subsection are categorised by the original SI algorithm they are inspired by. The robotic algorithms discussed herein have mainly used ideas from SI for modelling the behaviours of individual robots, where each robot is treated as an agent/particle in the corresponding SI algorithm.

4.1.1. Particle swarm optimisation

Particle Swarm Optimisation (PSO) is an SI algorithm which was developed by Kennedy and Eberhart [49,50] in an attempt to graphically simulate the flocking behaviour of birds [51]. PSO models a set of potential problem solutions as particles that are flown through a problem space, with attractions to positions at which best results (fitness) are achieved. A particle updates its velocity and subsequently its position in the search space according to its own *personal best* position and also the overall best position achieved within its neighbourhood, which may be global [50] or local [51]. The velocity (v) and position (x) updates of the ith particle performed at the kth step of the PSO algorithm are

$$v_i^k = w v_i^{k-1} + c_1 r_1 (p_i - x_i^{k-1}) + c_2 r_2 (p_n - x_i^{k-1}) \quad (1)$$

and

$$x_i^k = x_i^{k-1} + v_i^k, \quad (2)$$

where v_i^k and x_i^k represent velocity and position vectors for the ith particle in the kth time-step, p_i represents the personal best position of the ith particle and p_n represents the overall best position of all particles within the neighbourhood. c_1 and c_2 are constants called cognitive and social scaling parameters respectively, and r_1, r_2 are random numbers drawn from a uniform distribution. The parameter w is termed *inertia weight* and was introduced in [52] to control how much the current velocity of the particle contributes to its velocity in the next iteration. It plays the role of balancing the global search and local search [52].

The original PSO algorithm was meant for solving global optimisation problems. Therefore it can be easily adapted for searching a single stationary target using multiple robots. Pugh and Martinoli presented the earliest MR search algorithm based on the principles of PSO [32], considering the stationary single-target case. They use a one-to-one mapping between particles in the PSO swarm and robots, with modifications to handle real world constraints. These include obstacles as well as limitations on movement and communication ranges. Like the particles in the SI algorithm, each robot has perfect knowledge of its position and heading in a global coordinate system. The main difference in this robotic algorithm is that each robot only considers other robots within its fixed communication range, leading to dynamic neighbourhoods as robots move.

The above-mentioned robotic algorithm is then modified to work on robots that are not aware of their global position. Because odometric information is very noisy, the robots can only know their current and immediately previous locations with reasonable accuracy. Therefore the strongest detection is limited to be either the current or immediately previous detection. If the current detection is stronger, there is no personal best component to the modification of the velocity. But if the last detection was stronger, the personal best component is in the direction towards its previous position. Because the robots have a short memory, they only communicate their current detections with each other.

Through simulation experiments, the above two robotic search algorithms are tested with varying numbers of robots and communication ranges. The authors [32] report that although the robots succeed in congregating around the target, they do not converge to stable positions. They suggest possible improvements such as decreasing the inertia weight linearly during the search and having robots that detect very strong signals stop exploring in order to record that position and serve as a constant beacon for others. Based on the fact that PSO has been used successfully on problems which have multiple optima as well of dynamic problems, they suggest that PSO may be adapted for MR search in more complex scenarios such as multiple targets and also in dynamic environments like in odour localisation in a natural situation [53].

The use of multiple swarms is the main method used to adapt PSO to solve multi-modal problems. Species-based PSO (SPSO) [54] and Niching PSO (NichePSO) [55] are such variations of the PSO algorithm that use subswarms to locate multiple optimal solutions for multi-modal optimisation problems. The multi-swarm idea has also been used in SR solutions to multi-target search. One such example is [31], where an MR PSO search algorithm is proposed for finding a known number of stationary targets within an indoor environment. In the problem addressed, each target was equipped with a mobile phone and robots were equipped with sensors to detect the RF signals emitted by the mobile phones. Each robot was assigned to a particular target at the start of the search. The robots assigned to the same target formed subswarms that stayed in their local neighbourhood throughout the search, and exchanged best detection positions with each other. In addition to the multi-swarm approach, they also addressed the problem of overshooting targets by using an adaptive Received Signal Strength (RSS) weighting factor in the velocity calculations to slow down the robot as the target is approached.

The clustering PSO algorithm (CPSO) presented in [30] was aimed at addressing dynamic optimisation problems where locating and tracking multiple changing optima over time is important. CPSO starts from an initial swarm called the 'cradle swarm', and employs a hierarchical clustering method to create subswarms. This enables CPSO to assign particles to different promising subregions, adaptively adjust the number of subswarms needed and automatically calculate the region for each subswarm. Even though this algorithm has not been implemented on a robotic system, it seems a promising candidate to be adapted into an SR algorithm.

4.1.2. Bees algorithm

Bees Algorithm (BA) [56] models the foraging behaviour of a colony of bees for the richest and closest food source. The original algorithm by Pham et al. [56] was a combination of neighbourhood search and random search, with the goal of finding a single value which represents the global optimum. Thus, if applied to target search, this algorithm would only be able to locate a single target.

Jevtić et al. [33,34] presented a distributed bees algorithm (DBA), which was a modified version of the original BA suitable for multi-target search and coverage in an unknown area. The objective of this algorithm is distributing a swarm of robots in the area so that targets with higher fitness attract more robots. Considering this task at hand, they identified two issues with the original BA to improve on. The first issue is the centralised nature of the selection of the best targets and recruitment of bees, and the second is the lack of collective component in function minimisation. In their method, when a robot found a target (say the ith target), it was automatically assigned to it. This robot then communicated information on the estimated target location (x_i, y_i) and the quality of the target (q_i) to other robots in its range. Unassigned robots that received information about different targets, calculated their "utilities" with respect to each of those targets. Utility (p_i^k) is the probability that the kth robot is assigned to the ith target, and is dependent on the target's quality (q_i) and the associated cost, which is the kth robot's distance to the ith target (d_i^k). Utility of the kth robot with respect to the ith target is calculated by

$$p_i^k = \frac{{q_i}^{\alpha} {\eta_i^k}^{\beta}}{\sum_{j=1}^{M} {q_j}^{\alpha} {\eta_j^k}^{\beta}}, \tag{3}$$

where M is the number of available targets, and α and β are control parameters used to bias the decision-making mechanism towards the quality of the solution or its cost, respectively. η_i^k denotes the visibility of the ith target to the kth robot, and is defined as the reciprocal value of the Euclidean distance between the robot and target.

$$\eta_i^k = \frac{1}{d_i^k} = \frac{1}{\sqrt{(x_i - x_k)^2 + (y_i - y_k)^2}}. \tag{4}$$

Once a robot has these probabilities for each target, it then probabilistically determines which target to go to. This decision making process is referred to as the *roulette wheel selection method*, which is very popular in genetic algorithms. In [33], results of the experiments with a few real robots were presented and in [34], results of simulation experiments to prove that DBA is highly scalable in terms of the number of robots and targets, and also adaptable to a non-uniform distribution of target qualities were presented.

4.1.3. Artificial Bee Colony Optimisation

Artificial Bee Colony (ABC) algorithm, introduced by Karaboga [57] is an optimisation algorithm based on a model of the foraging behaviour of a honeybee colony, for solving multidimensional and multimodal optimisation problems. Unlike other swarm intelligence algorithms where individuals in the swarm represent candidate solutions, in ABC, the food sources represent possible solutions while bees act as variation agents responsible for generating new sources of food.

In the model, the colony of artificial bees consists of three groups: (1) employed bees that are linked to a particular food source, (2) onlookers who observe the waggle dance of the employed bees within the hive to select a food source and (3) scouts who search randomly for new food sources. Initially, all food source positions are discovered by scout bees. Thereafter, employed bees and onlooker bees continue to exploit the food sources until they are ultimately exhausted. Once the food source is exhausted, the employed bee which was exploiting it becomes a scout bee and searches for further food sources once again [58].

The search process of the ABC algorithm is robust, as the exploration and exploitation processes are carried out simultaneously: onlookers and employed bees perform exploitation of the search space, while the scouts perform the exploration [57].

Similar to other SI algorithms like PSO and Ant Colony Optimisation, ABC has also been used for path planning in mobile robots. One such example is [45], where the ABC algorithm was used for path planning of micro-robots used in drug delivery. In the first step of their method, initial solutions consisting of velocity and angle for each robot are generated, and treated as 'food sources'. Employed bees update these food sources, and the micro-robots use the food sources generated by the employed bees to update their positions. The onlookers select a food source probabilistically, where the food sources with better fitness have higher chances of being selected. The probability that a food source will be selected is given by

$$p_i = \frac{f_i}{\sum_{n=1}^{N} f_n}, \tag{5}$$

where N is the number of food sources and f_i is the fitness value of food source i.

Another example is [59], where the ABC algorithm was combined with the time rolling window strategy [60] to design a novel path planning method for a mobile robot in a dynamic environment. They show through simulations that the proposed method is able to effectively avoid obstacles, static and dynamic, and the planning method is suitable for real-time applications.

4.1.4. Ant Colony Optimisation

The Ant Colony Optimisation (ACO) algorithm developed by Dorigo et al. [61,62] is based on the foraging behaviour of some ant species: how they seek the shortest paths between the nest and food sources. Ants lay down pheromones as they wander randomly searching for food sources, and then use these trails of pheromones to guide them back to the nest along the shortest path. The shortest path is found from the pheromone levels alone. As pheromones evaporate with time, its level on longer paths become lower than those on shorter paths. For the problem of target search and tracking, algorithms based on pheromone-trails are mostly used for optimal path planning.

Hoff et al. [63] presented two ant-inspired robot foraging algorithms which allow coordination between robots. This algorithm uses direct communication between the robots instead of using environmental markers. They assume that the robots have limited sensing and communication capabilities and no explicit global positioning. Both algorithms use the concept that each robot can dynamically take on one of the two roles of stationary environment beacon or wandering robot. However, they differ in when the beacon role is chosen and what kind of information the beacon emits.

ACO has also been modified for multiple odour source localisation [64,65]. This modified ACO algorithm includes the three stages, namely, local traversal search, global search, and pheromone update. The first two are extra search modes compared to the original ACO, and are added to improve the search performance of the robot system. There is also an odour source verification procedure that is performed at certain intervals, in order to speed up the search and localisation of the odour sources.

The ACO algorithm has also been applied to the multidimensional assignment problem in target tracking. In [66], a novel data association technique based on an improved ACO algorithm (ACODA) was proposed. Inspired by how ACO is applied to the travelling salesman problem [67], ACODA models each measurement as an ant, each track as a city, and the problem of data association as the food locating by ants. Here, a measurement is taken to be from either a target or a false detection due to clutter. The algorithm is tested in simulation with varying numbers of targets (2–6) moving on a 2D plane in straight line paths. Experiment results show that the ACODA algorithm demonstrates superior performance both in computational time and accuracy.

4.1.5. Bacterial Foraging Optimisation

Bacterial Foraging Optimisation (BFO) was proposed by Passino [68] as a multimodal function optimisation algorithm inspired by the chemotactic behaviour of bacteria such as the *Escherichia coli* (*E. coli*) bacterium in an environment with nutrients [69]. Bacteria movements mainly consists of two motile behaviours: 'run' ('swim'), which is movement in a particular direction, and 'tumble', which is a change in direction. The basic idea behind BFO is natural selection: eliminating bacteria with poor foraging strategies and favouring those with better foraging strategies.

The position displacement of a bacterium during one step of the algorithm is given by

$$\theta_i(j+1,k,l) = \theta_i(j,k,l) + C(i)\phi(j), \tag{6}$$

where $\theta_i(j,k,l)$ represents the position of the ith bacterium at the jth chemotactic step in the kth reproductive loop of the lth elimination-dispersal event, $C(i) > 0, i = 1,2,\ldots,N$ is the basic chemotactic step size that defines the length of steps during runs and $\phi(j)$ is a unit length random direction which is generated within the range $[0, 2\phi]$ during the chemotactic step. The bacterium compares the fitness values at the previous and current positions and makes a decision where to move next. If the current position has a higher fitness value, it swims in the same direction ($\phi(j)$) without tumbling. Otherwise, it tumbles, generating a new random value for $\phi(j)$. At the end of the specified number of chemotactic steps, a reproduction step is executed. During this step, bacteria are evaluated and sorted in descending order of fitness, in order to distinguish the healthy bacteria. The first half of the bacteria are retained and duplicated while the second half is eliminated. After the specified number of reproduction steps, an elimination-dispersal step is executed in order to help hasten the process of optimisation. Here, bacteria are dispersed according to the specified elimination and dispersal probability.

In [70], the BFO algorithm was adapted to chemical concentration map building. They implemented an MRS to search an unknown area in order to find the region with the highest chemical gas concentration as well as to build the gas concentration map. They only implemented the chemotactic step of the algorithm, as the other two steps are not possible for robotic systems. As with the original BFO algorithm, the robots do not communicate their

fitness values with each other, and search on their own. In BFO, the reproduction step is what guides the swarm towards better areas in the search space, but the algorithm in [70] omitted this step and did not utilise any form of communication between the robots to compensate for the lack of cooperation within the swarm. Therefore, the robots failed to gather around the optimal positions and ended up in different positions at the end of the search. But for the problem they addressed, communicating the detections to a remote computer and building a gas concentration map, the algorithm has proven to perform rather well.

4.1.6. Glowworm Swarm Optimisation

Glowworm Swarm Optimisation (GSO) by Krishnanand and Ghose [71] is a distributed algorithm modelled on the behaviour of glowworms, which is capable of capturing multiple local optima of multimodal functions. The special feature of this algorithm is the *adaptive local-decision domain* which enables the automatic partitioning of the swarm into smaller groups, enabling them to converge on multiple sources simultaneously [72]. The agents in this algorithm carry a luminescence quantity called 'luciferin' along with them for stigmergic communication among the members, somewhat analogous to the pheromone deposits in ACO [73]. The main difference is that this luminescence move along with the glowworms, unlike the pheromones that stay at places visited by the ants.

The GSO algorithm starts by placing the glowworms randomly in the workspace so that they are well dispersed. Initially, all the glowworms contain an equal quantity of luciferin. Each iteration consists of a luciferin-update phase followed by a movement-phase based on a transition rule [73]. The luciferin update rule is given by

$$l_j(t+1) = (1-\rho)l_j(t) + \gamma J_j(t+1), \tag{7}$$

where $l_j(t)$ represents the luciferin quantity in agent j at time t, ρ is the luciferin decay constant ($0 < \rho < 1$), γ is the luciferin enhancement constant, and $J_j(t)$ represents the value of the objective function at the location of agent j at time t. The luminescence level of a glowworm is an indication of the net improvement it has made by traversing from its initial location to the present location [71]. The decaying nature of the luciferin (first term in Eq. (7)) indirectly allows the agents to escape inferior regions and move towards promising regions of the objective functions space [73].

During the movement-phase, each glowworm decides using a probabilistic mechanism to move towards a neighbour that has a higher luciferin value than its own. The glowworms have a higher probability of moving towards the brightest neighbours. Once a glowworm i decides to move towards a brighter glowworm j, it moves to the next position

$$x_i(t+1) = x_i(t) + s\frac{x_j(t) - x_i(t)}{\|x_j(t) - x_i(t)\|}, \tag{8}$$

where $x_i(t)$ represents the location of glowworm i at time t and s is the step size. A suitable function is chosen to update the local-decision range, enabling each glowworm to select its neighbours in such a way that its movements get biased towards the nearest source.

A distributed algorithm based on a GSO has been applied for localising radiation sources using a team of mobile robots [73]. These robots perform subtasks such as relative localisation of neighbours, selection of a leader among current neighbours, updating of the associated luciferin and local-decision range, and making a step-movement towards the selected leader. As the local-decision range (r_d^i) update rule proposed in [71] results in oscillatory behaviour, the following new update rule is proposed.

$$r_d^i(t+1) = \begin{cases} r_d^i(t) + \beta_1|N_i(t)|, & \text{if } |N_i(t)| \leqslant n_t, \\ r_d^i(t) - \beta_1|N_i(t)|, & \text{otherwise,} \end{cases} \tag{9}$$

where β_1 and β_2 are constant parameters and n_t is an explicit threshold parameter used to control the number of neighbours at each iteration. It is reported that a substantial enhancement in performance is achieved by using this new rule. However, the algorithm is validated through experiments using only four real robots and one sound source.

GSO has been proven effective for pursuing multiple mobile targets [36] using simulation results for single and two source cases. Through these numerical experiments, upper bounds on the relative speed of a moving source below which the pursuers succeed in chasing the source are identified. However, as these simulations consider agents as points, agent collisions are ignored.

In a subsequent study [74], real-robot-experiments using a set of four wheeled robots are carried out in order to assess the GSO algorithms suitability for multiple source localisation tasks. Similar to the PSO-inspired robotic search algorithm [32], the original SI algorithm is modified in order to make it suitable for a robotic implementation. The changes are made taking into consideration, real world constraints such as limitations on linear and angular movements, collision avoidance and special leapfrogging behaviour where agents move over one another to perform local search. However, despite the aim of localising multiple signal sources simultaneously, they only conduct real-robot-experiments source localisation using two robots and a single light source. They do claim however that their GSO-based approach can be extended to multiple signal source localisation by using a larger number of these robots.

4.1.7. Firefly Algorithm

Firefly Algorithm (FA) was formulated by Xin-She Yang [75,76] based on the flashing pattern of tropical fireflies. It is very similar to GSO in its basic principle, i.e. fireflies are also attracted to 'brighter' neighbours. However, unlike the glowworms, the fireflies glow with intensities directly proportional to the value of the objective function at their current positions. The attractiveness of a firefly is proportional to its brightness, and this decreases with the distance at which it is observed. The variation of attractiveness β with distance r is given by

$$\beta = \beta_0 e^{-\gamma r^2}, \tag{10}$$

where β_0 is the attractiveness at $r = 0$ and γ is the fixed light absorption coefficient of the medium. The movement of firefly i which is attracted to a brighter firefly j is given by

$$x_i^{t+1} = x_i^t + \beta_0 e^{-\gamma r_{ij}^2}(x_j^t - x_i^t) + \alpha_t \epsilon_i^t, \tag{11}$$

where x_i^t and x_j^t denote at the positions of fireflies i and j at time t, respectively. The second term is due to the attraction to the brighter neighbour and the third term is randomisation. α_t is the randomisation parameter which can be tuned to scale the random component and ϵ_i^t is a vector of random numbers. This random vector is typically drawn from either Gaussian or uniform distribution, but can also be extended to other probability distributions [76]. Unlike in GSO where one brighter neighbour is probabilistically chosen to move towards, in FA, each firefly is attracted to all other brighter fireflies and consequently moves towards an average position.

The main advantages of FA are the automatic subdivision and the ability to deal with multimodality [77]. Because the algorithm is based on attraction and the attraction decreases with distance, the population can automatically divide into subgroups. This enables the swarm to simultaneously cover different modes or local optima, among which the best global solution can be found. Given a large enough population size with respect to the number of modes, the swarm would be able to find all optima simultaneously.

This is what makes FA very efficient at multimodal optimisation problems.

FA has been adapted for solving path planning problems [78]. The new path planning method based on firefly algorithm that was proposed in [79], improved the solution quality and convergence speed by making the randomisation parameter and light absorption parameter adaptive. In addition, a new modified FA (MFA) for solving the path planning problem for uninhabited combat air vehicle (UCAV) has also been developed [80]. In this new algorithm, a modification was applied for exchanging information between top fireflies during the process of light intensity updating, which accelerated the global convergence speed whilst preserving the strong robustness of the classical firefly algorithm [78].

FA has also been modified for optimisation in dynamic environments [81–84] and tested on the moving peaks benchmark problem [85]. Dynamic environment is usually solved by FA using the multi-swarm population scheme, which can respond quicker to the changing environment [81,82]. Additionally, the adaptation to changing environment is faster, when the control parameters of FA are taken into account. In [81], FA was hybridised with learning automata that was employed to tune the control parameters for dynamic environments. The adaptation of control parameters has also proven useful for solving the multi-objective problems [84].

4.1.8. Biased Random Walk

Based on Biased Random Walk (*BRW*) [86], Dhariwal et al. [87] present an MR algorithm for locating stationary gradient sources with time-varying intensities. Similar to BFO, this algorithm is also inspired by the chemotactic movements of bacteria. When the robot does not detect anything, it moves a fixed-length distance in a particular direction before randomly changing its heading (*tumbling*). But when there is a positive change in a robot's successive detections, it decreases its tumbling frequency and consequently increasing its run length.

This method has a very small memory requirement, as the robot is only required to remember its last sensor reading. The processing needed is also minimal, as the robot only compares its successive sensor readings at each time step. The algorithm was verified through extensive simulations for different source models and robot deployment strategies. In the experiments with real robots, they employed photo sensors on robots to detect a light source inside a test bed with an overhead camera to detect the positions of the robots. It was observed that all targets are tracked simultaneously, and that sources with higher intensities are tracked by a larger fraction of robots. The main drawback of the algorithm is that it does not use any form of communication among the robots, and thus it is possible for all the robots to converge around the same source.

4.2. Search and tracking algorithms based on other methods

This subsection discusses some non-SI based algorithms for target search and tracking applications.

4.2.1. Distributed Kalman Filter (DKF)

Wang and Gu [88] addressed the problem of tracking a single moving target with a group of robots equipped with vision cameras to detect the target. Their distributed tracking algorithm comprises of a distributed Kalman filter (*DKF*) for target position estimation and a distributed flocking algorithm for robot motion control. Although target position estimation using multiple sensor data is possible with a centralised Kalman filter (CKF) [89], it needs to be executed at a unit where all sensor measurements are available. As it is neither scalable nor robust for a single robot to receive measurements from the entire group of robots, they propose a DKF as an alternative. In the proposed DKF, each robot would only communicate information with its neighbours, and the DKF is executed locally using its own measurements and neighbour information. The information exchanged between neighbouring robots comprises of the sensor measurement and the predicted result. They considered the case where only a part of the robot group could detect the target, and modelled the neighbour's information as additional measurement. This modelling results in an *implicit consensus algorithm* among the robots which can directly detect the target, and also enable the robots which cannot detect the target to use the neighbour's information to update its target position estimate and maintain a stable estimated result. In the next step, i.e. robot flocking control, these target position estimates are used as a tracking destination in an artificial potential function approach, and the entire group of robots is made to follow and track the target. The flocking controller comprises of two components: tracking control (cohesive potential function) for following the target and separation control (repulsive potential function) for robot collision avoidance.

4.2.2. Potential fields

A-CMOMMT. Parker [4] was the first to formalise the problem of cooperative multi-robot observation of multiple moving targets (*CMOMMT*). She approached the two dimensional version of this problem as an optimisation problem of maximising "the collective time during which each target is being observed by at least one robot during the mission". Her distributed control algorithm *A-CMOMMT* is inspired by *potential fields*, and considers two types of local force vectors that influence robots' movements: attractive forces (f) towards nearby targets and repulsive forces (g) towards nearby robots. The direction of movement for robot ith robot is given by

$$\sum_{k=1}^{n} w_{ik} f_{ik} + \sum_{j=1, j\neq i}^{m} g_{ij}, \tag{12}$$

where n is the number of targets and m is the number of robots. In order to encourage robots to cover more of the targets, she introduces weights (w_{ix}) to the attractive forces towards targets (f_{ix}). These weights are assigned considering whether a target is being observed by other robots, causing robots to be less attracted to targets that are under observation by others. A robot is considered to be observing a target, if the target is inside the robot's sensing range (Fig. 1). Each robot communicates the positions of all targets within its sensing range to other robots in its communication range. It is assumed that the robots share a global coordinate system. If the ith robot detects that the target O_x is inside the sensing range of another robot, it will set the weighting for that target (w_{ix}) to a low value. w_{ix} is not set to zero as this would increase the likelihood of losing the target. This weight is also dependent upon the estimated target-density in the vicinity, because if targets are sparsely located in the area, the risk of losing track of targets is higher.

After comparing with three other approaches using both simulations and real robots, Parker concludes that her method outperforms the others in situations where targets outnumber robots. For situations of lower number of targets, she suggests that prior knowledge about the number of targets can be used in calculating the weights for the local force vectors. *Local minima* or *trap situations* is a well-known drawback of potential field based control algorithms. For example, as obstacles exert repulsive forces onto the robot, and the sum of all forces determines the subsequent direction and speed of the robot, a U-shaped obstacle will trap it inside, not allowing it to reach the targets. Parker did not address the problem of local minima.

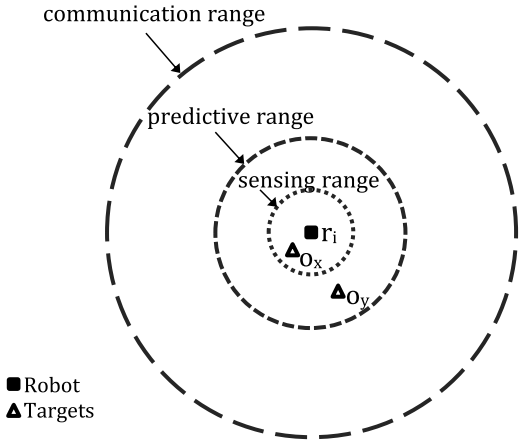

Fig. 1. Defined ranges for the ith robot (r_i): sensing range (the range in which it can directly localise targets), predictive tracking range (the range in which targets localised by other robots can affect its movement) and communication range (the range in which it can communicate with other robots) [4]. Robot r_i knows two types of targets: directly sensed (e.g., O_x) and virtually sensed through predictive tracking (e.g., O_y).

Virtual impedance method for motion planning. Ota et al. [90] proposed a virtual impedance method for motion planning for a distributed MRS in an unknown dynamic environment. The system consists of a hierarchical architecture with two layers. At the upper layer, sub-shortest paths that avoid static obstacles are generated. Then at the lower layer, on-line avoidance is made with virtual impedance against moving obstacles such as other robots.

The virtual impedance method is an extension of the artificial potential field method and is a decentralised on-line planning method [90]. This method determines the motion of each robot by means of three types of virtual forces: (1) the force generated between a reference position of the robot at the present time and the real position of the robot, (2) the force generated between two robots, and (3) the force generated between a robot and an obstacle. Simulation results indicate the effectiveness of the proposed architecture. Authors conclude that although the mutual avoidance of robots in the proposed method has room for improvement, it is practical enough for relatively sparse environments.

4.2.3. Formation-based target following

Lee et al. [35] tackled the problem of coordinated tracking of multiple targets moving unpredictably using a mobile robot swarm. They consider the case when the target is only visible to a portion of the robots. They presented a distributed approach with two algorithms: local interaction and target tracking. The local interaction algorithm enables three neighbouring robots to generate an *equilateral triangle* configuration of a certain predefined size. The target tracking algorithm defines how to find a robot's moving direction towards a desired target, and simultaneously configure the positions of the neighbouring robots into equilateral triangle configuration to the desired target direction. When a robot detects multiple targets, it selects a target to follow based on the relative degree of attraction termed as the favourite vector (f_k) with magnitude $\|f_k\| = \left\|1/d_k^2\right\|$, where d_k denotes the distance between target g_k and the robot. Basically the robot chooses to follow the target that is closest to itself. If a robot detects a target, it chooses its neighbours based on the direction of the target and proximity. When it cannot detect a target, it chooses its neighbours based on its heading and proximity. This method of choosing neighbouring robots to maintain formations with, acts as an implicit consensus algorithm. This allows the robots that detect targets to guide the tracking and the robots who do not directly detect a target to still follow one by following its neighbours.

The computation of target position does not require the memory of previous states. The main advantage of this approach is that it does not rely on some major assumptions which are common in other works. The algorithms do not use robot identifiers, a common coordinate system or explicit communication among the robots. Mathematical proof for convergence properties and scalability of the algorithms are provided, in addition to being implemented and verified through simulations and real-robot experiments. It should be noted that the complexities of their algorithms are quite high compared to SR algorithms, even though they have minimised the computational burden with the selection of equilateral triangles for the formation. As this approach does not use any form of explicit communication among the robots, the robots need to rely on their own observations for coordination. To this effect, a novel proximity sensor termed *Dual Rotating Infrared* (*DRIr*) sensor was introduced. A pair of these was mounted on each robot in order to sense other robots at full 360°.

5. Comparative analysis

The target search and tracking problems using an MRS can be divided into two main sub-problems. This first is target state estimation with a single robot which involves computing the positions and velocities targets in the robot's range of view from its sensor measurements. The second is the coordination of motion between the robots to track more targets over time [25]. Most of the research in the area of SR focus on the latter aspect of the problem.

Table 2 outlines the various parameters defining the problem setups (discussed in Section 3) used in some of the MR search and tracking algorithms described in Sections 4.1 and 4.2. It is evident from the table that only A-CMOMMT [4] has addressed the problem of tracking multiple moving targets that are larger in number than the robot team.

As described in Section 1, in order to reap the maximum benefits of an SRS, certain properties should be maintained. Table 3 outlines the characteristics of the different algorithms introduced in Sections 4.1 and 4.2, emphasising on the necessary properties for them to be fitting for SRSs.

Having no leader increases the robustness of the algorithm because there is no single vital point of failure. It also increases scalability of the algorithm because of the reduced communication between robots. Scalability is the most important property for any distributed MRS. Using only local communication is also important for achieving scalability. Requiring global communication would not only restrict the spread of the robots due to their limited communication range but also cause communication overloading when the swarm size increases. Having no robot identifiers also contribute to preserving system scalability. This is because there is always a limit to the number of unique identifiers that can be generated, be it colours or graphical patterns. Furthermore, assigning of identifiers is a form of centralisation. Having a common coordinate system is also somewhat of a restrictive assumption when it comes to SRSs. Having a very large numbers of robots makes it infeasible for a central positioning system to track them all, and the swarm may operate in locations where GPS and similar systems are unavailable [32].

SRSs emphasise on simplicity of individual robots and achieving complex tasks by working in large numbers. This calls for algorithms that have light enough computational requirements to run on limited hardware resources achieving real-time performance. Generally SI algorithms tend to be less complex, and in many cases, problem size is not directly linked with the algorithm complexity [91]. In the PSO-based algorithms [32,31], computations performed at each time-step are very simple, consisting of only a single comparison and vector addition. The BA-based algorithm

Table 2
Problem classification in various target search and tracking studies.

	Problem Characteristics						
	Number of targets	Targets/trackers ratio	Mobility of targets	Environment complexity	Prior knowledge of target motion	Cooperation	Coordination
Pugh & Martinoli [32]	1	$\ll 1$	Stationary	Empty space	N/A	Target estimation	Implicit
Parker [4]	Multiple	>1	Mobile	Empty space	None	Target allocation	Implicit
Derr & Manic [31]	Known number	≤ 1	Stationary	Cluttered	N/A	Target estimation	Implicit
Wang & Gu [88]	1	<1	Mobile	Empty space	None	Target estimation	Implicit
Jevtić et al. [34]	Multiple	<1	Stationary	Cluttered	N/A	Target allocation	Implicit
Lee et al. [35]	>1	$\ll 1$	Mobile	Empty space	None	Target estimation	Implicit

Table 3
Target search and tracking algorithm comparison.

	Pugh & Martinoli [32]	Derr & Manic [31]	Jevtić et al. [33,34]	Turdeuv et al. [70]	Dhariwal et al. [87]	Parker [4]	Wang & Gu [88]	Lee et al. [35]
Leaderless	✓	✓	✓	✓	✓	✓	✓	✓
Local/no communication	✓		✓	✓		✓	✓	✓
No robot Identifiers	✓		✓		✓	✓	✓	✓
No common coordinates	✓	✓			✓			✓
Simple computations	✓	✓	✓	✓	✓	✓		
No memory of previous states	✓		✓			✓		✓
Mathematically proven properties								✓
Verified through simulations	✓	✓	✓		✓		✓	✓
Verified through real robot experiments			✓	✓	✓	✓	✓	✓

in [33], each robot only calculates its own utilities for each target. Similarly, BFO-based algorithm in [70], the GSO algorithm in [36] and BRW algorithm in [87], also have simple computations. The non-SI algorithms discussed here generally require more computational power than the SI algorithms, with the exception of the A-CMOMMT algorithm. The time update and measurement update calculations of the DKF algorithm and separation control of the flocking algorithm are computationally far more complex compared to the SI algorithms, and this increases with the number of neighbouring robots. The algorithm in [35] also has large number of computational steps including computation of favourite vector for selecting a target to follow, computation for neighbour selection, computation of distances and angles with neighbours for formation control.

The SI algorithms only use local communication or do not use any explicit communication at all. In the two PSO-based algorithms [32,31], the robots only share their detections with their neighbours, which does not cause a lot of data traffic in communication. The BRW algorithm in [87] uses no communication between robots. The non-SI algorithm [88] also uses local communication, however, the robots exchange a lot of information including the detections and the predictions.

In order to search or track multiple targets simultaneously, the robots should be partitioned and assigned to different targets. The highest performing automatic robot partitioning technique can be seen in the GSO algorithms in [73,36], where the adaptive local-decision domain enables the automatic formation of robot clusters that can converge on multiple targets simultaneously. The simplest approach is seen in the PSO algorithm in [31], where the robot partitioning is done in manually by assigning a target to each robot at the start of the search, resulting in subswarms being formed by robots tracking the same target. In the BRW algorithm in [87], partitioning of the robots is done in a random manner as the robots do not cooperate with each other. This partitioning does not always guarantee that all the targets will be covered, as there is the possibility of all the robots converging around the same target. In the BA-based algorithm [33,34], the probabilistic target allocation results in the swarm being partitioned among the targets according to their quality values. In [35] the robots are programmed to follow the closest target, and this results in the robot population being partitioned so that every target is followed.

As SRSs are built up of a large number of robots, the possibility of robot collisions is very high. Therefore adopting an obstacle avoidance mechanism is crucial for achieving a good system performance. There are a few different obstacle avoidance methods incorporated in the works discussed in this paper. The simplest approach is to use robots' on-board sensors to detect immediate obstacles. This is the method used in GSO-based approach in [73] and also the PSO-based approach in [32] which is stated to use Braitenberg obstacle avoidance [92]. Another popular method common to some algorithms is artificial repulsion, where obstacles are modelled as artificial repulsive forces acting on the robots [4]. Even SI-based algorithms like the BFO algorithm in [70] incorporate artificial repulsive forces in their control algorithm in order to avoid obstacles and other robots. In the PSO algorithm [31], when robots calculate their next positions, they check for other robots in their path. Upon detecting a potential collision, the algorithm will calculate a new random direction for the robot and check for collisions again, until a collision-free path is encountered. If a collision-free path is not available, the robot waits for the other robots to move out of its way before proceeding. Both the ABC-based path planning algorithms in [45,59] include obstacle avoidance in each path segment. The micro-robots in [45] detect the boundaries of static obstacles and adjust their trajectories such that they keep a specified distance from them. In the distributed flocking algorithm in [88,35], inter-robot collision avoidance is not explicitly discussed. Even though the formations that the robots maintain would prevent such collisions, it should be noted that maintaining the formations stably is difficult.

In [35], scalability and convergence of the algorithms are proved mathematically for the case where the robots are faster than

the targets. However, most of the algorithms discussed here are not analysed theoretically, and do not provide any mathematical proof of properties like convergence and stability. This however, is an open problem concerning all meta-heuristic algorithms [91]. The efficiency of the algorithms are often demonstrated either through comparison with other algorithms and/or applications to well-known problems, but the mathematical analysis lacks behind [91]. PSO is among the small minority of algorithms for which convergence analysis has been carried out. Even though GSO algorithm includes some theoretical analysis on the clustering behaviour of the swarm, it is not up to the desired level [72].

All of the SI algorithms discussed in this paper have been verified through simulations, and some [33,73] have also been tested using real-robot experiments. The tendency to use simulations in verifying SRSs is due to the unavailability of a large number of robots to carry out real-world experiments. Even the works that include experiments with real robots have only used a small number of robots due to this limitation, and have resorted to simulations in order to test their algorithms with a higher number of robots. All of the non-SI based algorithms discussed [88,4,87,35] have been verified through experiments using simulations and also real robots.

The necessity to sample the gradient of some physical, chemical, biological, or electromagnetic property in order to locate potential sources or entities is common to most target search and tracking applications [72]. However, a good search or tracking algorithm should not be susceptible to plateaus (dead space) in the gradient field. This problem can be avoided by incorporating *randomness* into the algorithm. The BRW method discussed in this paper is a perfect example of this: when the robots sense a positive gradient they take larger steps in that direction, but in the absence of a gradient, the robots move in a fixed length random walk. In the case of GSO however, the glowworms only move towards 'brighter' neighbours and this handicaps them when they are at a dead space in the gradient field. This was demonstrated in [93,94], where GSO was compared against BRW using benchmark datasets having such dead spaces.

In addition to the more popular Brownian random motion, strong support has also been found recently for *Lévy flight* search patterns across a vast array of animals including open-ocean predatory fish [95] and birds [96]. Lévy flights are a special class of random walk with step lengths drawn from the Lévy distribution which has a power-law tail, instead of the normal Gaussian distribution used in Brownian motion. This results in many small-step 'walk clusters' interspersed by longer relocations [96]. Lévy flights have been adopted to improve recent SI algorithms such as GSO [97], FA [98], and Bat Algorithm [99] and even MR search [100].

The problem of tracking multiple moving targets calls for an algorithm that is capable of finding multiple optima in parallel, tracking located optima and also balancing the exploitation of known optima and the search for new optima [101]. In order to find multiple optima in parallel, the algorithm should be able to maintain diversity in the population so as to avoid the whole population converging to one optimum without finding all the optima [85]. Lévy flights trajectory can ensure the diversity of the population against premature convergence and make the algorithm effectively jump out of local minima [99]. Also, it has been observed that past information is often helpful in dynamic optimisation problems [85]. As the current state of the environment and fitness landscape is often similar to previously seen states, the use of past information may make it easier to find promising solutions in the new environment [102]. Furthermore, because past solutions provide additional points to search from after a change, they may help inject diversity into the search process once the search has converged [102].

6. Conclusions and future research

SRSs have the potential to be used for real-world tasks owing to their robustness, flexibility, scalability, as well as cost effectiveness. One such task that has a variety of applications is target search and tracking. This paper has described and compared search and tracking algorithms that are applicable to robotic swarms. Among the different problem setups considered in these works, the most challenging but promising application scenario is the use of swarms of robots to track multiple moving targets.

Tracking multiple moving targets with a robotic swarm is a complex distributed optimisation problem that is both multi-modal and dynamic. Algorithms for this problem should have the ability to simultaneously find multiple targets (optima), track them and balance the exploitation of known targets and the search for new targets. This paper has described several algorithms that provide distributed but approximate solutions to this optimisation problem, including CPSO, GSO and FA. As the complexities of SI algorithms are not directly linked to problem size, they are well suited to SRSs for real-world applications. However, our review reveals that most algorithms do not provide any provable performance guarantees due to the complex nature of swarms in mathematical analysis. Also, quantitative comparison between different SI techniques proved difficult because of the lack of standard benchmarking methods.

References

[1] C.-Y. Chong, D. Garren, T. Grayson, Ground target tracking—a historical perspective, in: 2000 IEEE Aerospace Conference Proceedings, vol. 3, 2000, pp. 433–448.
[2] F. Viani, P. Rocca, G. Oliveri, D. Trinchero, A. Massa, Localization, tracking, and imaging of targets in wireless sensor networks: An invited review, Radio Sci. 46 (5) (2011) RS5002.
[3] A. Yilmaz, O. Javed, M. Shah, Object tracking: A survey, ACM Comput. Surv. 38 (4) (2006) 13.
[4] L. Parker, Distributed algorithms for multi-robot observation of multiple moving targets, Auton. Robots 12 (3) (2002) 231–255.
[5] K. Zhou, S.I. Roumeliotis, Multirobot active target tracking with combinations of relative observations, Trans. Rob. 27 (4) (2011) 678–695.
[6] R.C. Arkin, Behavior-based Robotics, Intelligent Robots and Autonomous Agents, MIT Press, Cambridge, Mass, 1998.
[7] A.J.C. Sharkey, Swarm robotics and minimalism, Connect. Sci. 19 (3) (2007) 245–260.
[8] A. Martinoli, Collective complexity out of individual simplicity: A review of swarm intelligence: From natural to artificial systems, by eric bonabeau, marco dorigo, and guy theraulaz, Artif. Life 7 (3) (2001) 315–319.
[9] E. Şahin, Swarm robotics: From sources of inspiration to domains of application, in: E. Şahin, W.M. Spears (Eds.), Swarm Robotics, in: Lecture Notes in Computer Science, vol. 3342, Springer, Berlin Heidelberg, 2005, pp. 10–20.
[10] Y. Tan, Z. yang Zheng, Research advance in swarm robotics, Def. Technol. 9 (1) (2013) 18–39.
[11] M. Brambilla, E. Ferrante, M. Birattari, M. Dorigo, Swarm robotics: a review from the swarm engineering perspective, Swarm Intell. (2013) 1–41.
[12] J.C. Barca, Y.A. Sekercioglu, Swarm robotics reviewed, Robotica 31 (2013) 345–359.
[13] I. Navarro, F. Matía, An introduction to swarm robotics, ISRN Robot. (2013).
[14] Y. Mohan, S.G. Ponnambalam, An extensive review of research in swarm robotics, in: World Congress on Nature Biologically Inspired Computing, 2009. NaBIC 2009, 2009, pp. 140–145.
[15] L. Bayindir, E. Şahin, A review of studies in swarm robotics, Turk. J. Electr. Eng. Comput. Sci. 15 (2) (2007) 115–147.
[16] G. Beni, From swarm intelligence to swarm robotics, in: E. Şahin, W.M. Spears (Eds.), Swarm Robotics, in: Lecture Notes in Computer Science, vol. 3342, Springer, Berlin Heidelberg, 2005, pp. 1–9.
[17] G. Beni, J. Wang, Swarm intelligence in cellular robotic systems, in: Proceed. NATO Advanced Workshop on Robots and Biological Systems, 1989.
[18] T. Fukuda, S. Nakagawa, Y. Kawauchi, M. Buss, Self organizing robots based on cell structures—cebot, in: IEEE International Workshop on Intelligent Robots, 1988, 1988, pp. 145–150.
[19] T. Fukuda, Y. Kawauchi, Cellular robotic system (cebot) as one of the realization of self-organizing intelligent universal manipulator, in: 1990 IEEE International Conference on Robotics and Automation, 1990, Proceedings, vol.1, 1990, pp. 662–667.

[20] A.J. Sharkey, N. Sharkey, The application of swarm intelligence to collective robots, in: J. Fulcher (Ed.), Advances in Applied Artificial Intelligence, IGI Global, 2006, pp. 157–185.

[21] M. Shimizu, A. Ishiguro, A self-reconfigurable robotic system that exhibits amoebic locomotion, in: IEEE/ICME International Conference on Complex Medical Engineering, 2007. CME 2007, 2007, pp. 101–106.

[22] M. Shimizu, A. Ishiguro, An amoeboid modular robot that exhibits real-time adaptive reconfiguration, in: Proceedings of the 2009 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS'09, IEEE Press, Piscataway, NJ, USA, 2009, pp. 1496–1501.

[23] H. Chung, S. Oh, D. Shim, S. Sastry, Toward robotic sensor webs: Algorithms, systems, and experiments, Proc. IEEE 99 (9) (2011) 1562–1586.

[24] M. Yogeswaran, S.G. Ponnambalam, Swarm Robotics: An Extensive Research Review, in: I. Fuerstner (Ed.), Advanced Knowledge Application in Practice, InTech, 2010, Ch. 14.

[25] B. Jung, Cooperative target tracking using mobile robots, Ph.D. thesis University of Southern California, Los Angeles, CA, USA, 2005, aAI3180426.

[26] F. Armaghani, I. Gondal, J. Kamruzzaman, D. Green, Dynamic clusters graph for detecting moving targets using wsns, in: 2012 IEEE Vehicular Technology Conference (VTC Fall), 2012, pp. 1–5.

[27] J.J. Faria, E.A. Codling, J.R. Dyer, F. Trillmich, J. Krause, Navigation in human crowds; testing the many-wrongs principle, Anim. Behav. 78 (3) (2009) 587–591.

[28] J. Krause, G.D. Ruxton, S. Krause, Swarm intelligence in animals and humans, Trends Ecol. Evol. 25 (1) (2010) 28–34.

[29] C. Li, S. Yang, A clustering particle swarm optimizer for dynamic optimization, in: IEEE Congress on Evolutionary Computation, 2009. CEC'09, 2009, pp. 439–446.

[30] S. Yang, C. Li, A clustering particle swarm optimizer for locating and tracking multiple optima in dynamic environments, IEEE Trans. Evol. Comput. 14 (6) (2010) 959–974.

[31] K. Derr, M. Manic, Multi-robot, multi-target particle swarm optimization search in noisy wireless environments, in: 2nd Conference on Human System Interactions, 2009. HSI'09, 2009, pp. 81–86.

[32] J. Pugh, A. Martinoli, Inspiring and modeling multi-robot search with particle swarm optimization, in: Swarm Intelligence Symposium, 2007, SIS 2007, IEEE, 2007, pp. 332–339.

[33] A. Jevtić, P. Gazi, D. Andina, M. Jamshidi, Building a swarm of robotic bees, in: World Automation Congress (WAC), 2010, 2010, pp. 1–6.

[34] A. Jevtić, A. Gutierrez, D. Andina, M. Jamshidi, Distributed bees algorithm for task allocation in swarm of robots, IEEE Syst. J. 6 (2) (2012) 296–304.

[35] G. Lee, N. Chong, H. Christensen, Tracking multiple moving targets with swarms of mobile robots, Intell. Serv. Robot. 3 (2010) 61–72.

[36] K. Krishnanand, D. Ghose, Chasing multiple mobile signal sources: A glowworm swarm optimization approach, in: Proceedings of the 3rd Indian International Conference on Artificial Intelligence (IICAI-07), Pune, India, 2007, pp. 1308–1327.

[37] T. Kirubarajan, Y. Bar-Shalom, Y. Wang, Passive ranging of a low observable ballistic missile in a gravitational field, IEEE Trans. Aerosp. Electron. Syst. 37 (2) (2001) 481–494.

[38] J. Spletzer, C. Taylor, Dynamic sensor planning and control for optimally tracking targets, Int. J. Robot. Res. 22 (1) (2003) 7–20.

[39] B. Jung, G.S. Sukhatme, Detecting moving objects using a single camera on a mobile robot in an outdoor environment, in: International Conference on Intelligent Autonomous Systems, The Netherlands, 2004, pp. 980–987.

[40] D. Montemerlo, S. Thrun, W. Whittaker, Conditional particle filters for simultaneous mobile robot localization and people-tracking, in: IEEE International Conference on Robotics and Automation, 2002. Proceedings. ICRA'02, vol. 1, 2002, pp. 695–701.

[41] M. Powers, R. Ravichandran, F. Dellaert, T. Balch, Improving multirobot multitarget tracking by communicating negative information, in: L. Parker, F. Schneider, A. Schultz (Eds.), Multi-Robot Systems, in: From Swarms to Intelligent Automata, vol. III, Springer, Netherlands, 2005, pp. 107–117.

[42] Z. Yan, N. Jouandeau, A. Ali Cherif, A survey and analysis of multi-robot coordination, Int. J. Adv. Robot. Syst. 10 (399) (2013).

[43] B.P. Gerkey, M.J. Matarić, A formal analysis and taxonomy of task allocation in multi-robot systems, Int. J. Robot. Res. 23 (9) (2004) 939–954.

[44] L.E. Parker, Multiple mobile robot teams, path planning and motion coordination, in: R.A. Meyers (Ed.), Encyclopedia of Complexity and Systems Science, Springer, New York, 2009, pp. 5783–5800.

[45] A. Banharnsakun, T. Achalakul, R. Batra, Target finding and obstacle avoidance algorithm for microrobot swarms, in: 2012 IEEE International Conference on Systems, Man, and Cybernetics (SMC), 2012, pp. 1610–1615.

[46] D. Michel, K. McIsaac, New path planning scheme for complete coverage of mapped areas by single and multiple robots, in: 2012 International Conference on Mechatronics and Automation (ICMA), 2012, pp. 1233–1240.

[47] M. Kapanoglu, M. Alikalfa, M. Ozkan, A. Yazıcı, O. Parlaktuna, A pattern-based genetic algorithm for multi-robot coverage path planning minimizing completion time, J. Intell. Manuf. 23 (4) (2012) 1035–1045.

[48] L. Parker, Decision making as optimization in multi-robot teams, in: R. Ramanujam, S. Ramaswamy (Eds.), Distributed Computing and Internet Technology, in: Lecture Notes in Computer Science, vol. 7154, Springer, Berlin Heidelberg, 2012, pp. 35–49.

[49] R. Eberhart, J. Kennedy, A new optimizer using particle swarm theory, in: Proceedings of the Sixth International Symposium on Micro Machine and Human Science, 1995. MHS'95, 1995, pp. 39–43.

[50] J. Kennedy, R. Eberhart, Particle swarm optimization, in: IEEE International Conference on Neural Networks, 1995. Proceedings, vol. 4, 1995, pp. 1942–1948.

[51] R. Eberhart, Y. Shi, Particle swarm optimization: developments, applications and resources, in: Proceedings of the 2001 Congress on Evolutionary Computation, 2001, vol. 1, 2001, pp. 81–86.

[52] Y. Shi, R. Eberhart, A modified particle swarm optimizer, in: The 1998 IEEE International Conference on Evolutionary Computation Proceedings, 1998. IEEE World Congress on Computational Intelligence, 1998, pp. 69–73.

[53] W. Jatmiko, K. Sekiyama, T. Fukuda, A pso-based mobile sensor network for odor source localization in dynamic environment: Theory, simulation and measurement, in: IEEE Congress on Evolutionary Computation, 2006. CEC 2006, 2006, pp. 1036–1043.

[54] X. Li, Adaptively choosing neighbourhood bests using species in a particle swarm optimizer for multimodal function optimization, in: K. Deb (Ed.), Genetic and Evolutionary Computation, GECCO 2004, in: Lecture Notes in Computer Science, vol. 3102, Springer, Berlin Heidelberg, 2004, pp. 105–116.

[55] R. Brits, A.P. Engelbrecht, F.V.D. Bergh, A niching particle swarm optimizer, in: Proceedings of the Conference on Simulated Evolution And Learning, 2002, pp. 692–696.

[56] D. Pham, A. Ghanbarzadeh, E. Koç, S. Otri, S. Rahim, M. Zaidi, The bees algorithm—a novel tool for complex optimisation problems, in: D. Pham, E. Eldukhri, A. Soroka (Eds.), Intelligent Production Machines and Systems, Elsevier Science Ltd., Oxford, 2006, pp. 454–459.

[57] D. Karaboga, An idea based on honey bee swarm for numerical optimization, Tech. Rep. TR06, Erciyes University, Engineering Faculty, Computer Engineering Department, 2005, Oct..

[58] D. Karaboga, Artificial bee colony algorithm, Scholarpedia 5 (3) (2010) 6915.

[59] Q. Ma, X. Lei, Dynamic path planning of mobile robots based on ABC algorithm, in: F. Wang, H. Deng, Y. Gao, J. Lei (Eds.), Artificial Intelligence and Computational Intelligence, in: Lecture Notes in Computer Science, vol. 6320, Springer, Berlin Heidelberg, 2010, pp. 267–274.

[60] X. Cai, Y. Li, T. Wu, Dynamic path planning of mobile robots in uncertain environments based on pso and receding horizon optimization, Bull. Sci. Technol. 24 (2) (2008) 260–265.

[61] M. Dorigo, G. Di Caro, L.M. Gambardella, Ant algorithms for discrete optimization, Artif. Life 5 (2) (1999) 137–172.

[62] M. Dorigo, G. Di Caro, New Ideas in Optimization, McGraw-Hill Ltd., UK, Maidenhead, UK, England, 1999, pp. 11–32. Ch. The Ant Colony Optimization Meta-heuristic.

[63] N. Hoff, A. Sagoff, R.J. Wood, R. Nagpal, Two foraging algorithms for robot swarms using only local communication, in: ROBIO, 2010, pp. 123–130.

[64] Y. Zou, D. Luo, A modified ant colony algorithm used for multi-robot odor source localization, in: D.-S. Huang, I. Wunsch, C. Donald, D. Levine, K.-H. Jo (Eds.), Advanced Intelligent Computing Theories and Applications. With Aspects of Artificial Intelligence, in: Lecture Notes in Computer Science, vol. 5227, Springer, Berlin Heidelberg, 2008, pp. 502–509.

[65] Y. Zou, D. Luo, W. Chen, Swarm robotic odor source localization using ant colony algorithm, in: IEEE International Conference on Control and Automation, 2009. ICCA 2009, 2009, pp. 792–796.

[66] X. Feng, Y. Liang, L. Jiao, Bio-inspired optimisation approach for data association in target tracking, Int. J. Wirel. Mob. Comput. 6 (3) (2013) 299–304.

[67] M. Dorigo, L. Gambardella, Ant colonies for the traveling salesman problem, BioSystems 43 (1997) 73–81.

[68] K. Passino, Biomimicry of bacterial foraging for distributed optimization and control, IEEE Control Syst. Mag. 22 (3) (2002) 52–67. cited By (since 1996) 863.

[69] C. Sierakowski, L.S. Coelho, Path planning optimization for mobile robots based on bacteria colony approach, in: A. Abraham, B. de Baets, M. Köppen, B. Nickolay (Eds.), Applied Soft Computing Technologies: The Challenge of Complexity, in: Advances in Soft Computing, vol. 34, Springer, Berlin Heidelberg, 2006, pp. 187–198.

[70] M. Turduev, M. Kirtay, P. Sousa, V. Gazi, L. Marques, Chemical concentration map building through bacterial foraging optimization based search algorithm by mobile robots, in: 2010 IEEE International Conference on Systems Man and Cybernetics (SMC), 2010, pp. 3242–3249.

[71] K.N. Krishnanand, D. Ghose, Detection of multiple source locations using a glowworm metaphor with applications to collective robotics, in: Proceedings 2005 IEEE Swarm Intelligence Symposium, 2005. SIS 2005, 2005, pp. 84–91.

[72] K. Mcgill, S. Taylor, Robot algorithms for localization of multiple emission sources, ACM Comput. Surv. 43 (3) (2011) 15:1–15:25.

[73] K.N. Krishnanand, P. Amruth, M.H. Guruprasad, S. Bidargaddi, D. Ghose, Glowworm-inspired robot swarm for simultaneous taxis towards multiple radiation sources, in: Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006, 2006, pp. 958–963.

[74] K. Krishnanand, D. Ghose, A glowworm swarm optimization based multi-robot system for signal source localization, in: D. Liu, L. Wang, K. Tan (Eds.), Design and Control of Intelligent Robotic Systems, in: Studies in Computational Intelligence, vol. 177, Springer, Berlin Heidelberg, 2009, pp. 49–68.

[75] X.S. Yang, Firefly algorithm, in: Nature-Inspired Metaheuristic Algorithms, 2008, pp. 79–90.
[76] X. Yang, Firefly algorithm, stochastic test functions and design optimisation, Int. J. Bio-Inspired Comput. 2 (2) (2010) 78–84.
[77] X. Yang, X. He, Firefly algorithm: recent advances and applications, Int. J. Swarm Intell. 1 (1) (2013) 36–50.
[78] I. Fister, I. Fister Jr., X.-S. Yang, J. Brest, A comprehensive review of firefly algorithms, Swarm Evol. Comput. 13 (1) (2013) 34–46.
[79] C. Liu, Z. Gao, W. Zhao, A new path planning method based on firefly algorithm, in: Proceedings of the 2012 Fifth International Joint Conference on Computational Sciences and Optimization, CSO 2012, IEEE, Harbin, Heilongjiang, China, 2012, pp. 775–778.
[80] G. Wang, L. Guo, H. Duan, L. Liu, H. Wang, A modified firefly algorithm for ucav path planning, Int. J. Hybrid Inf. Technol. 5 (3) (2012) 123–144.
[81] A.A. Abshouri, M.R. Meybodi, A. Bakhtiary, New firefly algorithm based on multi swarm & learning automata in dynamic environments, in: IEEE proceedings of the Third International Conference on Signal Processing Systems, ICSPS 2011, Yantai, China, 2011, pp. 73–77.
[82] S.M. Farahani, B. Nasiri, M.R. Meybodi, A multiswarm based firefly algorithm in dynamic environments, in: Third Int. Conference on Signal Processing Systems, ICSPS 2011, Yantai, China, 2011, pp. 68–72.
[83] N. Chai-ead, P. Aungkulanon, P. Luangpaiboon, Bees and firefly algorithms for noisy non-linear optimisation problems, in: Proceedings of the International MultiConference of Engineers and Computer Scientists, IMECS 2011, vol. 2, Hong Kong, 2011, pp. 1449–1454.
[84] B. Nasiri, M.R. Meybodi, Speciation-based firefly algorithm for optimization in dynamic environments, Int. J. Artif. Intell. 8 (S12) (2012) 118–132.
[85] I. Moser, R. Chiong, Dynamic function optimization: The moving peaks benchmark, in: E. Alba, A. Nakib, P. Siarry (Eds.), Metaheuristics for Dynamic Optimization, in: Studies in Computational Intelligence, vol. 433, Springer, Berlin Heidelberg, 2013, pp. 35–59.
[86] W. Alt, Biased random walk models for chemotaxis and related diffusion approximations, J. Math. Biol. 9 (2) (1980) 147–177.
[87] A. Dhariwal, G. Sukhatme, A.A.G. Requicha, Bacterium-inspired robots for environmental monitoring, in: 2004 IEEE International Conference on Robotics and Automation, 2004. Proceedings. ICRA'04, vol. 2, 2004, pp. 1436–1443.
[88] Z. Wang, D. Gu, Cooperative target tracking control of multiple robots, IEEE Trans. Ind. Electron. 59 (8) (2012) 3232–3240.
[89] R.E. Kalman, A new approach to linear filtering and prediction problems, Trans. ASME J. Basic Eng. 82 (1960) 35–45. Series D.
[90] J. Ota, T. Arai, Y. Yoshimura, N. Miyata, E. Yoshida, D. Kurabayashi, J. Sasaki, Motion planning of multiple mobile robots by a combination of learned visibility graphs and virtual impedance, Adv. Robot. 10 (6) (1995) 605–620.
[91] X.-S. Yang, Metaheuristic optimization: Algorithm analysis and open problems, in: P.M. Pardalos, S. Rebennack (Eds.), Experimental Algorithms, in: Lecture Notes in Computer Science, vol. 6630, Springer, Berlin Heidelberg, 2011, pp. 21–32.
[92] V. Braitenberg, Vehicles: Experiments in Synthetic Psychology, MIT Press, Cambridge, 1984.
[93] K. McGill, S. Taylor, Comparing swarm algorithms for multi-source localization, in: 2009 IEEE International Workshop on Safety, Security Rescue Robotics, SSRR, 2009, pp. 1–7.
[94] K. McGill, S. Taylor, Comparing swarm algorithms for large scale multi-source localization, in: IEEE International Conference on Technologies for Practical Robot Applications, 2009. TePRA 2009, 2009, pp. 48–54.
[95] N.E. Humphries, N. Queiroz, J.R. Dyer, N.G. Pade, M.K. Musyl, K.M. Schaefer, D.W. Fuller, J.M. Brunnschweiler, T.K. Doyle, J.D. Houghton, G.C. Hays, C.S. Jones, L.R. Noble, V.J. Wearmouth, E.J. Southall, D.W. Sims, Environmental context explains lévy and brownian movement patterns of marine predators, Nature 465 (7301) (2010) 1066–1069.
[96] N.E. Humphries, H. Weimerskirch, N. Queiroz, E.J. Southall, D.W. Sims, Foraging success of biological Lévy flights recorded in situ, Proc. Natl. Acad. Sci. 109 (2012) 7169–7174.
[97] Y. Zhou, Y. Wang, S. He, J. Wu, A novel double glowworm swarm co-evolution optimization algorithm based lévy flights, Appl. Math. Inf. Sci. 8 (1L) (2014) 355–361.
[98] X.-S. Yang, Firefly algorithm, lévy flights and global optimization, in: M. Bramer, R. Ellis, M. Petridis (Eds.), Research and Development in Intelligent Systems XXVI, Springer, London, 2010, pp. 209–218.
[99] J. Xie, Y. Zhou, H. Chen, A novel bat algorithm based on differential operator and lévy flights trajectory, Comput. Intell. Neurosci. (2013).
[100] D.K. Sutantyo, S. Kernbach, V.A. Nepomnyashchikh, P. Levi, Multi-robot searching algorithm using lévy flight and artificial potential field, CoRR.
[101] D. Parrott, X. Li, Locating and tracking multiple dynamic optima by a particle swarm model using speciation, Trans. Evol. Comput. 10 (4) (2006) 440–458.
[102] G.J. Barlow, Improving memory for optimization and learning in dynamic environments (Ph.D. thesis), Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, 2011, July.

Madhubhashi Senanayake is an M.Phil. student at the Faculty of Information Technology in Monash University, Australia. Her thesis focuses on searching and tracking targets with swarms of robots. She obtained her honours degree in Electronic and Telecommunication Engineering from the University of Moratuwa, Sri Lanka in 2005. Prior to commencing postgraduate studies at the Monash Swarm Robotics Lab in 2013, she gained over seven years of experience as a software engineer in the embedded systems industry, mainly in Japan.

Ilankaikone Senthooran is currently a Ph.D. student at the Faculty of Information Technology in Monash University, Australia where he is focusing his research on autonomous MAVs. He obtained the B.Sc. Engineering honours degree from the University of Moratuwa, Sri Lanka in 2005. He received the M.Eng. degree in Computer Science from the Tokyo Institute of Technology, Japan, where he completed several conference publications and thesis focusing on automatic model verification and code generation. He also has industry experience in both engineering and managerial roles.

Jan Carlo Barca's major research interests are in the areas of swarm robotics, swarm intelligence and distributed sensing. He is currently the Director of Monash Swarm Robotics Laboratory and is involved in a wide range of research projects on swarms of UAVs and swarm intelligence. He has also been involved in several research projects on distributed control of ground moving vehicles. Jan Carlo has published work at several international conferences and in high level journals. He has also managed projects in both industry and academia.

Hoam Chung received the B.A. and M.S. degrees in precision mechanical engineering from Hanyang University, Seoul, Korea, in 1997 and 1999, respectively, and the Ph.D. degree in mechanical engineering from the University of California Berkeley, Berkeley, in May 2006. He has worked in the Intelligent Machines and Robotics Laboratory at the University of California Berkeley as a Postdoctoral Researcher and a Research Scientist from 2006 to 2009. He joined the Faculty of Engineering at Monash University, Clayton, Vic., Australia, in 2010, and is now a Lecturer in the Department of Mechanical and Aerospace Engineering. His research interests include receding horizon control theory and application, semi-infinite optimisation problems, real-time system design and implementation, and autonomous unmanned systems.

Joarder Kamruzzaman is currently an Associate Professor in the Faculty of Science and Technology, Federation University Australia. Prior to joining Federation University, he worked in Monash University, Australia as an Associate Professor and Bangladesh University of Engineering and Technology, Bangladesh. His research interests include wireless sensor networks, Internet of Things and computational intelligence. He has published over 190 articles which include over 50 journal publications. Two of his conference papers were awarded best papers in mainstream IEEE conferences. He is currently serving as an Editor of Elsevier Journal of Networks and Computer Applications.

Manzur Murshed received the BScEngg (Hons) degree in computer science and engineering from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 1994 and the PhD degree in computer science from the Australian National University (ANU), Canberra, Australia, in 1999. He is currently an Emeritus Professor Robert HT Smith Professor and Personal Chair at the Faculty of Science and Technology, Federation University Australia. He has had served as an Associate Editor of IEEE Transactions on Circuits and Systems for Video Technology in 2012 and as a Guest Editor of special issues of Journal of Multimedia in 2009–2012.