# A Brief Survey of Problems and Solutions in Robotic Swarms

S.J.A. Bekhoven S.P. Metman M.J. Rogalla

March 30, 2014

**Abstract**

Swarm robotics has become a prominent and promising research area in the recent years. It has great potential use for a large variety of applications. In order to not forget the big picture, we believe that a survey of a few basic problems in this field should be given. This survey presents a concise overview of a few problems that the robotic swarms research area has faced. For each problem we provide a small discussion as to which approaches were chosen. Finally we mention the remaining issues for each problem. Each of the algorithms can be categorized by their usage of range information and location information. We show the problems which come with each of these categories and compare the scalability and performance of each approach. This provides insight into the properties of the algorithms which affect the scalability and the performance. Addressing these problems and their approaches gives a better overview and offers inspiration to solve other problems. Finally we give a general conclusion regarding the recent advancements in robotic swarms.

## 1 Introduction

Swarm robotics has become a prominent and promising research area in the recent years. It has great potential use for a large variety of applications, some of which have already been successfully implemented. We want to provide a global overview of the main problems found in the research area of robotic swarms. Many articles already exist which give an overview of applications and used practices in this area, but often do not explain the problems underlying these practices. Therefore, we focus on providing a problem-oriented overview of the robotic swarms, while also providing a general overview of best practices and solutions of these problems.

We start by defining some terminology, since some of the terms used in robotic swarms are ambiguous and can be interpreted in different ways. We define a swarm as a scalable network of robots which consists out of more than two robots. Furthermore, we only consider robotic swarms in which every robot has some form of distributed intelligence. An exception is of course when a swarm of multiple robots is controlled by one control station. Because the swarm robots have to communicate either directly or indirectly with each other, the swarm will still have some form of distributed intelligence to function.
Robotic swarm algorithms can roughly be characterized by their location type and their information type. Algorithms can then respectively be either *location-based* or *location-free* and either *range-based* or *range-free*:

**Location-free** Robots have no knowledge of their absolute location but may keep track of their relative location.

**Location-based** Robots have perfect knowledge of their absolute location.

**Range-free** Robots do not communicate or communicate via some kind of central base.

**Range-based** Robots communicate within predetermined range.

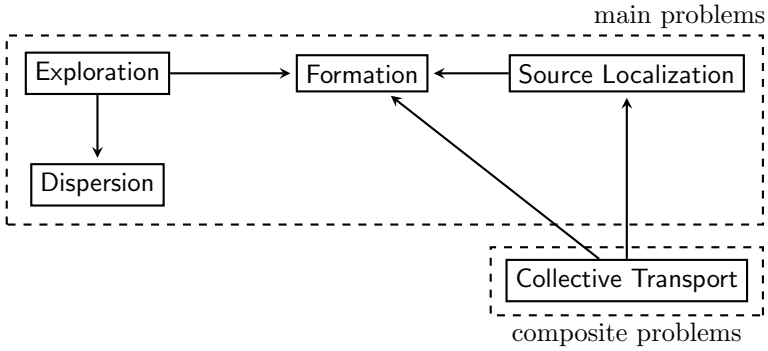


Figure 1: Problem Composition Overview

In the following sections, we compare the algorithms by two characteristics. The first characteristic is *scalability*, by which we mean the ability of maintaining performance when the population in the robot swarm is increased. The second is *performance*, by which we mean the general efficiency. We define efficiency separately for each problem before comparing the algorithms of that problem. We do this because the solutions have different ways of expressing efficiency for each problem.

A composite problem is a problem composed of multiple main problems in such a way that these main problems influence the working of the solutions. Although such a composite problem can be singled out, a lot of the main problems have some form of overlap too, although not with significant impact. The main problems Dispersion and Source Localization both include the Formation problem, and the Collective Transport is composed of the Formation problem among others. The Exploration problem is more of an extension to the Dispersion problem. This relationship is shown in Figure 1.

The remainder of this paper is then structured as follows. In Section 2 until Section 5 we define the main problems in robotic swarms. Then, in Section 6, we discuss the composite problem Collective Transport. For each problem we mention the possible real-life applications, their subproblems and the underlying algorithms of the solutions. After that we discuss the characteristics of each algorithm, the corresponding (dis)advantages and where possible some remaining problems. For more information on the operation of the mentioned algorithms, you can check the references provided. Finally we briefly discuss our observations in Section 7.

# 2 Formation

Formation is the problem of controlling the relative position and orientation of the robots in a group while allowing the group to move as a whole. [7] Formation is one of the key problems in robotics swarms, as it is a primitive in many other problems and composite-problems. In particular it is used in the collective transport problem, where a swarm has to hold a formation to move an object. An example of such an application is moving an object with robotic cars. [23] An application in which formation is also extensively used is area surveillance, where formation is used to increase coverage. [5] A more specific example is indoor surveillance, where ground robots are already being used. [11, 35]

Two main subproblems arise from the original formation problem: the *communication problem* and the *formation stability* problem.
In the *communication subproblem* the algorithm needs to come with a communication algorithm for usage within the robotic swarm. Each of these algorithms differ in these inter-robot communication strategies. Some algorithms only rely on local communication while other algorithms communicate via a centralized system. This affects if an algorithm is range-free or range-based. Some algorithms keep track of each

robot's location and are thus location-based, while others are location-free. These locations can be be communicated through different ways of communication.

The *formation stability problem* is another subproblem, which each of these algorithms have to deal with. Specifically, the algorithm should be able to dampen the effects of disturbance propagation. This means that when one robot in the formation encounters a disturbance and moves out of formation, how is this situation handled in the swarm. The effect of such a disturbance should be reduced. We call this the *source disturbance dampening* subproblem. For each algorithm, we review how these problems are solved.

## 2.1 Algorithms

Many different algorithms have been used to solve the formation problem. [6, 7] These algorithms can categorized as: leader-follower algorithm [7, 9], behavior-based algorithm [2, 18], and virtual structure algorithm [32, 12].
The algorithms are build upon in later algorithms to create more novel algorithms. A few of these algorithms are: virtual space configuration [43], fuzzy formation control [31], and team-work software control. [15] We compare these algorithms primarily by their performance and their scalability. We define *performance* in how quickly the formation can be formed again after the formation is lost (disturbance dampening). This can also be referred to as the stability of the formation. We discuss each of these algorithms separately in the next section and make a comparison at the end of the chapter.

### 2.1.1 Leader-follower algorithm

First, we discuss the leader-follower algorithm. In the leader-follower algorithm a robot of the swarm is designed as the leader. The leader moves along a predefined path while the other robots, the followers, are maintaining a desired distance and orientation to the leader. [7] This can be implemented by equipping the leader robot with a omni-directional camera and logical sensors, like an obstacle detector and a collision detector. The leader then instructs the followers through local communication. [9] The main problem with this algorithms is that it depends heavily on the leader and when something happens to the leader, the algorithm fails. Source disturbance dampening is done by the leader; the leader communicates any disturbances to its followers.

### 2.1.2 Behavior-based algorithm

The second algorithm we discuss is the behavior-based algorithm. In the behavior-based algorithm, every robot in the swarm is programmed with a certain behavior. These behaviors may differ between robots. Some examples of these behaviors are the collision-avoidance behavior and the target-seeking behavior. The action that is taken is decided by weighing the relative importance of each behavior. [7] These behaviors can then be used to maintain certain formations like a line, a column, a diamond and a wedge. [2] The dynamics and stability of this algorithm are calculated with the Lapyunov function, which is used to account for many stability issues. [18] In this algorithm, the decisions are not made locally, but real-time data is sent to a system which then decides what each robot should do based on their behaviors. Disturbance source dampening is applied by the different behaviors assigned to each swarm robot.

### 2.1.3 Virtual structure algorithm

The last of the more basic algorithms is the virtual structure algorithm. The virtual structure algorithm considers the formation as a single virtual rigid structure such that the behavior of the robotic system is similar to that of a physical object. Desired trajectories are not assigned to each single robot but to the entire formation as a whole. The behavior of the formation in this case is exactly predictable but also generates a large overhead. [7] Such a virtual structure is thus decentralized and appropriate

distributed control strategies can be made. [32] The control system can be structured in four sequential stages: defining the dynamics of the virtual structure, translating the motion of the virtual structure into the desired motion for each robot, deriving output feedback for each robot, and introducing formation feedback from each robot to the virtual structure. [12]

### 2.1.4 Virtual space configuration

Aside from these three algorithms which have become a standard way for implementing formation control, more novel algorithms have been developed. One of these algorithms is the virtual space configuration algorithm. [43] This algorithm uses a leader-follower algorithm, but each robot uses virtual space and virtual robots inside that space. These virtual robots are then used by each robot to avoid collisions. The main difference is then that multiple follower robots in formations can maintain formation without crashing (using only virtual collisions), and they can have scalability of formation. This algorithm is range-based and location-free, but produces a lot of overhead in local communication.

### 2.1.5 Fuzzy formation control

Another novel algorithm is fuzzy formation control. [31] In robotic swarms, optimal control techniques have been found to treat robust stabilization and tracking problems. But in these cases, the exact model of the system has to be known. But this fuzzy formation control technique provides a robust scheme with a fuzzy logic estimator to implement effective controls for uncertain dynamic models. This way, the exact model does not need to be known, as nonlinearity, external disturbances and model uncertainties are dealt with by fuzzy approximation.

### 2.1.6 Teamwork software control

Lastly, the last formation control we discuss is the teamwork software control. [15] This algorithm combines different control algorithms and tries to integrate this in one formation control algorithm. It is mostly behavior-based, and each of the robots in the swarm has its own behavioral process. But instead of carrying out its behavior directly, each robot communicates with each other to set a “team flag”. When every robot agrees with this flag, only then is this order carried out. This resembles the virtual structure control algorithm a little, as in that the swarm is considered as a whole. This algorithm is location-free but range-based, and produces more overhead than the behavior-based algorithm.
All the algorithms try to solve the formation problem. These algorithms are all location-free, which makes sense. Because when an algorithm is location-based, all robots know their exact locations and can adjust their position, and the problem is already solved.
The most important attribute for each algorithm is the performance, so how stable the formation is, as in how good the formation can be kept. This performance increases in the later algorithms, which expand upon the more basic algorithms. But, as the performance increases, the scalability lower for algorithms which communicate via a central framework. Take for example the teamwork software control algorithm. This algorithm provides high performance, but also creates lots of overhead activity due to its central communication. This decreases the scalability of this algorithm significantly.

## 2.2 Discussion

Concluding this chapter we discuss the future problems of these algorithms. As can be seen from the algorithms, centralized communication reduces scalability. Although centralized communication does provide high performance, the lower scalability is a downside. Decentralized algorithms are more likely to be scalable, and is an interesting topic for future solutions to the formation problem. Another problem faced by these algorithms is reliability, especially for the leader-follower algorithm. If the leader is lost (for example destroyed), the algorithm does not work anymore. A new leader could be chosen among the followers, but this is not taken into account in all the leader-follower algorithms. This is a problem that most of these algorithms do not take in account.

| Algorithm | Range | Location | Performance | Scalability |
|---|---|---|---|---|
| Leader-follower | Range-based | Location-Free | Medium | Low |
| Behavior-based | Range-free | Location-Free | Medium | High |
| Virtual structure | Range-based | Location-Free | High | Medium |
| Virtual space | Range-based | Location-Free | High | Low |
| Fuzzy control | Range-free | Location-Free | High | Medium |
| Teamwork control | Range-based | Location-Free | High | Low |

Table 1: Overview of Formation Algorithms

# 3 Dispersion

We consider dispersion as one of the key problems in robotic swarms. [41, 24, 20] Dispersion can be compared to the blanket coverage problem, which is defined in one of the first papers written on the topic of swarm robots: *Command Control for Many-Robot Systems.* [13] The objective of blanket coverage is defined as "achieving a static arrangement of elements that maximizes the detection rate of targets appearing within the coverage area." [13] The difference between dispersion and formation is often confused. While formation is trying to maintain explicitly specified spacing relationships, dispersion tries to find the best "formation" for the current environment.
Dispersion has a great number of applications. Some of these applications include, but are not limited to exploration, surveillance, military response, disaster response and planetary exploration. [20, 29, 24]

## 3.1 Algorithms

In this section we discuss a few categories of approaches which are used to solve the dispersion problem. First we discuss the simpler algorithms which are based on the approach which includes randomness such as random-walk and wall-following. After that we show which problems the dispersion problem has faced in the past and how the newer algorithms have solved these problems. Further we motivate as to why the algorithms are put into a specific category and discuss the scalability and performance for each type of algorithm. When discussing the *performance* we mean the time and ability to fully create a uniform dispersion in any given environment, whether it is static or dynamic. When discussing the *scalability*, we mean the ability of maintaining the same performance when the population in the robot swarm is increased.

### 3.1.1 Random-based Approach

The algorithms in this category solve the dispersion problem with an approach which relies on randomness to perform the dispersion. All of these algorithms are mainly location-free and primarily do not measure distances making them also range-free.
The most pre-eminent algorithm in this category is the *Random-Walk* algorithm due to its simplistic nature and efficiency. [26] The *Random-Walk* algorithm changes the robot's orientation randomly and moves it forward until an obstacle is detected. The algorithm then repeats the steps indefinitely. The algorithm is very scalable, due to the simple set of steps that have to be performed and due to its non-interdependence of other robots in the swarm. A minor drawback with the *Random-Walk* algorithm is that it does not guarantee uniform dispersion. Due to the reliance on randomness, the algorithm is not optimal. If low-energy consumption has a high priority in the system, we do not recommend the usage of this algorithm. In all other situations which allow the usage of the *Random-Walk* algorithm, the algorithm is greatly recommended due to its satisfying performance and highly rated scalability.
Another algorithm which is also used for dispersion purposes is the *Follow-Wall* algorithm. [26] Originally the algorithm is not random, but follows an exact set of rules depending on the environment. The algorithm is created for a non-dynamic environment and not for usage in robotic swarms. The robot is

unable to see the difference between robots and other obstacles. This creates the possibility for robots to constantly follow each other, while thinking that they are actually moving around a wall or other static obstacles. The performance rating is low due to the meager amount of randomness and its inability to guarantee the uniformity of the dispersion. The scalability is also rated low, due to the performance deterioration when increasing the scale.

### 3.1.2 Graph Theory Approach

The most-common dispersion algorithms which use a graph theory approach can be divided into two categories: tree-search algorithms and graph-connectivity algorithms.
The tree-search inspired algorithms have two strategies: the *Depth-First Leader-Follower(DFLF)* Strategy and the *Breadth-First Leader-Follower* Strategy. [14] These strategies rely on building a tree of all the possible moves that each robot can can make. This tree can be created if and only if it is possible to have a total overview of the environment in which the bots are located. Therefore the implementation of these algorithms are mainly considered as *location-based* algorithms. The *BFLF* algorithm requires the robots to travel less and has reduced complexity level compared to the *DFLF* algorithm. Still it maintains the same quality of the solution and thus has our preference. The performance of these algorithms is great in terms of quality of the solution. However, due to the high complexity of these types of algorithms, the scalability of this type of algorithms is rather low. Another downside is that the centralized control method brings is a much greater risk of failure in hostile environments.
One of the algorithms regarding connectivity in graphs is the clique-intensity algorithm. [41] The clique-intensity algorithm is range-free since it simply tries to detect other surrounding swarm-bots within the limited connection range and does not have the need to measure distances. The performance and scalability of the algorithm is very high, due to decentralized control. The clique-intensity algorithm faces problems due to the fact that there are high amounts of noise in the wireless intensity signals when used in real-world applications.

### 3.1.3 Artificial Potential Fields Approach

Artificial Potential Fields (APF) algorithms use fields to repel robots away from each other and attract them to their goal. [16] There are many differences between implementations of these algorithms, but all implementations use relative locations, both measuring and bearing. Therefore the algorithms which use this approach are location-free. [28] The algorithms are well-scalable due to their relatively low complexity. The performance of the algorithms is rated high, as the robots are uniformly dispersed. Some minor problems with this type of algorithm exist. Minor changes in sensing positions from other robots can cause ripple effects throughout the network. This can however be solved by incorporating *dead-zones*. These dead-zones allow very minor changes in location, without the immediate reaction of the network and thus the ripple effects are reduced. [28]

### 3.1.4 Inverse-Vector Approach

Some examples of the Inverse-Vector approach are the seek-open algorithm [26], the Fiducial algorithm [26] and the Uniform Directed Dispersion (UDD) algorithm [24]. Each of the algorithms sense where obstacles and other robots are relatively positioned and calculate a vector of that data. Afterwards they calculate the inverse of that vector and move into that direction. The *Fiducial* algorithm has an advantage over the *Seek-Open* algorithm in that it uses a beacon like system. This prevents robots from running into each other and encourages uniformity of the distribution. The seek-open and UDD algorithm use other distance measures mostly using ultrasonic sensors. In both cases the algorithm is a range-based algorithm, where every robot is able to get the relative location. It is important to stress that the algorithms should perform periodic checks to detect dynamic changes in the environment, such as other moving robots.

## 3.2 Discussion

There are many problems with the implementation of the dispersion algorithms. Many problems have been solved, but there are also some remaining problems left to be solved. In this section we discuss both topics. Some problems concerning small fluctuations in relative position data can have a major effect in range-based approaches such as the Artificial Potential-Field approach and the Inverse-Vector approach. This has been solved in *Swarm Dispersion via Potential Fields, Leader Election, and Counting Hops* with the introduction of *dead-zones*. These dead-zones are limited error ranges, which allow for small location changes, without affecting the whole robotic swarm. In location-based algorithms, there are problems with the building of the exact overview of the environment. Very often applications do not allow for this, due to for example dynamic changes in the environment or a lack of observability. Research in this area should be focused on high-accuracy real-time observation tools in combination with range-based algorithms which prevent obstacle-collision.

| **Algorithm** | **Approach** | **Range** | **Location** | **Performance** | **Scalability** |
|---|---|---|---|---|---|
| Random Walk | Random-Based | Range-Free | Location-Free | Medium | High |
| Follow Wall | Random-Based | Range-Free | Location-Free | Low | Low |
| DFLF | Graph-Theory | Range-Based | Location-Based | Medium-High | Low |
| BFLF | Graph-Theory | Range-Based | Location-Based | High | Low |
| Clique-Intensity | Graph-Theory | Range-Free | Location-Free | Medium | High |
| APF | Potential-Fields | Range-Based | Location-Free | High | High |
| Directed Dispersion | Inverse-Vector | Range-Based | Location-Free | Medium | High |
| Seek Open | Inverse-Vector | Range-Based | Location-Free | Low | Medium |
| Fiducial | Inverse-Vector | Range-Based | Location-Free | Medium | High |

Table 2: Overview of Common Dispersion Algorithms

# 4 Exploration

Another fundamental problem in swarm robotics is the task to fully explore an unknown environment. The main goal is to minimize the overall exploration time while still exploring the whole environment. The main problem faced when trying to achieve this goal is finding appropriate target points for each individual robot so that they simultaneously explore different regions of the environment. [4] Exploration is found in many robotic swarms problems, for example in *path-finding*, *collective transport* and *surveillance*. Practical applications that use exploration are for example rescue missions. [27, 29] In this paper by Naghsh [27], a robotic swarm applying exploration is used to assist navigation for firefighters. It is used in situations in which their vision is blocked by smoke and obstacles. A last example of an application is cleaning. [42] Here, exploration is used to clean a surface with cleaning robots as fast and as efficient as possible. Exploration is used in many more different robotic swarm applications and is a building block for many other problems.

## 4.1 Algorithms

The exploration problem has been studied in detail for single robots [19, 1] as well as for robotic swarms. The first real approach towards robotic swarm exploration is based on finding frontiers, cells that are reachable and adjacent to unexplored cells. [44] This approach is extended in multiple ways of which we will mention two. [39, 37] After that, we will discuss some other solutions to the exploration problem. [38, 47] When comparing we define the *performance* of the exploration algorithms as the explored area per distance traveled.

### 4.1.1 Frontier-based Algorithms

A first approach is frontier-based exploration, which is inspired by the question "Given what you know about the world, where should you move to gain as much new information as possible?". Its goal is to gain as much information as possible when traveling towards a new location. An evidence grid is used in which the occupancy probability is stored for each cell, so the algorithm is *location-based*. The cells have, dependent to their occupancy probability, a certain state which is either open, unknown or occupied. Each open cell that is adjacent to an unknown cell is labeled as a frontier cell. Every group of frontier cells above a certain size is considered a frontier. Once the frontiers have been detected, the robot navigates towards the nearest unvisited frontier. If the robot is unable to make progress to its destination, it will add the frontier to the list of inaccessible frontiers. Each time a robot does or does not reach a frontier it performs a sensor sweep and adds the new information to a local grid, which is communicated and then merged with every robots global map. [44]

A limitation of the general frontier-based approach is that since navigation is independent, robots may waste time navigating to the same frontier. This will either cause an avoidance maneuver or the robots will block each other. In the last case the robots will mark their destination frontiers as inaccessible which of course is not preferable. Furthermore global communication is assumed, which in real-world applications is nearly impossible since robots have limited communication range. Several attempts have been made to solve these problems by extending the general frontier-based algorithm. An example is the implementation of a bidding algorithm. The robots first select a certain target according to the frontier-based algorithm. It then broadcasts a bid according to: the target of the robots in the sensor range, the distance to the targeted cell and a nearness measure. This nearness measure is a factor which is dependent of the number of robots in the neighborhood to keep the robots together and thus sustain communication. After waiting for constant time, if the robot bids the highest value, it travels towards its target. If this approach would have assumed global communication it would have made sure two robots never travel towards the same frontier. Since limited communication range is assumed a nearness measure is implemented to make the robots tend to stay together to sustain communication. This causes partitioning and therefore the robots sometimes still choose the same frontier to travel to. All together this approach results in less repeated coverage and therefore less exploration time. [37]

Another limitation of the general frontier-based approach is that it is possible that the robots concentrate in certain parts of the area and therefore reach other parts much later. In case of for example search and rescue missions, it is important that the area is explored more equally distributed. An approach that solves this problem is the algorithm based on K-means clustering. First the algorithm divides the unknown space in as many regions as robots by the K-means clustering algorithm with $K$ the number of robots. The algorithm then assigns all robots to a certain region by distance calculation, making a difference between accessible and inaccessible regions. After that each robot gets assigned to a frontier cell in its region. Robots with accessible regions will be assigned to frontier cells by the distance between the robot and the frontier and receive a penalty for choosing one close to another already chosen by another robot. Robots with inaccessible regions receive another penalty for the distance from the frontier cell and the centroid of their own region. When this formula gets maximized, all robots will tend to choose frontiers in their own region or close to their own region. If a robot reaches its destination, the target assignment will be repeated. This approach does not result in lower exploration time, but does achieve good results in equally distributed exploring of the area. [39]

### 4.1.2 Market Economy Algorithm

A completely different approach is based on a market economy. Robots generate a list of goal points via a simple strategy, for example randomly or greedily. The robot then sets up an auction to sell its goal points and tries to buy better ones from other robots. Each goal point is awarded a certain revenue according to the amount of information it will provide and a certain cost according to the resources it will use to achieve it. When they have tried to sell all of their tasks and have bought the interesting

ones, the robot orders its goal point list greedily on distance and sets course for the first one in the list. At regular intervals the robots exchange pieces of their map to each other for a certain cost/revenue depending on the expected utility. The relatively simple algorithms used to generate goal points should be optimized by maximizing benefit (information gained) while minimizing the costs (in terms of travel distance). By allowing the robots to communicate via the market place architecture, the performance increases with a factor of 3.4 compared to a random walk in a four robot system. [47]

### 4.1.3 Heterogeneous Exploration Algorithm

In this approach robots with different size, speed and sensor range are used. The unknown area is treated as an occupancy grid with a status for each cell: occupied or free. All robots start by filling a space quantum varying in size according to the specification of the robots. If they explore a cell in the occupancy grid, the status is set to occupied and is from then on seen as an obstacle by all robots. When the robot sees a cell it cannot reach, it has found what we call a *tunnel*. The robot than places information about the tunnel at the robot call queue of another smaller robot. If that robot is not able to reach the unexplored cell as well, it will pass it to an even smaller robot and so on. When a robot finishes exploring its space quantum it either travels towards the tunnel placed on its queue or starts exploring the next adjacent space quantum. The robots continuously share their occupancy grid and the motion they are intending to avoid collisions. This approach assumes global and intensive communication amongst all robots which is very hard in real-life applications. [38]

## 4.2 Discussion

In the normal frontier-based algorithm we see an approach in which global communication is assumed, which means low scalability. Furthermore no coordination is implemented, which influences the performance negatively. This coordination is available in the next algorithm in the form of the bidding principle and therefore has higher performance. Because this approach does not assume global communication it is also highly scalable. The K-means clustering principle does not perform better than the bidding principle, but does explore the area equally distributed, which can be preferable in search and rescue missions. This approach again is range-free and based on a central base, which causes low scalability.

In the market economy based algorithm the exploration problem is addressed by optimizing the utility of and the distance traveled towards each cell which makes it very effective. However, this implementation is still centralized and therefore less scalable. In the frontier based algorithm with bidding principle not only the utility and the distance to be traveled towards it are taken into account, but also a certain nearness measure, which makes the robots tend to stay together, so communication stays possible. Therefore this algorithm is not only effective, but also highly scalable.

In general we see that most of the robotic swarm exploration algorithms keep track of some kind of map and are therefore location-based. This seems sensible, because it is the only way to know if a robot has been to a certain location already and thus not has to explore it again. On the other hand, to use the advantages of swarm robotics, this map has to be shared with other robots. Nearly all robotic swarm exploration algorithms assume global communication and are therefore range-free. However, in real-life applications robots have limited communication range, so either a central base (which dramatically decreases scalability) or some kind ad-hoc network (which would decrease the exploration efficiency) needs to be created. This brings us to a contradiction in which on the one hand we want the robots to share information for efficiency and thus stay together, and on the other hand want the robots to spread widely to gain as much new information as possible. The investigated algorithms have tried to optimize this contradiction. We think that the frontier-based algorithm with bidding coordination is a good way to go and should be further optimized in the future to come nearer to an optimal solution.

| Algorithm | Range-type | Location-type | Performance | Scalability |
|---|---|---|---|---|
| Frontier-based | Range-free | Location-based | Medium | Low |
| Frontier-based with bidding | Range-based | Location-based | High | High |
| Frontier-based with K-means clustering | Range-free | Location-based | High | Low |
| Market Economy | Range-based | Location-based | Medium-high | Medium |
| Heterogeneous Exploration | Range-free | Location-based | Medium | Low |

Table 3: Overview of Common Exploration Algorithms

# 5 Source Localization

Source localization with robotic swarms is a problem which has been receiving a lot of research attention in the past few years. The main solution is to design an algorithm that effectively allows a swarm of robots to explore an unknown area and find (multiple) source(s). Source localization can be used for many real-world applications. One of these applications is chemical plume tracing, in which localization is used to detect clouds of high density chemicals. [45] Another example similar to chemical plume tracing is radiation source search. [3] The difference is that in this case the source of leaking radiation is searched for, and not only for high density clouds. A third example is searching for fire, used to assist fire-fighters in their every-day work. [21] Evidently, source localization can be used to look for all kinds of emission sources, if the robots have the right sensors installed. [8] Although many practical applications can be found, a large amount of the work done in this field is purely theoretical. This is due to the fact that the price of these individual robots is still rather high and thus it is expensive to produce a swarm.

## 5.1 Algorithms

The algorithms for localizing (multiple) source(s) can be roughly divided in two categories: hill climbing algorithms and biologically inspired algorithms. In this chapter we discuss some examples of algorithms belonging to each category and compare them to each other. When comparing the *performance* of the algorithms we mean the distance to the source(s) after a certain amount of time.

### 5.1.1 Hill Climbing Algorithms

The first solution for the source localization problem is implemented with a gradient ascent algorithm. This algorithm uses a mobile robot with an electric nose which follows a gas gradient to the source. [33] By using multiple robots executing this kind of algorithm the source can be found even faster. Because by communicating the concentration they currently sense and combine that with the range and direction they receive it from, a wider and more precise gradient can be derived. [36]

In another more advanced gradient-based algorithm robots use a signal strength indicator to predict the location of the source with a certain probability. Then, they add this to their map and report it back to a central base. This base collects all readings and maps to create a global map and an uncertainty area which is sent back to all robots. With this data the robots continuously predict the source position with increasing accuracy as they move towards the source. [46]

In general, gradient-based algorithms robots have perfect knowledge of their location and are therefore *location-based*. Furthermore they are *range-free* since they use a central base to communicate. This affects the scalability of the algorithms in a negative way, since it is dependent of the capacity of the central base. Gradient-based algorithms are limited to single-source localization and can converge at local maxima instead of the real source. Some attempts have been made to prevent the robots from converging at a local maximum. This is done by implementing for example a random walk [10] or a swarm approach [8]. We discuss these algorithms in the next section (Section 5.1.2). Gradient-based

algorithms seem to perform better than random walk algorithms as is shown by simulation in the paper by Zhang et al. [46]

### 5.1.2 Biologically Inspired Algorithms

In nature swarms of organisms also have the need to look for targets (for example for hunting). So, inspired by nature, several algorithms have emerged. Some are extensions of the hill climbing algorithms, but in this chapter we mainly focus on swarming algorithms and random walk algorithms.

A swarming algorithm for localizing multiple sources is Particle Swarm Optimization (PSO), which is originally inspired by flocks of birds. In PSO a number of particles is randomly distributed over an unknown space of a problem or function. Each particle - in our case a robot - evaluates its current location according to a certain fitness function. Then it calculates the best position to go to according to its own historical best position and the historical best positions of the particle(s) in its neighborhood within a certain range. To prevent the particles from converging at a local maximum or just one of the sources a certain randomness is implemented. By continuously looking for a better position by sharing information, the swarm of particles eventually positions itself at the position of the source(s). [30] PSO based algorithms in general are more complex than for example gradient-based algorithms and therefore require more processing power. This however results in a more robust algorithm which perform better in unstable environments than for example gradient-based algorithms. [22].

In Glowworm Swarm Optimization (GSO) is inspired by nature through glowworms. The algorithm starts by distributing the glowworms randomly over the area. The glowworms, according to the fitness function, carry a certain luminescence quantity called luciferin. The closer they get to the source the more luciferin they contain and the more they attract other glowworms. At every movement step each glowworm moves towards a neighbor within a certain range that carries more luciferin. Thus, eventually they converge at the source(s). To prevent the robots from converging at local optima or just one source, the communication range of each robot varies at each step with a certain randomness. In comparison to PSO based algorithms GSO algorithms are completely *memoryless*. The number of sources found by GSO is proven to be a strong function of the sensor range. The sensor range, and with that the robots neighborhood, is therefore made a variable parameter. In comparison, in the PSO based algorithms the particle neighborhood exists of a constant amount of robots. In gradient-based algorithms blocking regions cause problems, while in GSO based algorithms the robots are able to select a feasible direction around the blocking region by communicating with each other. [17]

Another algorithm is the Biasing Expansion Swarm Approach (BESA). In this algorithm robots have communication possibilities over a limited range and together create an ad-hoc wireless network for global communication capability. The algorithm is therefore *range-based*. Each robot maintains an occupancy grid map to represent the environment which is initiated with all cells unexplored and is therefore *location-based*. After deployment robots share their locations, and sensed concentrations, with the swarm. The robots then communicate and navigate with three relatively simple rules: separation, cohesion and alignment. Robots can only move to the expansion cells. These are cells that are unexplored, unoccupied and next to another robot in the swarm. To make sure the swarm moves to the emission source, each expansion cell is given a certain biasing parameter. This biasing parameter is based on the number of robots, the distance between the expansion cell and the robot and the concentration sensed at the particular cell. The robot chooses the expansion cell with the highest parameter, so that the swarm eventually moves in the direction of the higher concentrations. The BESA algorithm in general performs twice times better than the general gradient-based approaches and is proven to be more robust for unstable environments. [8]

A rather simple algorithm inspired by nature is the Biased Random Walk (BRW) approach inspired by bacteria. This algorithm does not use communication or any form of localization and is therefore *location-free* and *range-free*. Robots only have the possibility to perform two actions: move or tumble. When

tumbling the robot just turns into a new random direction. When moving the robot travels a certain distance into the chosen direction. If the robot senses some form of emission from the source, the tumble frequency is lowered, so the distance moved per step increases. Eventually the robots conglomerate at multiple sources. One of the issues with multiple source localization is not converging at local maxima. Because of the high amount of randomness in the BSR algorithm, it seems to almost always find all sources. Sources with low emission will of course end up with a smaller fraction of the robots, but there are enough applications in which this would not matter. [10].

## 5.2 Discussion

| **Algorithm** | **Range-type** | **Location-type** | **Performance** | **Scalability** |
|---|---|---|---|---|
| Hill climbing | Range-free | Location-based | Medium | Low |
| PSO | Range-based | Location-free | High | High |
| GSO | Range-based | Location-free | High | High |
| BESA | Range-based | Location-based | Medium | High |
| BRW | Range-free | Location-free | Low | High |

Table 4: Overview of Common Source Localization Algorithms

The biggest subproblem of the source localization problem with robotic swarms is trying to find multiple sources. A lot of research is done in single source localization, but the extension to multiple sources seems to be rather difficult. For example simple hill climbing algorithms are limited to single source localization, because they simply follow the first gradient they find in an ascending way. This makes its performance quite weak. In the other algorithms, attempts have been made to overcome this problem by introducing some kind of randomness. In PSO it is added to the velocity of the robots, in GSO a varying communication range is implemented and in the BSR a random direction is chosen. In BESA however, an ad-hoc network is set up which gives robots the possibility to share their findings. In this way they help each other to find all sources and by the rule of separation do not enter each others cells so they do not converge at local maxima.

In general we see a trade-off has to be made. If we choose for a location-based algorithm which keeps track of explored sources/parts in a map, the robotic swarm needs global communication. This can be managed by for example using a central base as in hill climbing algorithms or setting up an ad-hoc network as in BESA algorithms. Choosing for a central base limits the scalability and when creating an ad-hoc network the swarm it is impossible to form partitions. On the other hand, if we choose for a location-free approach, we do not have an overview and thus are not sure if we have covered all sources. This can be managed by implementing some form of randomness as described above. Furthermore these approaches often require a randomly distributed start, which is rather difficult in real-life approaches. Either ways (location-free or location-based) result in medium to high performance.

The BSR algorithm does not participate in this trade-off, given the fact that is is neither location-based or range-based and does not use any form of communication. This obviously results in lower complexity, but also in lower performance.

In the end all algorithms face the same problem of converging and then staying at sources. In many real-life applications however it is not necessary to stay at a source once its found; it only has to be found. The answer to the question what to do when the task is finished, stays slightly unanswered.

Finally it is important to acknowledge that it is difficult to fairly compare source localization algorithms. Not only are there a lot of factors that influence the performance of the algorithms, there is also a general lack of base cases and references that form a proper ground truth for comparison.

# 6 Collective Transport

Collective transport of objects is the problem in which a swarm of robots locates a payload and collectively moves the payload to another place, like a home base. This problem is a composite problem, consisting of the already mentioned problems formation and source localization. Therefore, it is closer to the application field than the other problems. Still, we want to discuss this problem because it is an important problem in the research field of robotic swarms and it is interesting to see how different subproblems can compose a bigger problem. Transporting objects by robotic swarms has many potential applications in many settings, from agriculture to construction to disaster relief. Especially in dangerous settings like war zones or radio-active areas, robotic swarms can be a powerful tool to safely retrieve many objects. For example, recently Amazon, a large online retailer, announced it would make use of unmanned flying robots to deliver parcels at everyone's door. [1]

## 6.1 Algorithms

Collective transport can be separated into two subproblems: providing safe and reliable transport and moving a payload to its destination. These respectively correspond to the formation problem and the source localization problem.
When considering the formation problem, the algorithm is not considered with finding a payload. Then, we can conclude that the swarm in this algorithm is not completely autonomous and that the whole swarm is controlled by a single controller. This implies that algorithms used for the formation problem are location-based, because the location is known by the controller. [25, 23] When considering the source localization problem, the algorithm is not concerned with transporting the payload. Then, we can conclude the swarm in this algorithm works autonomously. This implies that the algorithms used for the source localization problem are location-free, because the location is now known by the controller. [34, 40]

### 6.1.1 Formation transport

There are two different properties important for formation transport: performance and scalability. We define *performance* as in how well the robots can stay in formation in dynamic environments.

The first algorithm we discuss is the *Aerial Equilibrium* algorithm. [25] This algorithm for aerial transport is implemented by attaching cables from quadrocopters to the payload. Then, each robot calculates its own movement in the swarm by a mathematical model. Because the swarm robots do not use local communication but get instructions from a user, this algorithm can be defined as a range-free algorithm. A change in the position of a robot is quickly communicated through the mathematical model, resulting in a good performance. Nevertheless, the scalability is not that good, because adding multiple robots will quickly result in robots getting in each others way. This is because the chance of touching each others rope increases with every robot added.

The second algorithm we discuss is called *Cluster Space Control* algorithm. [23] It uses a multi-robot formation control framework called cluster space control. It is utilized to control a swarm of four four-wheeled robots. A user using this type of transportation uses a joystick to input user controls to the whole swarm, after which the centralized control framework specifies the formation and the position and shape of the swarm. Because of the centralized communication there is no local ranged communication, so this algorithm is range-free. This control framework allows the user to effectively transport large objects with a scalable set of robots. The performance for this algorithm is very high, because the framework responds well to changes in the environment. The scalability is low, because the framework has to hold connections to every robot.

[1] Amazon's PrimeAir service, to be released sometime in 2015. www.amazon.com/b?node=8037720011

These two algorithms solve the same problem, namely the problem of transporting large and/or heavy objects, in two entirely different ways. They are both range-free and location-based, but differ mostly in scalability. As both algorithms have centralized communication, scalability is low for both. We summarize the properties in Table 5.

| **Algorithm** | **Range** | **Location** | **Performance** | **Scalability** |
|---|---|---|---|---|
| Aerial Equilibrium | Range-free | Location-based | Medium | Low |
| Cluster Space Control | Range-free | Location-based | High | Low |

Table 5: User-controlled swarm transportation algorithms

In the first algorithm, each swarm robot individually calculates its position to the other robots and the payload and the resulting tension on the payload. This way, the rotation and position can be controlled by the position and tension of each robot. With the second algorithm, that uses the cluster *cluster space control*, the calculations are not done individually. Instead, the framework calculates the position of each robot and the orientation of the payload. Therefore these methods both provide stability to the transported payload but in two completely different ways.

The problems these algorithms have are that the robots should be coordinated to rely on the location of other robots. With the mathematical model these are calculated through the tension of the cable attached to each robot, but in the framework it is calculated centrally. Problems that are not treated in these articles is how these swarms can locate a payload and autonomously can deliver it to a location. Another problem for which no solution is provided is that when these swarms grow to account for heavier objects, the robots will get in each other's way. Especially considering transportation, a technique in which every robot must latch on to a payload, this can be hard to practically scale.

### 6.1.2 Source localization transport

The source localization problem consists of finding the payload and transporting it to its destination. This subproblem can again be divided in four smaller problems: finding the payload, grabbing the payload, finding target location, moving the payload to the location. By defining these subproblems, the problem is easier to solve and easier to explain each algorithm. The *performance* is defined in how fast the payload is moved to the destination.

The first algorithm we discuss is the flocking algorithm. [34] Simply put, it locates the payload through a flocking algorithm and let the swarm robots push the payload to the goal location. First every robot looks if it can see the payload location. If so, move towards the location. If not, the robot compares its own heading with the other robots nearby and corrects it until the robot finds the payload location. In this paper, the payload location and the goal location are found with light sensors, communicates headings with infrared sensors and uses bump sensors to avoid obstacles. The robot latches on to the payload with a velcro strip. After finding the payload, the robot pushes from an arbitrary side, calculating the amount of force needed in relation to the middle point of mass with its bump sensors and transporting it to the goal location with its light sensor. And because the robots latch onto an arbitrary side, scaling of the swarm is extremely effective.

The following algorithm uses very simple robots, which is desirable. [40] It is based on granular convection, also known as the Brazil Nut Effect. The algorithm can be implemented in three different ways: (1) A swarm with homogeneous robots with no explicit communication, (2) a swarm of heterogeneous robots composed of robots that change their direction with two different probabilities, and (3) a swarm of heterogeneous robots that uses local communication to adjust the fraction of robots that change their direction with higher probability.

Each implementation has a higher performance than the last one, but also increase in communication. The way this algorithm works is as follows. The goal of the swarm robots is to transport the object to the goal location. Each swarm robot vibrates with a random force. The goal location outputs a repulsive force. As the robots are randomly placed around the payload and the goal location, the payload gets kicked around by the vibrating swarm robots. This continues until the payload is kicked to the target location. When the swarm increases, the amount of time it costs to transport the payload significantly decreases. With heterogeneous robots, there is more communication between robots to organize the vibrations of the robots.

The algorithms we describe are all location-free, as mentioned earlier. Flocking is a range-based algorithm, because the robots communicate the headings locally. Of course, granular convection with local communication is also range-based by definition. Although granular convection is mostly range-free, its scalability overall is high, because there is no central communication. With the granular convection method, adding local communication and using heterogeneous robots increases the performance. This is also not detrimental to the scalability of the approach.

| **Algorithm** | **Range** | **Location** | **Performance** | **Scalability** |
|---|---|---|---|---|
| Flocking | Range-based | Location-Free | High | High |
| Homogeneous granular convection | Range-free | Location-Free | Low | High |
| Heterogeneous granular convection | Range-free | Location-Free | Medium | High |
| Heterogeneous granular convection with local communication | Range-based | Location-Free | High | High |

Table 6: Overview of Collective Transport Algorithms

In the mentioned algorithms, moving the payload is mostly done by pushing. In real-life applications this would not always be a good option and a more sophisticated method of attaching should be used. When considering search-and-rescue operations for example, a solution to effectively grab the person to be rescued should be thought of. This is a problem that is yet to be identified in these algorithms and poses a good question for the future.

# 7 Discussion

In the previous sections, we reviewed the main problems found in the field of robotic swarms. However, these problems often overlap. This is because most of the problems found in robotic swarms often consist of multiple different problems. We focused on each main problem, highlighting the communication methods of every solution and properties of these communication methods. We summarize these properties in a Venn-diagram in figure 2, allowing for a compact overview of these solutions. The algorithms we discussed in previous sections provide useful insight, from which we can derive several conclusions. Together with the Venn-diagram, we explain how we came to these conclusions.

*Location-based* approaches keep track of some kind of map in such a way that the exact location of each swarm robot is accurately known. Because of this, location-based approaches are often able to act efficiently and remove redundancy. However, a couple of drawbacks can be defined.

If an algorithm is *location-based* and *range-based*, the robots can create some kind of ad-hoc network to have global communication possibilities. This global communication is very useful for exact orders and robot movement. But, such an ad-hoc network has a limited range. So, robots in the swarm can not split up from the rest of the groups to work outside this range. This takes away one of the great possibilities of robotic swarms and could decrease performance.

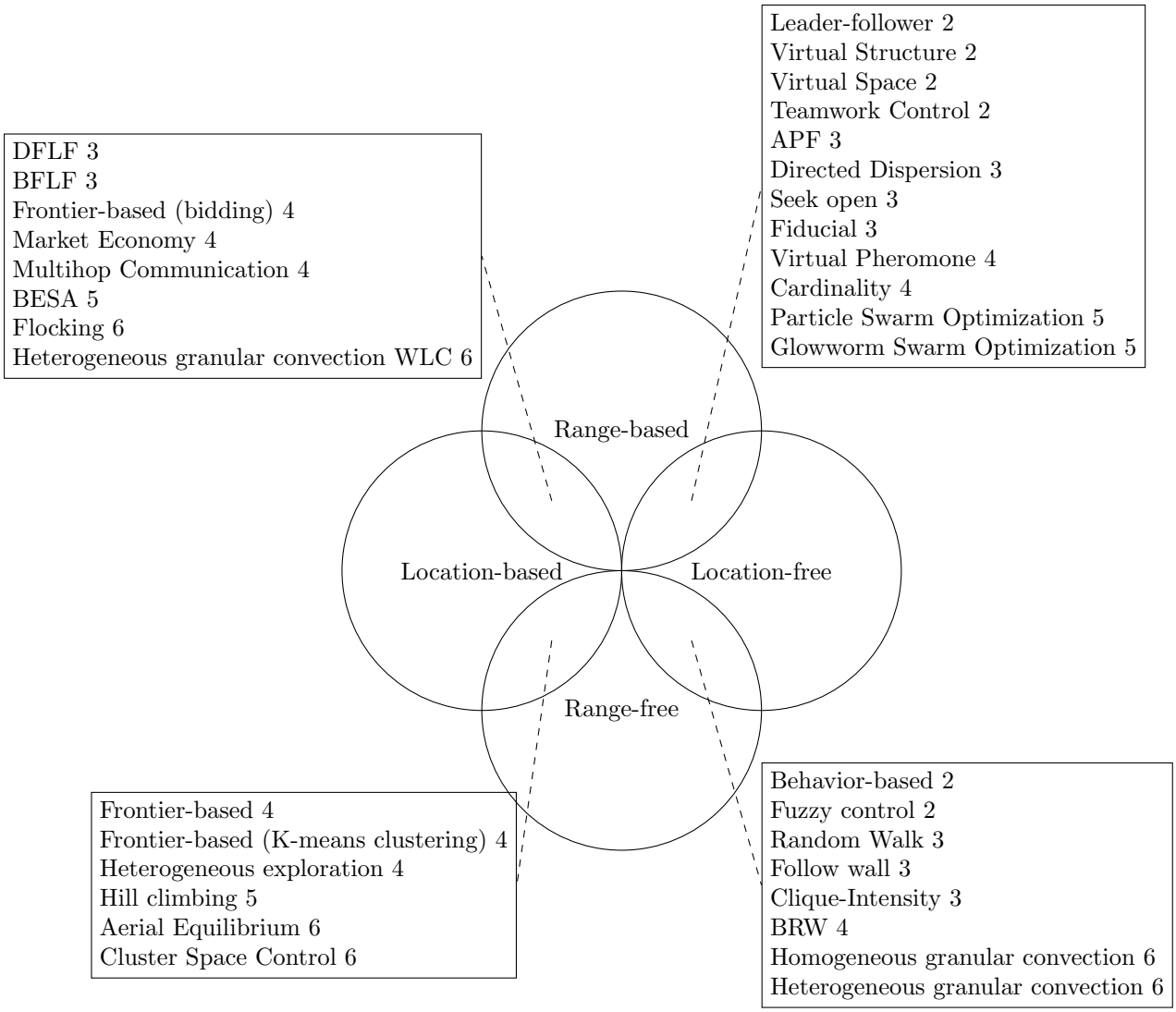


Figure 2: Overview of Algorithms

A robotic swarm can also accept that they do not have global communication. Then, they can only share their map knowledge or location history with their neighbors, which will decrease performance.

In *location-based* and *range-free* algorithms, the robots can communicate via some central base, but not locally. This can increase performance dramatically, because redundancy in communication can be removed completely. This is caused by the fact that every robot has access to all location information of all robots. However, this will result in very low scalability, since the reliability of the central communication is entirely dependent on the capacity of the central base.

*Location-free* approaches do not keep track of a map and are only able to determine some kind of relative position to each. Alternatively they do not keep track of locations at all. This means they are able to optimally adapt to dynamic environments and can be implemented in low-level robots with cheap sensors.

When an algorithm is both *location-free* and *range-free*, there generally exists no form of coordination. These algorithms are called collective algorithms and are often based on stochastic parameters. These

robots all have distributed algorithms, but are often heterogeneous to carry out different tasks. These algorithms are often highly scalable, because of the low communication overhead and cheap robots. Because they are highly scalable, many robots can exist in a swarm, and will eventually execute a task faster than a single robot.

The majority of the algorithms that we discuss are *location-free* and *range-based*. The main reason for this is that the algorithms mentioned in this category are very often highly scalable, have great performance and can be implemented with very simple robots. The difference with *range-free* algorithms however is that in *range-based* algorithms there exists some form of local communication. This can increase the efficiency of such an algorithm dramatically, because the versatility of the swarm increases. The robots can for example choose to stay together and share the information they get from their sensors, but can also choose to spread out in different groups to divide tasks depending on sensor input. Because these algorithms are so versatile, they are well-suited for many different real-life applications.

We would like to emphasize that there are some problems in robotic swarms, which we did not discuss in this survey. One such a problems is the surveillance problem, which efficiently tries to patrol a certain area. Another problem is the mapping problem, which tries to achieve collaborative mapping using robotic swarms. The reason we did not include these problems is because these are mostly all composite problems, of which we already have described their subproblems. Therefore we believe that these section would create a lot of redundancy and their contribution would be limited. Composite problems are closer to the application field of robotic swarms. For future research, we suggest to put more focus on the application aspect of robotic swarms.

## Acknowledgment

The authors would like to thank A. Loukas and A.S. Pruteanu for their valuable comments and suggestions to improve the quality of this survey.

# References

[1] Susanne Albers, Klaus Kursawe, and Sven Schuierer. Exploring unknown environments with obstacles. In *Proceedings of the tenth annual ACM-SIAM symposium on Discrete algorithms*, pages 842–843. Society for Industrial and Applied Mathematics, 1999.

[2] Tucker Balch and Ronald C Arkin. Behavior-based formation control for multirobot teams. *Robotics and Automation, IEEE Transactions on*, 14(6):926–939, 1998.

[3] Shishir Bashyal and Ganesh Kumar Venayagamoorthy. Human swarm interaction for radiation source search and localization. In *Swarm Intelligence Symposium, 2008. SIS 2008. IEEE*, pages 1–8. IEEE, 2008.

[4] Wolfram Burgard, Mark Moors, Cyrill Stachniss, and Frank E Schneider. Coordinated multi-robot exploration. *Robotics, IEEE Transactions on*, 21(3):376–386, 2005.

[5] Axel Bürkle, Florian Segor, and Matthias Kollmann. Towards autonomous micro uav swarms. *Journal of intelligent & robotic systems*, 61(1-4):339–353, 2011.

[6] Yang Quan Chen and Zhongmin Wang. Formation control: a review and a new consideration. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 3181–3186. IEEE, 2005.

[7] Luca Consolini, Fabio Morbidi, Domenico Prattichizzo, and Mario Tosques. Leader–follower formation control of nonholonomic mobile robots with input constraints. *Automatica*, 44(5):1343–1349, 2008.

[8] Xiaohui Cui, C Tim Hardin, Rammohan K Ragade, and Adel Said Elmaghraby. A swarm approach for emission sources localization. In *Tools with Artificial Intelligence, 2004. ICTAI 2004. 16th IEEE International Conference on*, pages 424–430. IEEE, 2004.

[9] Aveek K Das, Rafael Fierro, Vijay Kumar, James P Ostrowski, John Spletzer, and Camillo J Taylor. A vision-based formation control framework. *Robotics and Automation, IEEE Transactions on*, 18(5):813–825, 2002.

[10] Amit Dhariwal, Gaurav Sukhatme, and Aristides AG Requicha. Bacterium-inspired robots for environmental monitoring. In *Robotics and Automation, 2004. Proceedings. ICRA'04. 2004 IEEE International Conference on*, volume 2, pages 1436–1443. IEEE, 2004.

[11] Donato Di Paola, Annalisa Milella, Grazia Cicirelli, and Arcangelo Distante. An autonomous mobile robotic system for surveillance of indoor environments. *International Journal of Advanced Robotic Systems*, 7(1):19–26, 2010.

[12] KD Do and J Pan. Nonlinear formation control of unicycle-type mobile robots. *Robotics and Autonomous Systems*, 55(3):191–204, 2007.

[13] Douglas W Gage. Command control for many-robot systems. Technical report, DTIC Document, 1992.

[14] Tien-Ruey Hsiang, Esther M Arkin, Michael A Bender, Sándor P Fekete, and Joseph SB Mitchell. Algorithms for rapidly dispersing robot swarms in unknown environments. In *Algorithmic Foundations of Robotics V*, pages 77–94. Springer, 2004.

[15] Gal A Kaminka, Meytal Traub, Yehuda Elmaliach, Dan Erusalimchik, and Alex Fridman. On the use of teamwork software for multi-robot formation control. In *Proceedings of the 2013 international conference on Autonomous agents and multi-agent systems*, pages 1163–1164. International Foundation for Autonomous Agents and Multiagent Systems, 2013.

[16] Oussama Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *The international journal of robotics research*, 5(1):90–98, 1986.

[17] KN Krishnanand and D Ghose. Detection of multiple source locations using a glowworm metaphor with applications to collective robotics. In *Swarm Intelligence Symposium, 2005. SIS 2005. Proceedings 2005 IEEE*, pages 84–91. IEEE, 2005.

[18] Jonathan RT Lawton, Randal W Beard, and Brett J Young. A decentralized approach to formation maneuvers. *Robotics and Automation, IEEE Transactions on*, 19(6):933–941, 2003.

[19] David Lee and Michael Recce. Quantitative evaluation of the exploration strategies of a mobile robot. *The International Journal of Robotics Research*, 16(4):413–447, 1997.

[20] Luke Ludwig and Maria Gini. Robotic swarm dispersion using wireless intensity signals. In *Distributed Autonomous Robotic Systems 7*, pages 135–144. Springer, 2006.

[21] Ali Marjovi, João Gonçalo Nunes, Lino Marques, and Aníbal de Almeida. Multi-robot exploration and fire searching. In *Intelligent Robots and Systems, 2009. IROS 2009. IEEE/RSJ International Conference on*, pages 1929–1934. IEEE, 2009.

[22] Lino Marques, Urbano Nunes, and Anibal T de Almeida. Particle swarm-based olfactory guided search. *Autonomous Robots*, 20(3):277–287, 2006.

[23] Ignacio Mas and Christopher Kitts. Object manipulation using cooperative mobile multi-robot systems. In *Proceedings of the World Congress on Engineering and Computer Science (WCECS12)*, 2012.

[24] James McLurkin and Jennifer Smith. Distributed algorithms for dispersion in indoor environments using a swarm of autonomous mobile robots. In *Distributed Autonomous Robotic Systems 6*, pages 399–408. Springer, 2007.

[25] Nathan Michael, Jonathan Fink, and Vijay Kumar. Cooperative manipulation and transportation with aerial robots. *Autonomous Robots*, 30(1):73–86, 2011.

[26] Ryan Morlok and Maria Gini. Dispersing robots in an unknown environment. In *Distributed Autonomous Robotic Systems 6*, pages 253–262. Springer, 2007.

[27] Amir M. Naghsh, Jeremi Gancet, Andry Tanoto, and Chris Roast. Analysis and design of human-robot swarm interaction in firefighting. *RO-MAN 2008 - The 17th IEEE International Symposium on Robot and Human Interactive Communication*, pages 255–260, August 2008.

[28] Anuraag Pakanati and Maria Gini. Swarm dispersion via potential fields, leader election, and counting hops. In *Simulation, Modeling, and Programming for Autonomous Robots*, pages 485–496. Springer, 2010.

[29] Jacques Penders, L Alboul, and U Witkowski. A robot swarm assisting a human fire-fighter. *Advanced Robotics*, 2011.

[30] Riccardo Poli, James Kennedy, and Tim Blackwell. Particle swarm optimization. *Swarm intelligence*, 1(1):33–57, 2007.

[31] Bijan Ranjbar-Sahraei, Faridoon Shabaninia, Alireza Nemati, and S Stan. A novel robust decentralized adaptive fuzzy control for swarm formation of multiagent systems. *Industrial Electronics, IEEE Transactions on*, 59(8):3124–3134, 2012.

[32] Wei Ren and Randal Beard. Decentralized scheme for spacecraft formation flying via the virtual structure approach. *Journal of Guidance, Control, and Dynamics*, 27(1):73–82, 2004.

[33] Roberto Rozas, Jorge Morales, and Daniel Vega. Artificial smell detection for robotic navigation. In *Advanced Robotics, 1991.'Robots in Unstructured Environments', 91 ICAR., Fifth International Conference on*, pages 1730–1733. IEEE, 1991.

[34] Michael Rubenstein, Adrian Cabrera, Justin Werfel, and James Mclurkin. Collective Transport of Complex Objects by Simple Robots : Theory and Experiments. pages 47–54.

[35] Paul E Rybski, Sascha A Stoeter, Michael D Erickson, Maria Gini, Dean F Hougen, and Nikolaos Papanikolopoulos. A team of robotic agents for surveillance. In *Proceedings of the fourth international conference on autonomous agents*, pages 9–16. ACM, 2000.

[36] G Sandini, G Lucarini, and M Varoli. Gradient driven self-organizing systems. In *Intelligent Robots and Systems' 93, IROS'93. Proceedings of the 1993 IEEE/RSJ International Conference on*, volume 1, pages 429–432. IEEE, 1993.

[37] Weihua Sheng, Qingyan Yang, Jindong Tan, and Ning Xi. Distributed multi-robot coordination in area exploration. *Robotics and Autonomous Systems*, 54(12):945–955, 2006.

[38] Karansher Singh and Kikuo Fujimura. Map making by cooperating mobile robots. In *Robotics and Automation, 1993. Proceedings., 1993 IEEE International Conference on*, pages 254–259. IEEE, 1993.

[39] Agusti Solanas and Miguel Angel Garcia. Coordinated multi-robot exploration through unsupervised clustering of unknown space. In *Intelligent Robots and Systems, 2004.(IROS 2004). Proceedings. 2004 IEEE/RSJ International Conference on*, volume 1, pages 717–721. IEEE, 2004.

[40] Ken Sugawara, Nikolaus Correll, and Dustin Reishus. Object transportation by granular convection using swarm robots. *Distributed Algorithmic Robotics*, 2012.

[41] Emre Ugur, Ali E Turgut, and Erol Sahin. Dispersion of a swarm of robots based on realistic wireless intensity signals. In *Computer and information sciences, 2007. iscis 2007. 22nd international symposium on*, pages 1–6. IEEE, 2007.

[42] Israel A Wagner, Yaniv Altshuler, Vladimir Yanovski, and Alfred M Bruckstein. Cooperative cleaners: A study in ant robotics. *The International Journal of Robotics Research*, 27(1):127–151, 2008.

[43] Sung-Gil Wee, Yoon-Gu Kim, Suk-Gyu Lee, and Jinung An. Formation control based on virtual space configuration for multi-robot collective navigation. In *Ubiquitous Robots and Ambient Intelligence (URAI), 2013 10th International Conference on*, pages 556–557. IEEE, 2013.

[44] Brian Yamauchi. Frontier-based exploration using multiple robots. In *Proceedings of the second international conference on Autonomous agents*, pages 47–53. ACM, 1998.

[45] Dimitri Zarzhitsky, Diana F Spears, and William M Spears. Distributed robotics approach to chemical plume tracing. In *Intelligent Robots and Systems, 2005.(IROS 2005). 2005 IEEE/RSJ International Conference on*, pages 4034–4039. IEEE, 2005.

[46] Qingquan Zhang, Gerald E Sobelman, and Tian He. Gradient-based target localization in robotic sensor networks. *Pervasive and Mobile Computing*, 5(1):37–48, 2009.

[47] Robert Zlot, Anthony Stentz, M Bernardine Dias, and Scott Thayer. Multi-robot exploration controlled by a market economy. 2002.