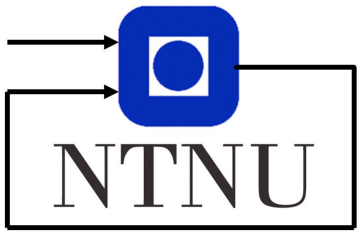
Department of Engineering Cybernetics

TTK4550 - Engineering Cybernetics, Specialization Project

Distributed coverage control for multi-agent multilateration networks

Author:
Magnus Berdal
Candidate number:
10078

Supervisors:
Behdad Aminian
Claudio Paliotta
Damiano Varagnolo



December 17th 2020

Abstract

First responders (FRs) in search-and-rescue missions frequently expose themselves to unknown and dangerous environments. With the increasing capabilities of micro areal vehicles (MAVs) and other mobile agent technologies, some of the risks FRs must take in order to save lives and secure the well-being of others can be mitigated. Deploying a swarm of MAVs into the environment to facilitate localization of the FRs and others can vastly improve the security of all parties involved in search-and-rescue missions.

In this report, multi-agent networks, and how these can be deployed into known areas to provide an infrastructure for localization is studied. A novel optimization-based distributed coverage control algorithm for deployment of a multi-agent multilateration network is proposed. The distributed objective function consists of two parts: local probability of coverage and dispersion from neighbors.

Simulations are performed in different environments and for varying swarm sizes. For all environments and swarm sizes simulations are performed where agents disregard dispersion from neighbors, and where agents actively disperse from their neighbors. The results obtained through simulations show that the initial coverage is always improved upon, when theoretically possible, by the proposed algorithm when agents disregard dispersing from their neighbors.

Simulations also show that actively dispersing agents, in some cases, can prevent the swarm from settling at clearly sub-optimal configurations where the environment is not, but could be, further explored and thus yield a substantially larger covered area. In other cases simulations show that actively dispersing agents cause a shift in behavior where agents reach a configuration in which maximum dispersion is reached, rather than the maximum covered area.

The results presented in this report show that the proposed coverage control algorithm cannot yet be utilized in real-world situations as it lacks robustness. However, this report lays the groundwork for further research and development. In an upcoming master's thesis, the multilateration network deployment problem will be further examined. This will hopefully yield results that can directly reduce the dangers to which FRs expose themselves.

This project has been carried out in collaboration with the INGENIOUS project funded by the European Union's Horizon 2020 research and innovation program under grant agreement No 833435 and the Norwegian Research Council project "Autonomous Underwater Fleets" under the grant agreement No 302435.

Contents

1 Introduction

A first responder (FR) is a person who is among those responsible for going immediately to the scene of an accident or emergency to provide assistance [1]. They will typically be employed by the emergency services such as the police, fire department, or health services, and take it upon themselves to secure the health of people, property, and the environment. Often this includes sacrificing their own safety in order to secure that of others. The need for quick action gives them no other choice.

Search-and-rescue personnel in particular expose themselves to dangerous environments. Burning buildings, collapsed or flooded caves etc. are just some examples of the rapid-changing hazardous environments in which FRs can find themselves. Often they enter without any knowledge of what is waiting for them on the other side. Imagine a burning building. The local fire department has just arrived. There is no time to spare, so the first firefighters enter to save the lives of those inside before the floor plan is inspected. It might not even be valid as the roaring fire continues to eat up the walls and support beams for the roof. Still, the firefighters enter in order to rescue those inside. If something is to happen to the entering firefighters there is no way to locate them, and the ones entering to save them still don't know what is waiting for them inside.

Advancements in technology can be used to limit the dangers to which FRs expose themselves. With the increasing performance and decreasing cost of mobile robots [2], and the ability to outfit them with whatever equipment one might think of, it is relevant to think of how this might be used to the advantage of first responders. Using disposable robots for tasks such as mapping unknown environments could drastically improve the safety of FRs. In the example stated above, mobile robots could be dispatched into the burning house and continuously feed the firefighters with a real-time map of the environment. Robots could also enter the building to set up an ad-hoc network, a dynamic and self-configuring network formed by a collection of mobile nodes [3], used for locating FRs inside GNSS denied environments. Alternatively, the swarm of agents could supply information about events happening, such as changes in the environment or the presence of poisonous gasses etc.

The INGENIOUS project is an EU-funded project that has as its mission to develop and test a next-generation integrated toolkit (NGIT) for collaborative response. This includes developing micro indoor aerial drones that are to enter dangerous buildings in order to create an indoor mesh network that allows for indoor localization of FRs. Developing a distributed deployment scheme in order to provide such a localization network is the task of this report.

2 Background

2.1 Potential, Limitations and Progress of the MAV

Micro Areal Vehicles (MAVs) have great potential in contributing to indoor search-and-rescue missions. Their small size and weight make them easy to transport and allows for rapid field deployment. Furthermore, they are agile, allowing them to operate in complex environments and access hard-to-get-to places.

Limitations of current technologies, prohibiting the existence of fully autonomous MAVs, are examined in [2]. They define the following requirements for a fully autonomous MAV:

- Inference: The ability to infer situational awareness from sensory measurements.
- Reasoning: The ability to define a mission based on abstract human-defined goals.
- Unsupervised learning: The ability to adapt and learn its own control strategies without human supervision.

Due to the constraints on the size of MAVs, the existence of fully autonomous MAVs is mainly dependent on the existence of small and efficient hardware components such as power supplies, sensors, and processors allowing them to quickly process incoming data and convert this information into actions. The authors of [2] conclude that as of March 2017 no fully autonomous MAV system exists.

Although a fully autonomous MAV system is yet to be realized, the advancement in transistor density shown in Figure 2, and the continuously diminishing price of microprocessors continue to allow for faster, and thus more complex, on-board computations. This allows for enhanced autonomous capabilities of the aircraft [2]. Increased onboard computation power enables the aircraft to perform computations of increased complexity locally without the aid of other computation units or human interference, while conforming with real-time constraints.

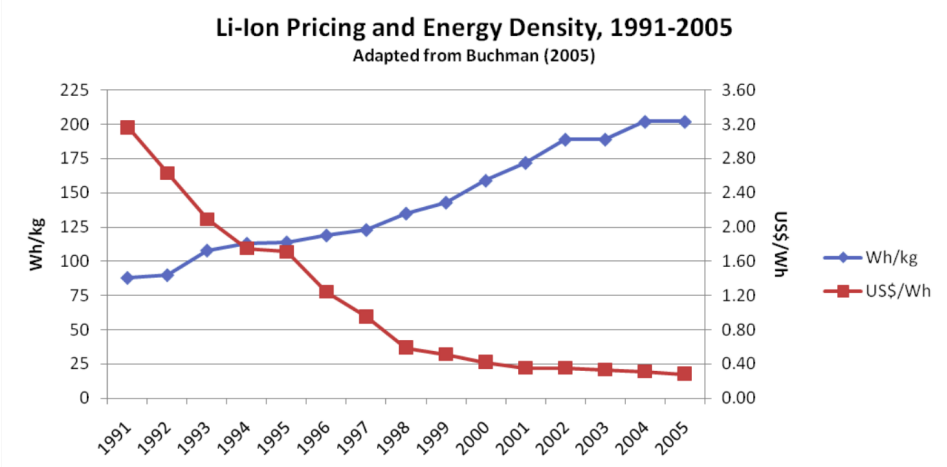


Figure 1: Evolution of energy density and cost of Lithium-ion batteries. Source: [4].

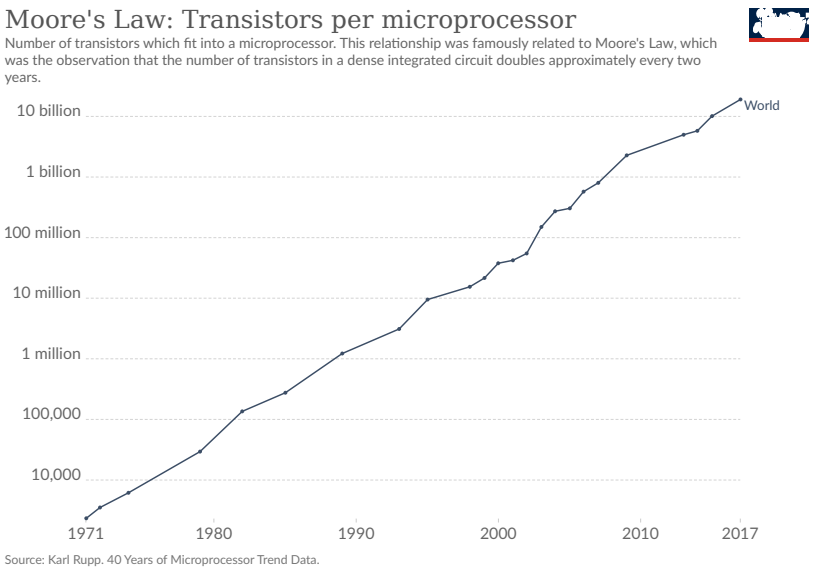


Figure 2: Evolution of the number of transistors per microprocessor.

Most MAVs utilize electrical propulsion systems. This is due to the ability to miniaturize electrical propulsion systems for MAV applications with minimal loss of efficiency. Additionally, electrical propulsion systems have faster response time than combustion-based systems [2]. Energy is typically stored on LiPo[1] batteries. Figure 1 shows how the energy density and price of Lithium-ion batteries have evolved over the last decades. As the energy density of Lithium-ion batteries already have, and is projected to continue to increase [5], the endurance of MAVs have, and will continue to, increase. Furthermore, more power-consuming equipment can be fitted on the MAVs, thus enabling MAVs to take on more complex and long-lasting missions, resulting in increased autonomy.

Not only hardware components have made leaps over the last decades. The advancement in autonomy of MAV systems through the creation of efficient and lightweight software has been a field of great scientific interest. The Robotics & Perception Group at the University of Zürich have performed excessive research into the field of visual-inertial based real-time navigation and mapping by MAVs in GNSS denied environments [6, 7]. The goal of which is enabling MAVs to localize themselves in and map unknown environments without external aid.

In [6] a fast and robust visual-odometry algorithm (SVO) is presented. The method is faster than previously presented methods and is therefore especially useful for MAV operations as it should be able to run on an onboard computer with limited processing power. It is concluded that the method is especially useful for state estimation onboard MAVs as the algorithm runs at a high enough rate to provide accurate state estimates. Furthermore, the use of depth-filters yields an accurate map of the environment with few outliers.

In [7] Scaramuzza et al. present a system for autonomous mapping of unknown indoor and outdoor environments performed by a single MAV using SVO. The MAV is supplied with a trajectory to follow, and using only onboard processing and sensing, follows said trajectory and provides at the same time a real-time map of the environment.

[1] Lithium-ion polymer

Developments such as those in [6] and [7] have increased the autonomous capabilities of MAVs, and as time passes more efficient software solutions will continue to do so.

2.2 The Deployment Problem & Blanket Coverage

Multi-agent distributed control is the task of making a set of agents work together to fulfill some collective goal, and doing so with only local information. One such task is studied in this report, self-deployment: Make a swarm of robots "[...] deploy themselves in an environment without central coordination"[8]. A subcategory of the deployment problem is referred to in the literature as the blanket coverage problem. In [8] the blanket coverage problem is defined as the task of finding a static configuration of agents such that a cost function is maximized over an area.

Cassandras and Zhong present in [9] a distributed method for multi-agent blanket coverage. They approach the blanket coverage problem using a probability-based objective function. With their approach, the goal of the set of agents is maximizing the joint detection probability of randomly occurring events in the environment. They also present an algorithm that preserves the connectivity of the network of agents. They conclude that using their method, the swarm of agents reach a configuration corresponding to a local maximum of the joint detection probability function. Furthermore, they find that when connectivity preservation is imposed, the swarm settles at a configuration in which the joint detection probability is smaller than when connectivity of the network is not taken into account.

In [10] Cassandras and Zhong "[...] address the problem of multiple local optima commonly arising in optimization problems for multi-agent systems[...]". Utilizing the same probability-based objective function as in [9], boosting functions are applied to encourage agents to explore poorly covered regions of the area and escape local optima. Three families of boosting functions are defined and tested through simulations. It is shown that the objective value post boosting is no worse than before.

Howard et.al. present in [11] another approach to the blanket coverage problem: virtual potential fields. It is assumed that agents can determine the range and bearing to both nearby agents and obstacles. Drawing inspiration from electrostatic fields, agents viewed as point charges who exert repulsive forces on one another, causing them to spread. Obstacles also exert forces on the agents so that agents do not collide with obstacles in the environment. They show through simulations that the potential field approach can be used to deploy multi-agent networks for the means of blanket coverage, and conclude that area coverage emerges from a combination of "[...] purely local rules"[11].

2.3 Multilateration

Multilateration is the process of determining the positions of unknown points in space by measurements of distances from known points [12]. In order to perform this task in two-dimensional space, at least three known points are needed.

Given $n \geq 3$ beacons located at positions $\mathbf{x}_a \in \mathbb{R}^2$, $0 \leq a < n$ where not all points lie on a single line, the location of an entity, denoted by $\mathbf{y} \in \mathbb{R}^2$, can be determined as follows:

1. The entity broadcasts signal and starts a timer at t_0.
2. Beacons at $\mathbf{x}_a$, $0 \leq a < n$ receive broadcasted signal and immediately responds with a packet containing $\mathbf{x}_a$.
3. When receiving the packet from the beacon at $\mathbf{x}_a$, the entity stores the time of reception in a variable $t_{1,a}$.
4. When at least 3 beacons have responded, the entity calculates the distance from itself to beacon at $\mathbf{x}_a$: $d_a = \frac{1}{2}s(t_{1,a} - t_0)$, where s is the propagation speed of the signal. The factor $\frac{1}{2}$ is due to the signal traveling two times the distance between the entity and the beacon placed at $\mathbf{x}_a$ (the ping travels from the entity to the agent, and the packet sent by the agent travels back again).
5. Based on the distances, d_a, and the positions of the beacons the entity can determine its position by calculating the point where circles centered at $\mathbf{x}_a$ with radii d_a intersect.

If sufficiently many beacons respond an ML (Maximum Likelihood) estimator of the position of the entity can be computed [13]. Defining the error function:

$$e_a(\mathbf{y}) = s(t_{1,a} - t_{0,a}) - \|\mathbf{x}_a - \mathbf{y}\| = d_a - \|\mathbf{x}_a - \mathbf{y}\| \tag{1}$$

An estimate of the position of the entity is obtained by solving:

$$\hat{\mathbf{y}} = \arg\min_{\mathbf{y}} \mathbf{E}^T\mathbf{E}, \quad \mathbf{E} = \begin{bmatrix} e_0(\mathbf{y}) \\ \vdots \\ e_{n-1}(\mathbf{y}) \end{bmatrix} \tag{2}$$

The position estimation error is affected by the measurement errors, by the geometry relating sensors and target, and by the estimation algorithm [14]. As the pings travel at large velocities (the speed of light) the resolution of the internal clock of the parties involved set a bound on the accuracy of the estimated position. As was found in [15], multilateration of the unknown position of an entity is most accurate when the entity is placed nearby or within the convex hull of the beacons. Hence spreading the beacons is desirable in order to obtain accurate position estimates. Figure 3 shows how the position of an entity can be determined from the known positions of 3 agents.

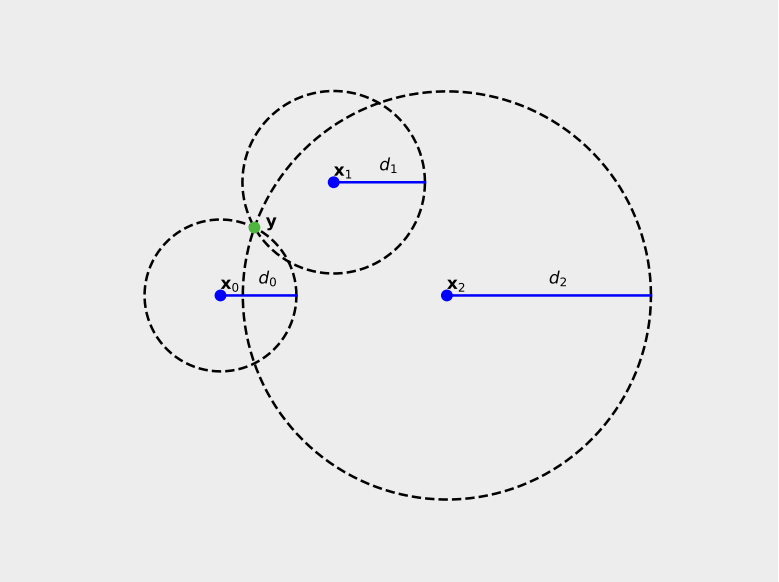


Figure 3: Position, $\mathbf{y}$, of entity determined by multilateration using known positions of 3 beacons placed at $\mathbf{x}_0$, $\mathbf{x}_1$ and $\mathbf{x}_2$. The distances d_0, d_1 and d_2 are determined by sending pings between the entity at $\mathbf{y}$ and the beacons. The point $\mathbf{y}$ lies at the intersection of the circles placed at $\mathbf{x}_a$ with radius d_a, $a = 0, 1, 2$.

3 Nomenclature

For ease of the reader, a summary of the notation and mathematical operators used are presented below.

- Ω: Mission space.
- $\mathcal{O}$: Set of obstacles inside the mission space.
- $\mathcal{F}$: Feasible space.
- $\mathbf{x}_a$: Position of agent a.
- r_a: Maximum communication range of agent a.
- $\hat{p}(\mathbf{x}_a, \mathbf{y})$: Local probability of agent a wrt. point $\mathbf{y}$.
- D_a: Communication disk of agent a.
- V_a: Visible set of agent a.
- U_a: Invisible set of agent a.
- $\mathcal{B}_a$: Neighbors of agent a.
- $\mathcal{C}_a$: Participants in the swarm who are not neighbors of agent a.
- $\mathbf{X}_{\mathcal{S}}$: Positions of all agents in the swarm $\mathcal{S}$.
- $\mathbf{X}_{\mathcal{S},a}$: Position of participant a in the swarm $\mathcal{S}$.
- $\Phi^n(\mathbf{X}_{\mathcal{S}}, \mathbf{y})$: Probability of n members in the swarm $\mathcal{S}$ being able to communicate with entity at position $\mathbf{y}$.
- $\Phi^{n^+}(\mathbf{X}_{\mathcal{S}}, \mathbf{y})$: Probability of n or more members in the swarm $\mathcal{S}$ being able to communicate with entity at position $\mathbf{y}$.
- $L(\mathbf{X}_{\mathcal{B}_a \cap \{a\}})$: Local probability of coverage for agent a.
- $D(\mathbf{x}_a, \mathbf{x}_j)$: Proximity of agent j relative to agent a.
- $H(\mathbf{X}_{\mathcal{B}_a \cap \{a\}})$: Local objective for agent a. Equal to the local probability of coverage for agent a if active dispersion is not applied.
- For a set $\mathcal{S}$ defined by $\mathcal{S} = \{s_0 \dots s_{N-1}\},\ N < \infty$ the following operators are defined:
  - $|\mathcal{S}| = N$: The size of $\mathcal{S}$.
  - $\text{Comb}(\mathcal{S}, n) = \{\mathcal{A} : \mathcal{A} \subseteq \mathcal{S}, |\mathcal{A}| = n\}$: the set of all subsets of $\mathcal{S}$ of size n.
- For a set $\mathcal{S}$ of points $\mathbf{x} \in \mathbb{R}^N$ defined by $\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^N : f(\mathbf{x}) \leq \mathbf{0}\}$ the following operators are defined:
  - $\delta\mathcal{S} = \{\mathbf{x} \in \mathbb{R}^N : f(\mathbf{x}) = \mathbf{0}\}$: The boundary of $\mathcal{S}$.
  - $\text{int}(\mathcal{S}) = \{\mathbf{x} \in \mathbb{R}^N : f(\mathbf{x}) < \mathbf{0}\}$: The interior of $\mathcal{S}$.
  - $A(\mathcal{S})$: The area of the space bounded by $\delta\mathcal{S}$.

4 System Description

4.1 Feasible space

As in [10], a *mission space*, Ω, is defined as a simple polygon [16]. Within the mission space there exists $N_o \geq 0$ obstacles, each one of which is defined as a simple polygon. The set of all obstacles, $\mathcal{O}$, is defined according to (3).

$$\mathcal{O} = \begin{cases} \{o_0 \dots o_{N_o-1}\} & , N_o > 0 \\ \emptyset & , N_o = 0 \end{cases}. \tag{3}$$

The obstacles in $\mathcal{O}$ constrains the movement of entities within the mission space, as it is not possible to penetrate the boundary of an obstacle. Due to this, once an entity is inside Ω, it is constrained to be positioned within Ω and outside $\text{int}(o)\ \forall\ o \in \mathcal{O}$. From this the *feasible space*, $\mathcal{F}$, is defined as all points where it is possible to place an entity:

$$\mathcal{F} = \{\mathbf{y} \in \mathbb{R}^2 : \mathbf{y} \in \Omega,\ \mathbf{y} \notin \text{int}(o)\ \forall\ o \in \mathcal{O}\} = \Omega \setminus \bigcup_{o \in \mathcal{O}} \text{int}(o). \tag{4}$$

4.2 Agent

An *agent* and its properties are defined in a similar manner as in [10]. An agent, denoted by an index a, is defined by its position $\mathbf{x}_a \in \mathbb{R}^2$ and its maximum range of communication r_a. From this the communication disk of an agent a is defined:

$$D_a = \{\mathbf{y} \in \mathbb{R}^2 : \|\mathbf{x}_a - \mathbf{y}\| \leq r_a\}. \tag{5}$$

Assuming line-of-sight (LoS) communication, meaning an agent cannot communicate with an entity if there is an obstacle or a mission space wall between them, the *visible set* of agent a is defined:

$$V_a = \{\mathbf{y} \in \mathbb{R}^2 : \mathbf{y} \in D_a, \lambda\mathbf{y} + (1-\lambda)\mathbf{x}_a \in \mathcal{F}\ \forall\ 0 \leq \lambda \leq 1\}. \tag{6}$$

The counterpart to the visible set, called the invisible set of agent a, is simply defined as:

$$U_a = \mathcal{F} \setminus V_a. \tag{7}$$

An example of the visible set for an agent is shown in Figure 4.

The probability of an agent a being able to communicate with another entity positioned at a point $\mathbf{y}$, from now on called the local probability of agent a with respect to $\mathbf{y}$, is defined according to:

$$\hat{p} : \mathbb{R}^2 \to [0,1], \quad \hat{p}(\mathbf{x}_a, \mathbf{y}) = \begin{cases} p(\|\mathbf{x}_a - \mathbf{y}\|) & , \mathbf{y} \in V_a \\ 0 & , \mathbf{y} \in U_a \end{cases}, \tag{8}$$

where $p(\cdot)$ is a non-increasing function of its argument.

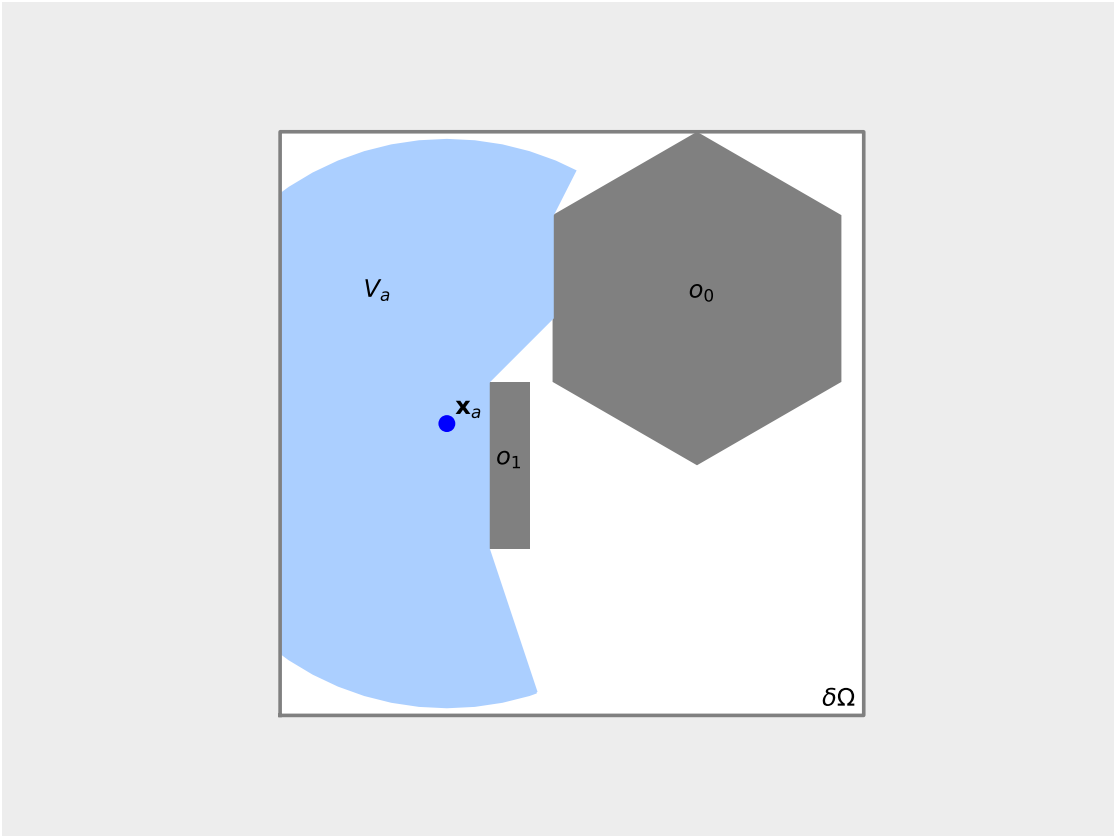


Figure 4: Visible set (light blue) for the agent placed at $\mathbf{x}_a$ in a rectangular mission space Ω with two obstacles ($\mathcal{O} = \{o_0, o_1\}$)

4.3 Swarm

A *swarm* $\mathcal{S}$ of size N is a set of agents $\{a_0 \dots a_{N-1}\}$. The state of the swarm is described by the state of its participants, and is expressed as a vector $\mathbf{X}_\mathcal{S} \in \mathbb{R}^{2N}$ as shown in (9).

$$\mathbf{X}_\mathcal{S} = \begin{bmatrix} \mathbf{x}_{a_0} \\ \vdots \\ \mathbf{x}_{a_{N-1}} \end{bmatrix}, \quad \mathbf{X}_{\mathcal{S},i} = \mathbf{x}_{a_i}. \tag{9}$$

The maximum radii of communication of the swarm are represented as the vector $\mathbf{r} \in \mathbb{R}^N$ as shown in (10).

$$\mathbf{r}_\mathcal{S} = \begin{bmatrix} r_{a_0} & \dots & r_{a_{N-1}} \end{bmatrix}^T \tag{10}$$

In [17] the probability of success in x out of n independent trials is defined as:

$$f_n(x;\mathbf{p}) = \sum_{\mathcal{A}\in\mathcal{F}_x} \Big(\prod_{i\in\mathcal{A}} p_i\Big)\Big(\prod_{i\in\mathcal{A}^c} 1-p_i\Big), \tag{11}$$

where $\mathbf{p} = \begin{bmatrix} p_0 & \dots & p_{n-1}\end{bmatrix}$ and p_i denotes the probability of success at the ith trial. $\mathcal{F}_x$ is equivalent to $\mathrm{Comb}(\{0\dots n-1\}, x)$, and $\mathcal{A}^c = \{0\dots n-1\}\setminus\mathcal{A}$.

Assuming that the distributions for all agents in a swarm $\mathcal{S}$ are independent, (11) can be used to express the probability of n members in the swarm being able to communicate with an entity at a point $\mathbf{y}$:

$$\Phi^n(\mathbf{X}_\mathcal{S},\mathbf{y}) = \sum_{A\in Comb(\mathcal{S},n)} \prod_{a\in\mathcal{A}} \hat{p}(\mathbf{x}_a,\mathbf{y}) \prod_{a\in\mathcal{S}\setminus\mathcal{A}} \big(1-\hat{p}(\mathbf{x}_a,\mathbf{y})\big). \tag{12}$$

Using (12) the probability of *at least* n members in a swarm $\mathcal{S}$ being able to communicate with an entity placed at $\mathbf{y}$ is defined as:

$$\Phi^{n^+}(\mathbf{X}_\mathcal{S},\mathbf{y}) = 1 - \sum_{i=0}^{n-1} \Phi^i(\mathbf{X}_\mathcal{S},\mathbf{y}). \tag{13}$$

5 Problem formulation

Using a swarm, $\mathcal{N}$, of N mobile agents with homogenous communication range, the swarm should be deployed in a mission space, Ω. The agents should, without centralized control, spread throughout the mission space and position themselves such that they can be used as beacons in a multilateration scheme in order to deliver precise positional data to entities entering the mission space.

5.1 Coverage

It is clear from Section 2.3 that in two-dimensional space three or more agents are needed to perform the task of multilateration. Hence a point $\mathbf{y}$ is said to be *covered* iff. the local probability of at least three agents with respect to $\mathbf{y}$ is non-zero. Thus, given a swarm $\mathcal{N}$, the point $\mathbf{y}$ is covered iff.

$$\Phi^{3^+}(\mathbf{X}_{\mathcal{N}}, \mathbf{y}) > 0, \tag{14}$$

where $\Phi^{3^+}(\mathbf{X}_{\mathcal{N}}, \mathbf{y})$ will be referred to as the probability of coverage.

5.2 Objective function derivation

The objective function presented here is inspired by [10], but is modified for the purpose of multilateration. Here a stricter definition of coverage is applied (see Section 5.1).

In order to formulate a distributed optimization algorithm, the probability of coverage is rewritten with focus on a single drone a. The results are stated in (15) and the derivations are available in Appendix A.

$$\Phi^{3^+}(\mathbf{X}_{\mathcal{N}}, \mathbf{y}) = \Phi^{3^+}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y}) + \Phi^{2}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y})\hat{p}(\mathbf{x}_a, \mathbf{y}) \tag{15a}$$

$$= (1 - \hat{p}(\mathbf{x}_a, \mathbf{y}))\Phi^{3^+}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y}) + \hat{p}(\mathbf{x}_a, \mathbf{y})\Phi^{2^+}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y}). \tag{15b}$$

From (15b) it is clear that the probability of the point $\mathbf{y}$ being covered can be seen as an interpolation between two probability measures, with the local probability of agent a as the interpolation variable. Low local probability of agent a means that the probability of coverage supplied by the swarm to a larger extent depends on the coverage supplied by the swarm excluding agent a. In the extreme case where the local probability of agent a is zero, the probability of covering $\mathbf{y}$ depends only on the coverage supplied by the swarm excluding agent a.

Higher local probability of agent a means that the contribution of a towards covering the point $\mathbf{y}$ is greater. Thus less weight is put on the probability of the swarm excluding agent a covering the point. Instead, more weight is put on the probability of at least *two* other agents being able to communicate with an entity at $\mathbf{y}$. This is due to the fact that the probability of agent a being able to communicate with the entity is higher, and only two or more agents must be able to communicate with the entity at $\mathbf{y}$ to make the total number of agents covering $\mathbf{y}$ three or more.

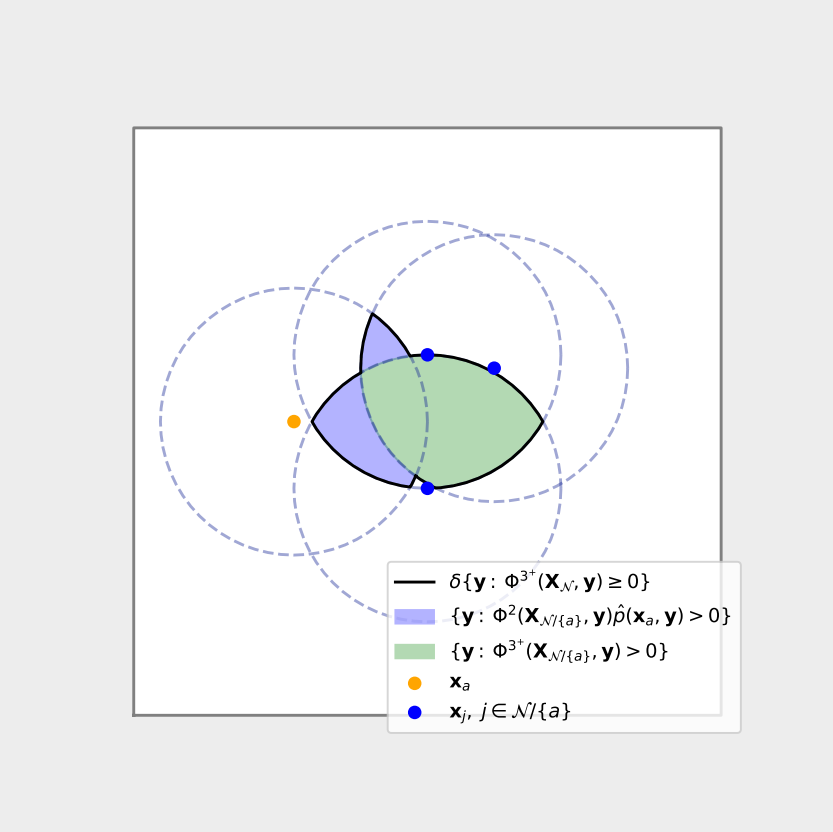


Figure 5: Non-zero regions of integrands in (16). Note that perturbing the orange circle (position of agent a) does not affect the green region, as it is defined only by the intersections of the disks surrounding the blue points (other agents in the swarm).

Using (15a) the overall probability of coverage over the feasible space can be written as:

$$\begin{aligned} P(\mathbf{X}_{\mathcal{N}}) &= \int_{\mathcal{F}} \Phi^{3^+}(\mathbf{X}_{\mathcal{N}}, \mathbf{y}) d\mathbf{y} = \int_{\mathcal{F}} \Phi^{3^+}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y}) + \Phi^{2}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y})\hat{p}(\mathbf{x}_a, \mathbf{y}) d\mathbf{y} \\ &= \int_{\mathcal{F}} \Phi^{3^+}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y}) d\mathbf{y} + \int_{\mathcal{F}} \Phi^{2}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y})\hat{p}(\mathbf{x}_a, \mathbf{y}) d\mathbf{y}. \end{aligned} \tag{16}$$

The first term in (16) is independent of the position of agent a in both its domain and integrand. This independence is visualized in Figure 5. Thus the overall coverage probability over $\mathcal{F}$ can be rewritten as:

$$P(\mathbf{X}_{\mathcal{N}}) = P(\mathbf{X}_{\mathcal{N}\setminus\{a\}}) + P_a(\mathbf{X}_{\mathcal{N}}), \tag{17}$$

where the *local* probability of coverage for agent a is defined as:

$$P_a(\mathbf{X}_{\mathcal{N}}) = \int_{\mathcal{F}} \Phi^{2}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y})\hat{p}(\mathbf{x}_a, \mathbf{y}) d\mathbf{y}. \tag{18}$$

As in [10] it is noted that from the viewpoint of agent a, the swarm can be partitioned into three disjoint sets: $\{a\}$, $\mathcal{B}_a$ and $\mathcal{C}_a$. Exploiting that all agents have homogenous range of communica-

tion, i.e.

$$r_a = r \; \forall \; a \in \mathcal{N}, \tag{19}$$

the sets $\mathcal{B}_a$ and $\mathcal{C}_a$ are defined as:

$$\mathcal{B}_a = \{j \in \mathcal{N} \setminus \{a\} : \|\mathbf{x}_a - \mathbf{x}_j\| \leq 2r\} \tag{20a}$$
$$\mathcal{C}_a = \{j \in \mathcal{N} \setminus \{a\} : \|\mathbf{x}_a - \mathbf{x}_j\| > 2r\}. \tag{20b}$$

The set $\mathcal{B}_a$, from now on called the neighbors of a, contains all agents in the swarm, $\mathcal{N}$, whose communication disks form a non-empty intersection with that of a. $\mathcal{C}_a$ contains all agents whose communication disks do not intersect with that of a. Furthermore note that

$$\begin{aligned} \mathcal{B}_a \cap \mathcal{C}_a &= \emptyset \\ \mathcal{B}_a \cup \mathcal{C}_a \cup \{a\} &= \mathcal{N}. \end{aligned} \tag{21}$$

Applying (20) to (18) yields:

$$\begin{aligned} P_a(\mathbf{X}_\mathcal{N}) &= \int_\mathcal{F} \Phi^2(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y}) \hat{p}(\mathbf{x}_a, \mathbf{y}) d\mathbf{y} \\ &= \int_\mathcal{F} \Big(\Phi^2(\mathbf{X}_{\mathcal{B}_a}, \mathbf{y}) + \Phi^2(\mathbf{X}_{\mathcal{C}_a}, \mathbf{y}) + \Phi^1(\mathbf{X}_{\mathcal{B}_a}, \mathbf{y}) \Phi^1(\mathbf{X}_{\mathcal{C}_a}, \mathbf{y}) \Big) \hat{p}(\mathbf{x}_a, \mathbf{y}) d\mathbf{y}. \end{aligned} \tag{22}$$

Partitioning the domain of integration into the visible set and invisible set of agent a, and noting that $\hat{p}(\mathbf{x}_j, \mathbf{y}) = 0 \; \forall \; j \in \mathcal{C}_a, \; \mathbf{y} \in V_a$ such that $\Phi^n(\mathbf{X}_{\mathcal{C}_a}, \mathbf{y}) = 0 \; \forall \; n \in \mathbb{Z}^+, \; \mathbf{y} \in V_a$, and $\hat{p}(\mathbf{x}_a, \mathbf{y}) = 0 \; \forall \; \mathbf{y} \in U_a$ yields:

$$\begin{aligned} P_a(\mathbf{X}_\mathcal{N}) &= \int_{V_a} \Big(\Phi^2(\mathbf{X}_{\mathcal{B}_a}, \mathbf{y}) + \Phi^2(\mathbf{X}_{\mathcal{C}_a}, \mathbf{y}) + \Phi^1(\mathbf{X}_{\mathcal{B}_a}, \mathbf{y}) \Phi^1(\mathbf{X}_{\mathcal{C}_a}, \mathbf{y}) \Big) \hat{p}(\mathbf{x}_a, \mathbf{y}) d\mathbf{y} \\ &+ \int_{U_a} \Big(\Phi^2(\mathbf{X}_{\mathcal{B}_a}, \mathbf{y}) + \Phi^2(\mathbf{X}_{\mathcal{C}_a}, \mathbf{y}) + \Phi^1(\mathbf{X}_{\mathcal{B}_a}, \mathbf{y}) \Phi^1(\mathbf{X}_{\mathcal{C}_a}, \mathbf{y}) \Big) \hat{p}(\mathbf{x}_a, \mathbf{y}) d\mathbf{y} \\ &= \int_{V_a} \Phi^2(\mathbf{X}_{\mathcal{B}_a}, \mathbf{y}) p(\|\mathbf{x}_a - \mathbf{y}\|) d\mathbf{y} = L(\mathbf{X}_{\mathcal{B}_a \cup \{a\}}), \end{aligned} \tag{23}$$

where it is used that $\hat{p}(\mathbf{x}_a, \mathbf{y}) = p(\|\mathbf{x}_a - \mathbf{y}\|) \; \forall \; \mathbf{y} \in V_a$. Thus the local probability of coverage for an agent a is dependent on the position, $\mathbf{x}_a$, of agent a in both domain and integrand, and the positions of the neighbors of agent a. The set of agents consisting of agent a and its neighbors will be referred to as agent a's local swarm.

As discussed in Section 2.3 it is beneficial that agents used for multilateration spread out to some extent in order to ensure sufficient accuracy of multilateration. Furthermore ensuring that agents spread throughout the mission space is desirable. In order to explicitly encourage agents to spread, a term penalizing an agent a for being close to another agent j is introduced. When agents are close to each other this term should induce some velocity in the agents causing them to spread. In [11] virtual potential fields are used to disperse agents. There the potential between an agent a and its neighbor j is modelled as:

$$U = -k\frac{1}{\|\mathbf{x}_a - \mathbf{x}_j\|}, \tag{24}$$

where k is a constant defining the strength of the field. Drawing inspiration from this the proximity term between two agents, a and j, is defined according to:

$$D(\mathbf{x}_a, \mathbf{x}_j) = k_1 e^{-k_2\|\mathbf{x}_a - \mathbf{x}_j\|}. \tag{25}$$

As opposed to (24), (25) is defined for all $\mathbf{x}_a$, $\mathbf{x}_j$ allowing for easier implementation. The function has two tunable parameters, $k_1,\ k_2 \geq 0$, whose effects are show in Figure 6.

Using (23) and (25) the *local* objective function for an agent a is defined as:

$$H(\mathbf{X}_{\mathcal{B}_a \cup \{a\}}) = L(\mathbf{X}_{\mathcal{B}_a \cup \{a\}}) - \sum_{j \in \mathcal{B}_a} D(\mathbf{x}_a, \mathbf{x}_j). \tag{26}$$

Due to the negative sign in front of the sum of proximity functions this will be referred to this as the active dispersion term.

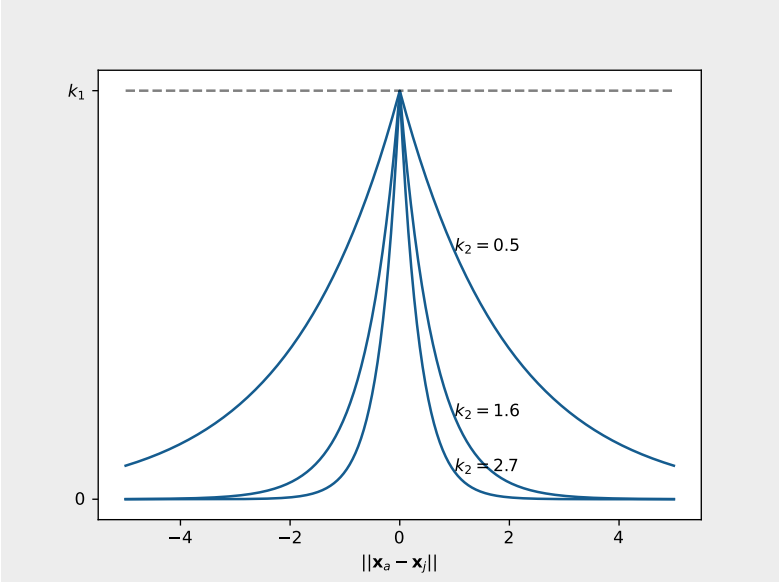


Figure 6: Proximity term in (25) for an agent, a, and its neighbor, j.

5.3 Constraints

It is clear from Section 2.3 that agents used as beacons in a multilateration scheme cannot be positioned on a straight line, as this would make it impossible to uniquely determine the unknown position of an entity. Only agents close to an entity whose position is to be determined will take part in the multilateration scheme. Due to this, it is imposed that only agents that together can be used for multilateration must satisfy the non-linear position requirement.

For an agent a with N neighbors $j \in \mathcal{B}_a$, $|B_a| = N$ the matrix $\mathbf{V}$ is constructed according to:

$$\mathbf{V}(\mathbf{X}_{\mathcal{B}_a}) = \begin{bmatrix} \mathbf{X}_{\mathcal{B}_{a,0}} - \mathbf{X}_{\mathcal{B}_{a,1}} & \dots & \mathbf{X}_{\mathcal{B}_{a,0}} - \mathbf{X}_{\mathcal{B}_{a,N-1}} \end{bmatrix}. \tag{27}$$

If agent a has less than two neighbors, or all of agent a's neighbors are positioned on a straight line, the matrix $\mathbf{V}(\mathbf{X}_{\mathcal{B}_a})$ will not span $\mathbb{R}^2$, i.e. $\text{Rank}(\mathbf{V}(\mathbf{X}_{\mathcal{B}_a})) < 2$.

Assuming that agent a has two or more neighbors, $|\mathcal{B}_a| \geq 2$, and all neighbors of a are positioned on a straight line, it is desirable that agent a is positioned sufficiently far away from the line. This constraint is implemented as follows: Without loss of generality, define:

$$\mathbf{v}(\mathbf{x}_a) = \mathbf{x}_a - \mathbf{X}_{\mathcal{B}_a,0} \tag{28a}$$

$$\mathbf{l} = \mathbf{X}_{\mathcal{B}_a,1} - \mathbf{X}_{\mathcal{B}_a,0}, \tag{28b}$$

meaning $\mathbf{v}(\mathbf{x}_a)$ is the vector from agent a to its first neighbor, and $\mathbf{l}$ is the vector between agent a's first and second neighbor. Thus $\mathbf{l}$ is parallel to the line that goes through all of agent a's neighbors. Equation (3.98) in [18] is used to project $\mathbf{v}(\mathbf{x}_a)$ into $\mathbf{l}$. The component of $\mathbf{v}(\mathbf{x}_a)$ that is parallel to the line through agent a's neighbor i and j is obtained as:

$$\mathbf{v}_{\parallel}(\mathbf{x}_a) = \frac{\mathbf{v}(\mathbf{x}_a)^T\mathbf{l}}{\mathbf{l}^T\mathbf{l}}\mathbf{l}. \tag{29}$$

The component of $\mathbf{v}(\mathbf{x}_a)$ perpendicular to the line through agent a's neighbors is obtained as:

$$\mathbf{v}_{\perp}(\mathbf{x}_a) = \mathbf{v}(\mathbf{x}_a) - \mathbf{v}_{\parallel}(\mathbf{x}_a). \tag{30}$$

Now the distance from agent a to the line through its neighbors can be computed as:

$$\|\mathbf{v}_{\perp}(\mathbf{x}_a)\| = \left\| \mathbf{v}(\mathbf{x}_a) - \frac{\mathbf{v}(\mathbf{x}_a)^T\mathbf{l}}{\mathbf{l}^T\mathbf{l}}\mathbf{l} \right\|. \tag{31}$$

Seen as it is impossible to place an agent such that it lays on a straight line through all it's neighbors if the neighbors do not already lay on a straight line, it is demanded that the non-linear position constraint must be fulfilled only when $\text{Rank}(\mathbf{V}(\mathbf{X}_{\mathcal{B}_a})) < 2$, where $\mathbf{V}(\cdot)$ is defined in (27). The non-linear position constraint is defined as:

$$\left\| \mathbf{v}(\mathbf{x}_a) - \frac{\mathbf{v}(\mathbf{x}_a)^T\mathbf{l}}{\mathbf{l}^T\mathbf{l}}\mathbf{l} \right\| \geq d_{min}, \tag{32}$$

where d_{min} is a tunable parameter that defines how close an agent is allowed get to the line through it's neighbors, and $\mathbf{v}$ and $\mathbf{l}$ are defined in (28).

Furthermore it is demanded that any two agents must be some minimum distance apart at any given time. This is due to the fact that agents colliding could cause damage to the hardware, and possibly render them unusable. This constraint is modelled as:

$$\|\mathbf{x}_a - \mathbf{x}_j\| \geq r_{min} \; \forall \; j \in \mathcal{B}_a, \tag{33}$$

where r_{min} is a tunable parameter that sets a limit to how close an agent can be positioned to any of its neighbors.

5.4 Optimization problem formulation

The non-linear position constraint (32) presented in Section 5.3 is dependent on the neighbor set of agent a, $\mathcal{B}_a$, as it should only be imposed if the neighbors of agent a lay on a straight line. Using the objective function in (26) and the constraints in discussed in Section 5.3, the local optimization problem for an agent a is defined as:

$$\max_{\mathbf{x}_a} \; H(\mathbf{X}_{\mathcal{B}_a \cup \{a\}}) \tag{34a}$$

$$\text{s.t. } \mathbf{x}_a \in \mathcal{F} \tag{34b}$$

$$|\{j \in \mathcal{B}_a : \|\mathbf{x}_a - \mathbf{x}_j\| \leq 2r\}| \geq 2 \tag{34c}$$

$$\|\mathbf{x}_a - \mathbf{x}_j\| \geq r_{min} \; \forall \; j \in \mathcal{B}_a \tag{34d}$$

$$\left\| \mathbf{v}(\mathbf{x}_a) - \frac{\mathbf{v}(\mathbf{x}_a)^T \mathbf{l}}{\mathbf{l}^T \mathbf{l}} \mathbf{l} \right\| \geq d_{min}, \quad \text{iff. Rank}(\mathbf{V}(\mathbf{X}_{\mathcal{B}_a})) < 2 \tag{34e}$$

where $\mathbf{V}(\cdot)$ is defined in (27) and $\mathbf{v}(\cdot)$ and $\mathbf{l}$ are defined in (28).

In situations where the local probability of coverage of agent a is constant, i.e. $\frac{\partial L(\mathbf{X}_{\mathcal{B}_a \cup \{a\}})}{\partial \mathbf{x}_a} = \mathbf{0}$, the solution of (34a) will be achieved at the position $\mathbf{x}_a$ that maximizes the dispersion term in (26). This will cause agent a to move as far away from its neighbors as possible. Such behavior is undesirable as this might render agent a neighbor-less. An agent a without neighbors fulfills $\mathcal{B}_a = \emptyset$ and hence $H(\mathbf{X}_{\mathcal{B}_a \cup \{a\}}) \equiv 0$. Thus re-optimization will result in $\mathbf{x}_a$ being unaltered, causing agent a to stay at $\mathbf{x}_a$ indefinitely. To prevent such behavior the constraint (34c) is imposed. This will prevent agent a from completely disconnecting from its local swarm.

Furthermore, observe that $\text{Rank}(\mathbf{V}(\mathbf{X}_{\mathcal{B}_a})) < 2$ if either all neighbors of agent a lie on a straight line or agent a has less than two neighbors. If agent a has less than two neighbors it is not possible to draw a line between the neighbors of agent a, and thus the non-linear neighbor constraint (34e) is undefined. This further motivates imposing the constraint that agent a should have at least two neighbors at the solution to (34).

The optimization problem formulated in (34) is non-convex [19]. The objective function is generally non-concave and the constraints form a non-convex feasible set. In Figure 7 a visualization of the objective function is shown for an agent with 5 neighbors.

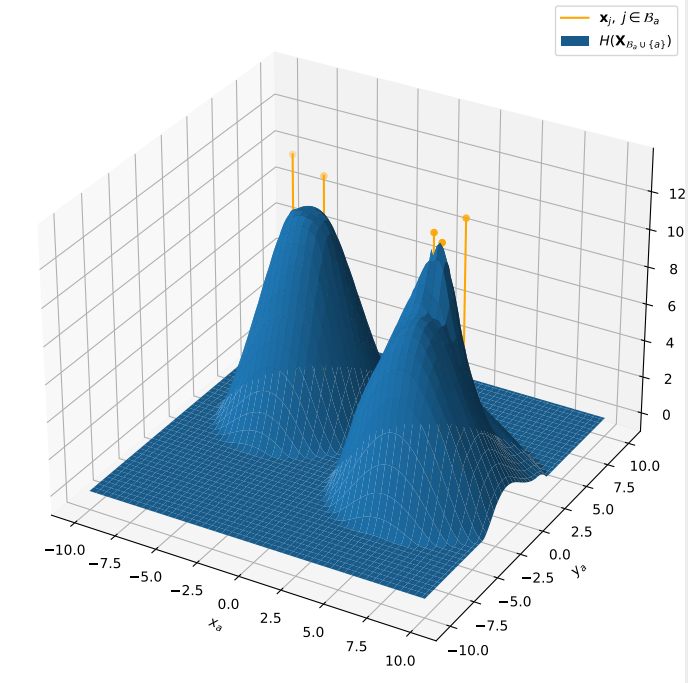


Figure 7: Objective function in (34a) for agent a with $|B_a| = 5$.

6 Implementation

6.1 Local probability

It is assumed that the agents have perfect communication capabilities within their maximum range. The local probability of an agent a wrt. a point $\mathbf{y}$ is defined as:

$$\hat{p}(\mathbf{x}_a, \mathbf{y}) = \begin{cases} 1, & \mathbf{y} \in V_a \\ 0, & \mathbf{y} \in U_a \end{cases} = 1_{\{\mathbf{y} \in V_a\}}, \tag{35}$$

where $1_{\{\cdot\}}$ is the indicator function, which is simply equal to one if the clause in the subscript is true and zero otherwise. This implies that the overall probability of coverage over the feasible space in (16) is simply the area of all intersections of three or more visible sets.

6.2 Computing the local probability of coverage

The neighbors of an agent a is partitioned into two sets:

$$\mathcal{B}_{aV} = \{j \in \mathcal{B}_a : \mathbf{y} \in V_j\} \tag{36a}$$

$$\mathcal{B}_{aU} = \{j \in \mathcal{B}_a : \mathbf{y} \in U_j\}. \tag{36b}$$

Thus for a given point $\mathbf{y}$, the set $\mathcal{B}_{aV}$ contains all neighbors of a whose visible set contains $\mathbf{y}$, and $\mathcal{B}_{aU}$ contains all neighbors of agent a whose visible set does not contain $\mathbf{y}$.

Now the local probability of coverage for agent a can be written as:

$$\begin{aligned} L(\mathbf{X}_{\mathcal{B}_a \cup \{a\}}) &= \int_{V_a} \Phi^2(\mathbf{X}_{\mathcal{B}_a}, \mathbf{y}) 1_{\{\mathbf{y} \in V_a\}} d\mathbf{y} = \int_{V_a} \Phi^2(\mathbf{X}_{\mathcal{B}_a}, \mathbf{y}) d\mathbf{y} \\ &= \int_{V_a} \sum_{n=0}^{2} \Phi^n(\mathbf{X}_{\mathcal{B}_{aV}}, \mathbf{y}) \Phi^{2-n}(\mathbf{X}_{\mathcal{B}_{aU}}, \mathbf{y}) d\mathbf{y} \\ &= \int_{V_a} \sum_{n=0}^{2} 1_{\{|B_{aV}|=n\}} 1_{\{2-n=0\}} d\mathbf{y} \\ &= \int_{V_a} 1_{\{|B_{aV}|=2\}} d\mathbf{y}. \end{aligned} \tag{37}$$

Thus the value of the local probability of coverage is equal to the area where the visible set of a overlaps with those of exactly two neighboring agents when assuming perfect communication within the entire visible set. This corresponds to the area of the blue region in Figure 5.

An alternative way of computing the value of the local probability of coverage in (37), used in the implementation, is presented below:

$$\begin{aligned} L(\mathbf{X}_{\mathcal{B}_a \cup \{a\}}) &= A\left(\bigcup_{\mathcal{A} \in \mathrm{Comb}(\mathcal{B}_a, 2)} \left[\left(V_a \cap \bigcap_{i \in \mathcal{A}} V_i \right) \setminus \left(\bigcup_{j \in \mathcal{B}_a \setminus \mathcal{A}} V_j \right) \right] \right) \\ &= A\left(V_a \cap \bigcup_{\mathcal{A} \in \mathrm{Comb}(\mathcal{B}_a, 2)} \left[\left(\bigcap_{i \in \mathcal{A}} V_i \right) \setminus \left(\bigcup_{j \in \mathcal{B}_a \setminus \mathcal{A}} V_j \right) \right] \right), \end{aligned} \tag{38}$$

where $A(\cdot)$ returns the area of its argument.

6.3 Optimizing swarm configuration

Running simulations is done by iteratively solving (34) for one agent at a time. Meaning at any time there is only a single agent computing its optimal position while all other agents are static. Once the agent currently computing its optimal position has done so, it moves to the optimum. Then the next agent does the same, and so on. This procedure is repeated until all agents are sufficiently close to a local optimum. The procedure is described in Algorithm 1.

Algorithm 1: Optimizing swarm configuration

Input: Size of swarm: N, Initial configuration of swarm: $\mathbf{X}_0$, Feasible space: $\mathcal{F}$, Tolerance: ϵ

$\mathbf{X}_\mathcal{N} \leftarrow \mathbf{X}_0$;
for $a \leftarrow 0$ **to** $N-1$ **do**
  $\mathbf{C}[a] \leftarrow$ False;
  $\mathbf{V}[a] \leftarrow$ Compute V_a using (6) with $\mathbf{x}_a = \mathbf{X}_\mathcal{N}[a]$;
end
while *not* $\mathbf{C}[a]$ *is* True *for all* $a = 0 \dots N-1$ **do**
  for $a \leftarrow 0$ **to** $N-1$ **do**
    $\mathbf{x}_{a,0} \leftarrow \mathbf{X}_\mathcal{N}[a]$;
    $\mathcal{B}_a \leftarrow$ Compute $\mathcal{B}_a$ using (20a) with $\mathbf{x}_a = \mathbf{x}_{a,0}$;
    $\mathbf{x}_a^* \leftarrow$ Solve (34) with initial guess $\mathbf{x}_{a,0}$ and neighbors $\mathcal{B}_a$;
    if $\|\mathbf{x}_a^* - \mathbf{x}_{a,0}\| \leq \epsilon$ **then**
      $\mathbf{C}[a] \leftarrow$ True
    end
    $\mathbf{V}[a] \leftarrow$ Compute V_a using (6) with $\mathbf{x}_a = \mathbf{x}_a^*$;
    $\mathbf{X}_\mathcal{N}[a] \leftarrow \mathbf{x}_a^*$;
  end
end
return $\mathbf{X}_\mathcal{N}$

6.4 Optimization solver

As (34) is a non-convex and constrained optimization problem it is solved using Sequential Quadratic Programming (SQP). In [20] a Nonlinear Program (NLP) is defined as:

$$\begin{aligned} \min_{\mathbf{x}\in\mathbb{R}^n} f(\mathbf{x}) \quad \text{s.t.} \quad & g_i(\mathbf{x}) = 0 \; i \in \{1 \dots m_e\} \\ & g_i(\mathbf{x}) \geq 0 \; i \in \{m_e + 1 \dots m\} \\ & \mathbf{x}^{lb} \leq \mathbf{x} \leq \mathbf{x}^{ub} \end{aligned} \tag{39}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ and $g : \mathbb{R}^n \to \mathbb{R}^m$ are assumed to be continuously differentiable. A SQP (Sequential Quadratic Programming) solver finds one, of possibly many, local minima of $f(\mathbf{x})$ within the feasible space defined by the constraints. The solution of (39) is found in an iterative manner. Given an initial guess, $\mathbf{x}_0$, of the solution, a SQP solver iteratively solves quadratic sub-problems, generating step size α_k and search direction $\mathbf{d}_k$. The solution to (39) is then updated as:

$$\mathbf{x}_{k+1} = \mathbf{x}_k + \alpha_k \mathbf{d}_k \tag{40}$$

Iteration is performed until some optimality condition is fulfilled [20].

Algorithm 1 is implemented in the Python programming language [21]. The Scipy [22] optimization library implements a wrapper for the SQP subroutine proposed in [20] which is used for solving (34).

6.5 Parameters

In all simulations the following parameter values are chosen

	Variable	Value
Maximum communication range	r	3
Minimum distance to neighbors	r_{min}	0.2
Minimum distance to line connecting neighbors	d_{min}	0.1
Convergence threshold	ϵ	10^{-2}
Dispersion decay	k_2	1
Dispersion gain	k_1	$\{0, 1\}$

6.6 Source code

The source code implementing Algorithm 1 and producing all plots shown in this report is accessible at [23].

7 Simulations

Simulations are done in four different mission spaces with properties summed up in Table 1.

Table 1: Mission space properties

Name	Size	Obstacles
Tinyworld	1.5-by-1.5 square	None
Tinyworld2	1.5-by-1.5 square	Central 0.5-by-0.5 square
Rectworld	10-by-10 square	None
Complexworld	15-by-25 rectangle	Horizontal wall, vertical wall and hexagon

For all mission spaces simulations are performed where the dispersion term is neglected ($k_1 = 0$), i.e. $H(\mathbf{X}_{\mathcal{B}_a \cup \{a\}}) = L(\mathbf{X}_{\mathcal{B}_a \cup \{a\}})$, and where the dispersion term is present ($k_1 = 1$). In all simulations the initial configuration of the swarm is a zigzag pattern starting in the bottom left corner of the mission space moving upwards. Agents are numbered from top to bottom, meaning the top-most agent is agent 0, and for a swarm of size N the bottom-most agent is agent $N - 1$. Thus the topmost agent will always be the first to optimize its position when running Algorithm 1.

Swarm configurations are plotted as follows: The boundary of visible set of an agent is plotted as a dashed line. All areas within the visible set of at least three agents is solid blue. Agents are visualized as orange dots.

In the step length plots, the step length trajectory of each agent is displayed as a uniquely coloured line. For simulations on large swarms including legends would cause the figures be unreasonably large. Due to this none of the step length plots include legends.

7.1 Tinyworld

The Tinyworld is constructed so that for three or more agents, the entire feasible space is covered regardless of the position of the agents.

Figure 8 shows the initial and final configuration of 3 agents when spawned in the Tinyworld environment, and no active dispersion is used ($k_1 = 0$). Figure 9 shows the covered area and step length per agent versus iteration count.

Results of optimizing agent positions with active dispersion ($k_1 = 1$, $k_2 = 1$) are shown in Figure 10 and Figure 11.

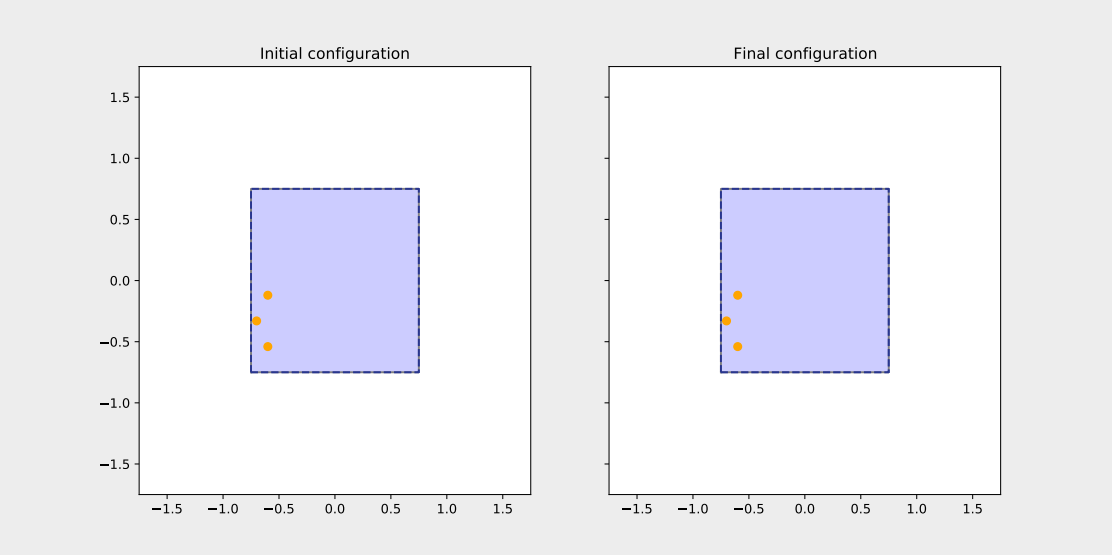
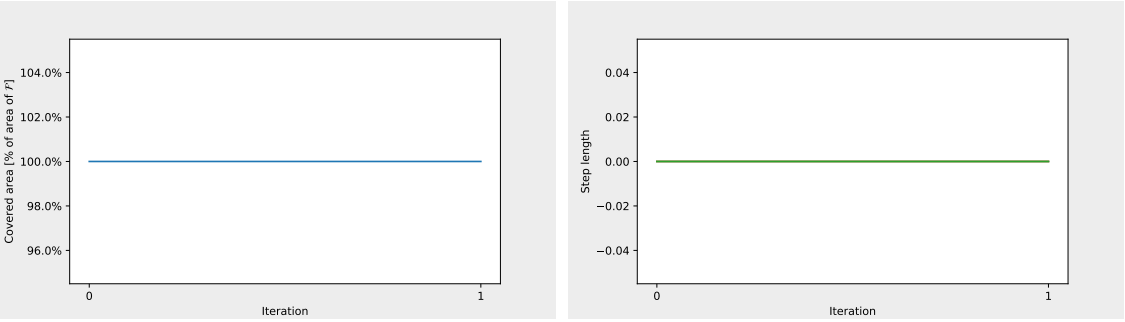


Figure 8: Initial and final configuration of 3 agents in the Tinyworld environment with $k_1 = 0$ (no active dispersion).



(a) Coverage evolution for 3 agents in the Tinyworld environment with $k_1 = 0$ (no active dispersion).

(b) Step length evolution for 3 agents in the Tinyworld environment with $k_1 = 0$ (no active dispersion).

Figure 9: Coverage percentage and step length evolution for 3 agents in the Tinyworld environment when no active dispersion is used.

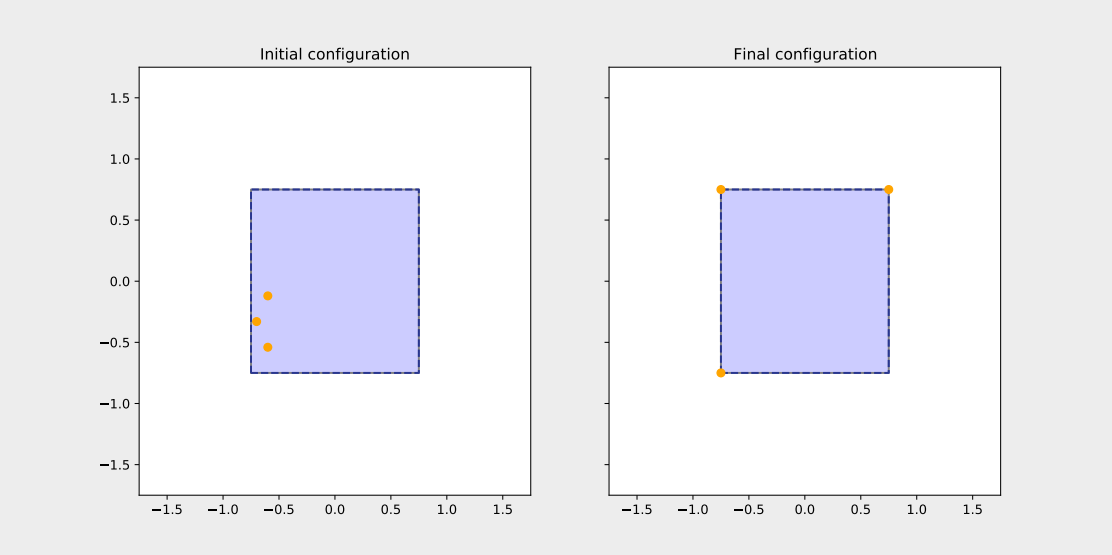
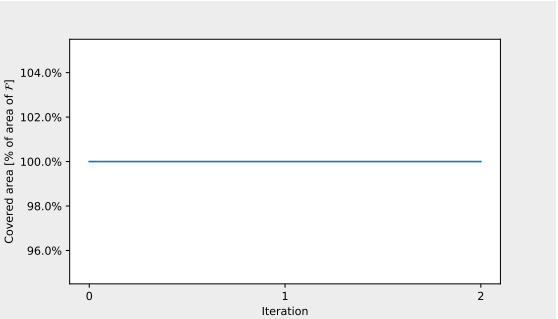
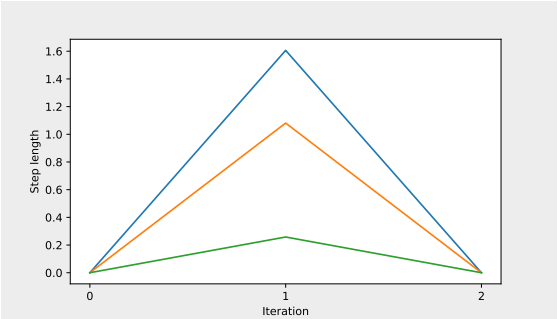


Figure 10: Initial and final configuration of 3 agents in the Tinyworld environment with $k_1 = k_2 = 1$ (active dispersion).



(a) Coverage evolution for 3 agents in the Tinyworld environment with $k_1 = k_2 = 1$ (active dispersion).



(b) Step length evolution for 3 agents in the Tinyworld environment with $k_1 = k_2 = 1$ (active dispersion).

Figure 11: Coverage percentage and step length evolution for 3 agents in the Tinyworld environment when active dispersion is used.

7.2 Tinyworld2

The Tinyworld2 environment is constructed to study how obstacles affect the configuration generated by Algorithm 1.

The results of a simulation for 3 agents without active dispersion ($k_1 = 0$) are shown in Figure 12 and Figure 13. Figure 16 and Figure 17 show the results of a simulation of 6 agents without active dispersion.

The results generated by Algorithm 1 when applying active dispersion ($k_1 = 1$, $k_1 = 2$) and running simulations for 3 and 6 agents are shown in Figure 14 and Figure 15, and Figure 18 and Figure 19 respectively.

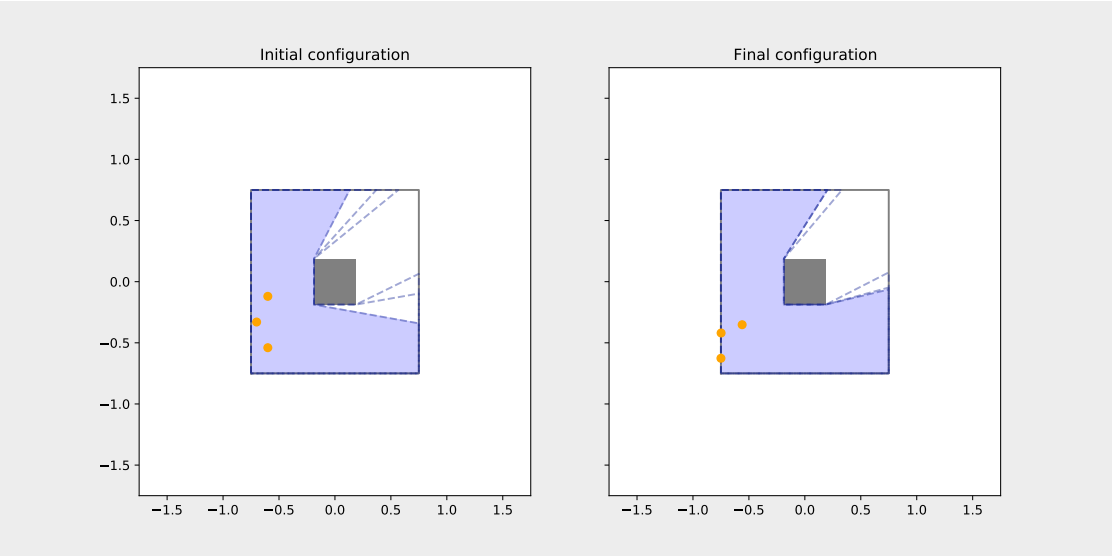
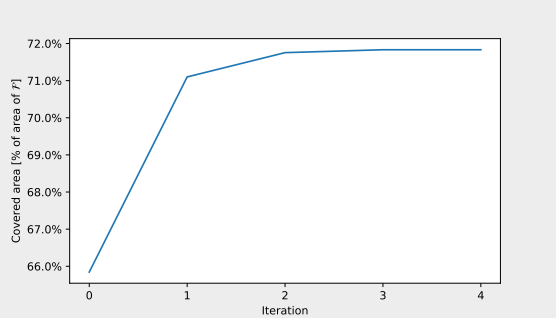
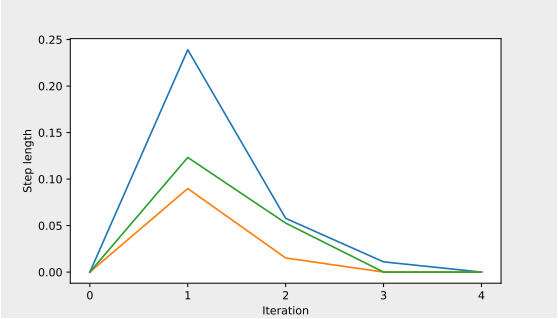


Figure 12: Initial and final configuration of 3 agents in the Tinyworld2 environment with $k_1 = 0$ (no active dispersion).



(a) Coverage evolution for 3 agents in the Tinyworld2 environment with $k_1 = 0$ (no active dispersion).



(b) Step length evolution for 3 agents in the Tinyworld2 environment with $k_1 = 0$ (no active dispersion).

Figure 13: Coverage percentage and step length evolution for 3 agents in the Tinyworld2 environment when no active dispersion is used.

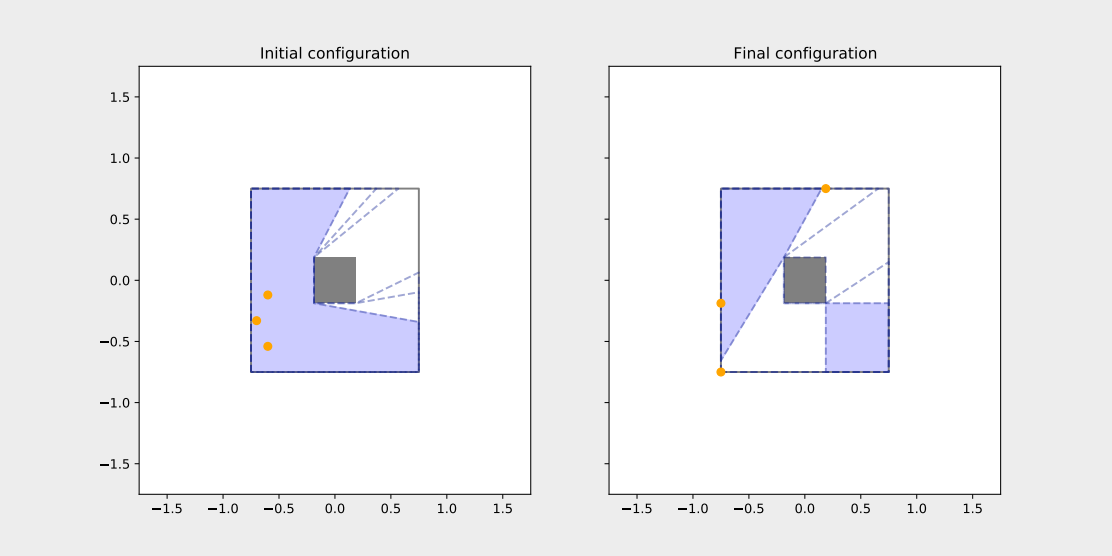
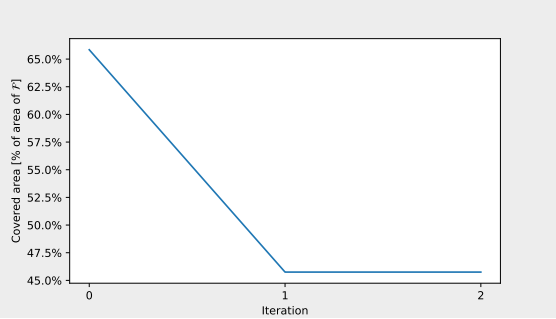
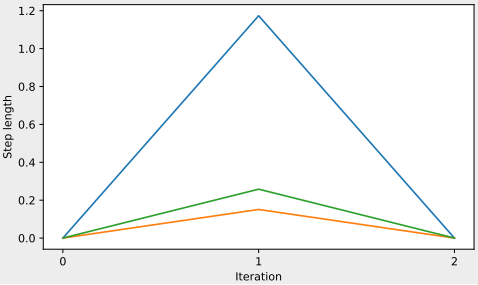


Figure 14: Initial and final configuration of 3 agents in the Tinyworld2 environment with $k_1 = k_2 = 1$ (active dispersion).



(a) Coverage evolution for 3 agents in the Tinyworld2 environment with $k_1 = k_2 = 1$ (active dispersion).



(b) Step length evolution for 3 agents in the Tinyworld2 environment with $k_1 = k_2 = 1$ (active dispersion).

Figure 15: Coverage percentage and step length evolution for 3 agents in the Tinyworld2 environment when active dispersion is used.

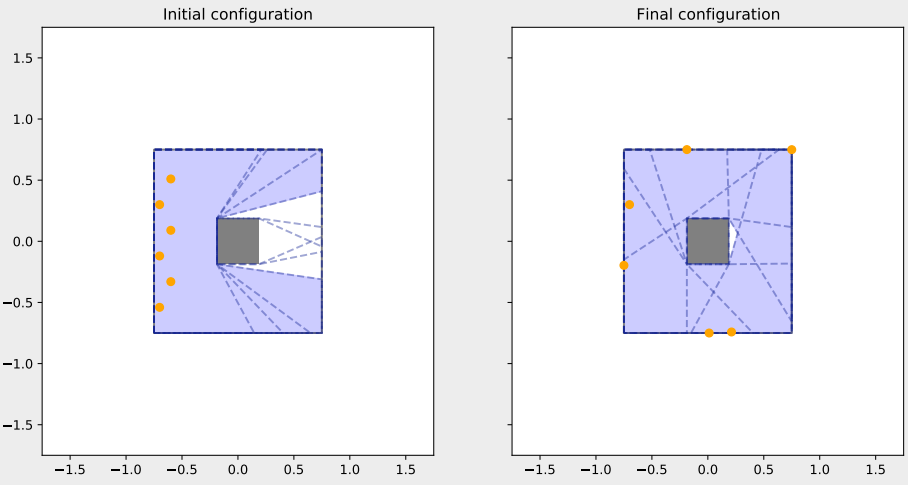
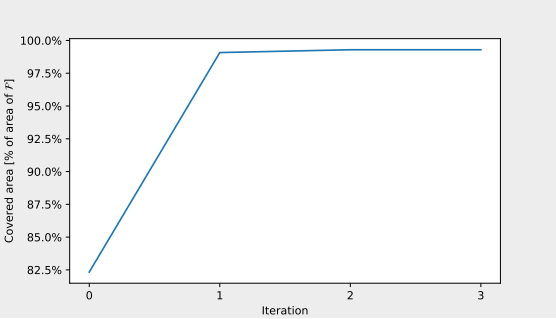
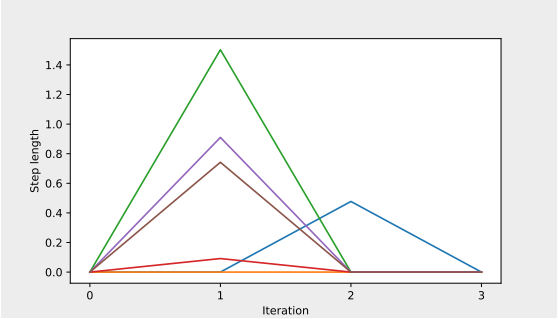


Figure 16: Initial and final configuration of 6 agents in the Tinyworld2 environment with $k_1 = 0$ (no active dispersion).



(a) Coverage evolution for 6 agents in the Tinyworld2 environment with $k_1 = 0$ (no active dispersion).



(b) Step length evolution for 6 agents in the Tinyworld2 environment with $k_1 = 0$ (no active dispersion).

Figure 17: Coverage percentage and step length evolution for 6 agents in the Tinyworld2 environment when active dispersion is not applied.

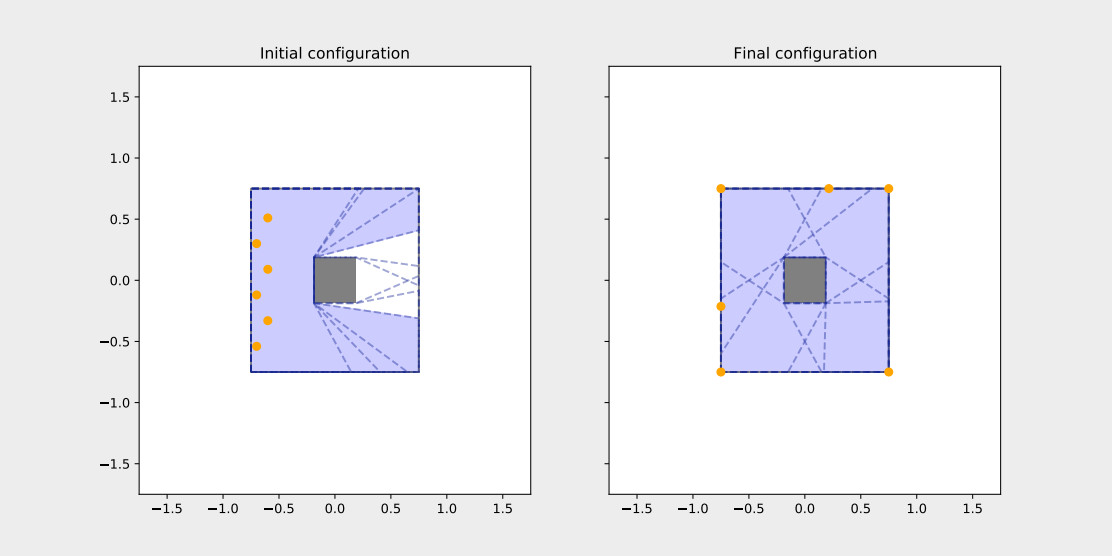
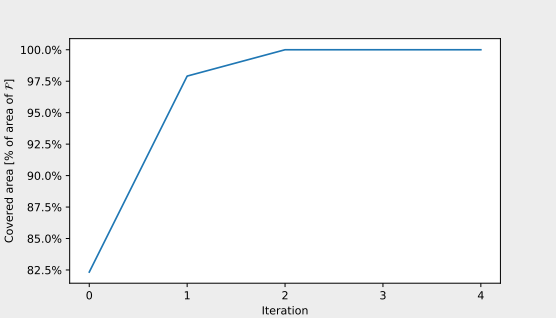
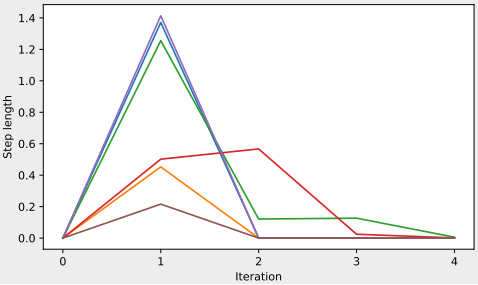


Figure 18: Initial and final configuration of 6 agents in the Tinyworld2 environment with $k_1 = k_2 = 1$ (active dispersion).



(a) Coverage evolution for 6 agents in the Tinyworld2 environment with $k_1 = k_1 = 1$ (active dispersion).



(b) Step length evolution for 6 agents in the Tinyworld2 environment with $k_1 = k_2 = 1$ (active dispersion).

Figure 19: Coverage percentage and step length evolution for 6 agents in the Tinyworld2 environment when active dispersion is applied.

7.3 Rectworld

The Rectworld is constructed to examine the behavior of Algorithm 1 for a larger mission space. Simulations are performed for swarms of 3, 6 and 15 agents. The results are displayed in Figures 20 to 31.

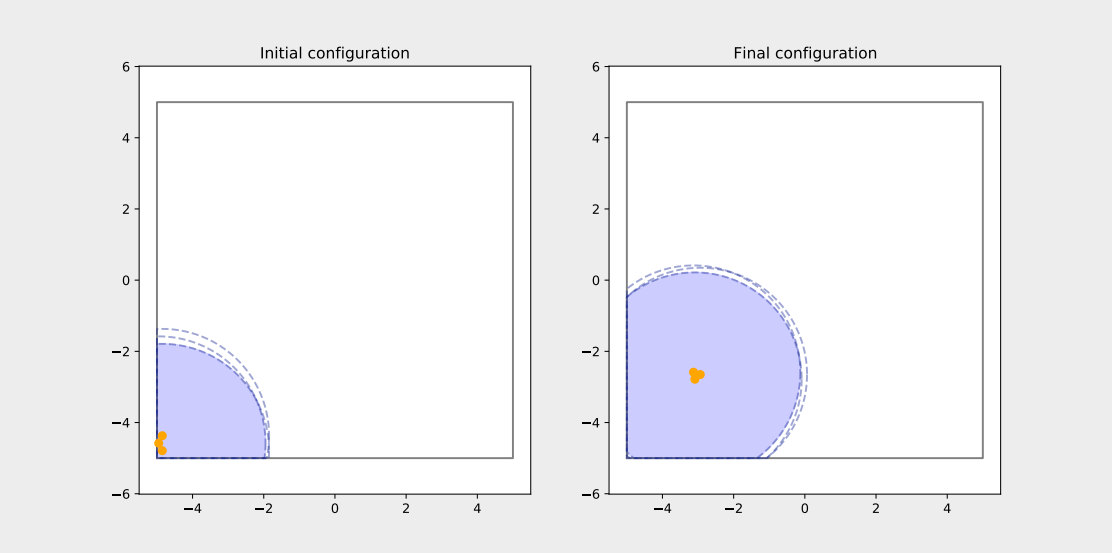
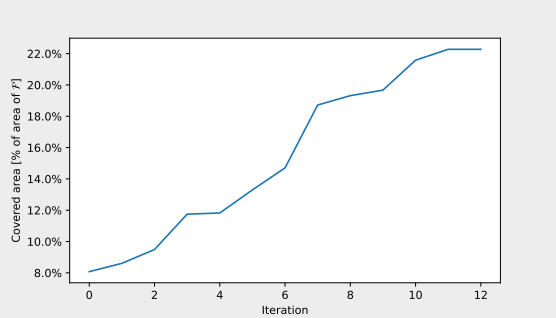
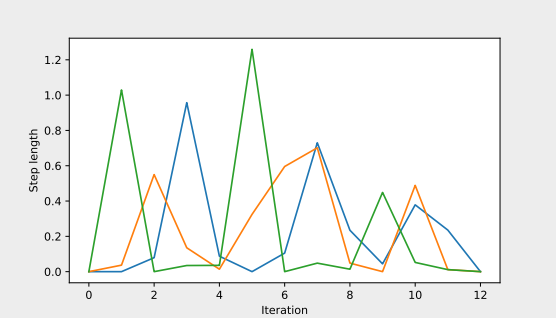


Figure 20: Initial and final configuration of 3 agents in the Rectworld environment with $k_1 = 0$ (no active dispersion).



(a) Coverage evolution for 3 agents in the Rectworld environment with $k_1 = 0$ (no active dispersion).



(b) Step length evolution for 3 agents in the Rectworld environment with $k_1 = 0$ (no active dispersion).

Figure 21: Coverage percentage and step length evolution for 3 agents in the Rectworld environment when no active dispersion is used.

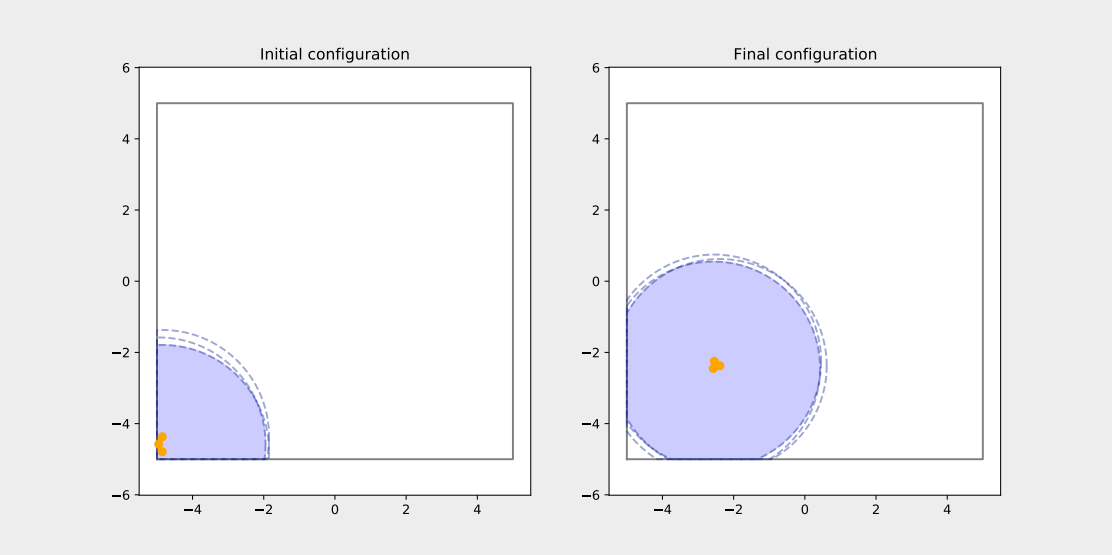
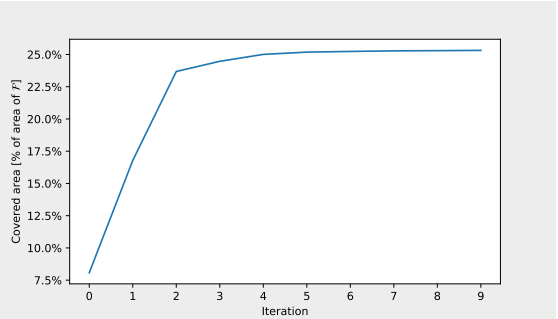
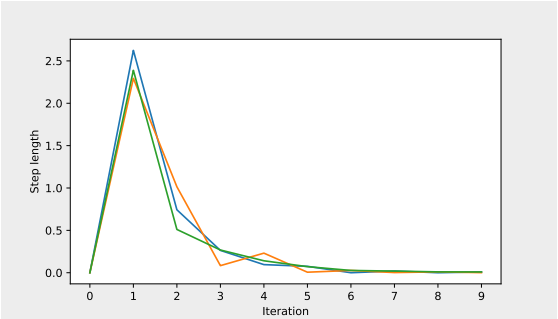


Figure 22: Initial and final configuration of 3 agents in the Rectworld environment with $k_1 = k_2 = 1$ (active dispersion).



(a) Coverage evolution for 3 agents in the Rectworld environment with $k_1 = k_1 = 1$ (active dispersion).



(b) Step length evolution for 3 agents in the Rectworld environment with $k_1 = k_1 = 1$ (active dispersion).

Figure 23: Coverage percentage and step length evolution for 3 agents in the Rectworld environment when active dispersion is used.

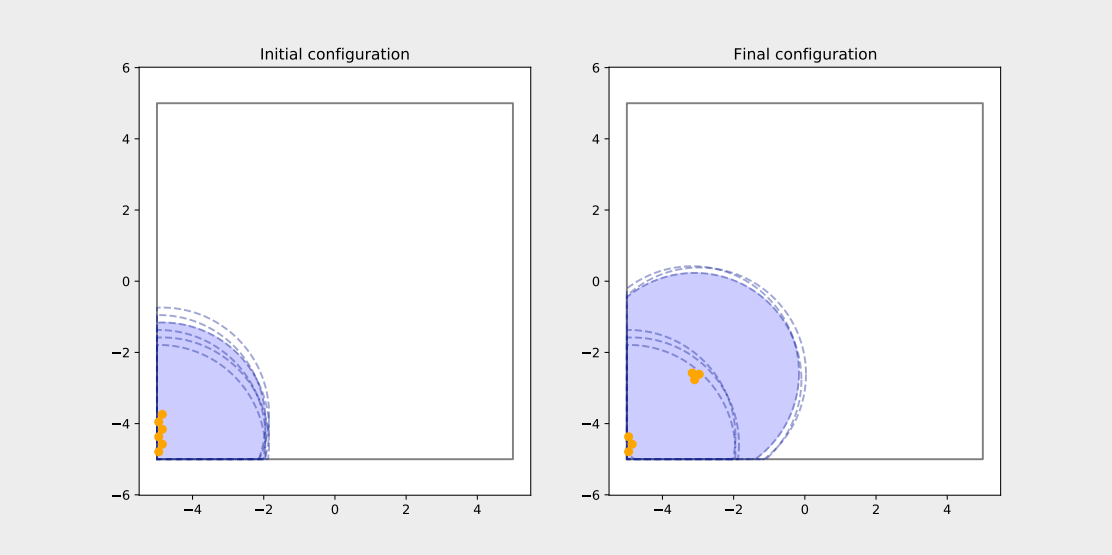
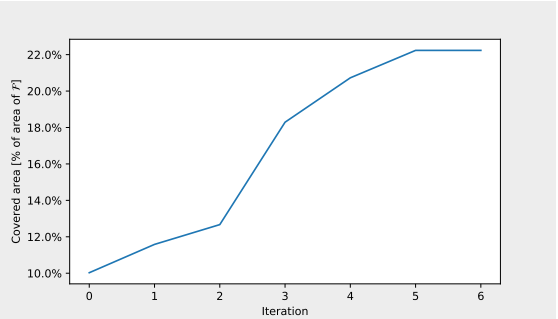
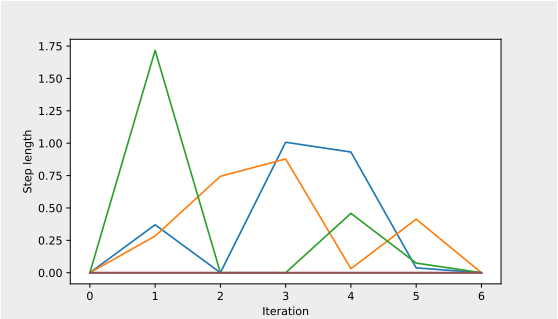


Figure 24: Initial and final configuration of 6 agents in the Rectworld environment with $k_1 = 0$ (no active dispersion).



(a) Coverage evolution for 6 agents in the Rectworld environment with $k_1 = 0$ (no active dispersion).



(b) Step length evolution for 6 agents in the Rectworld environment with $k_1 = 0$ (no active dispersion).

Figure 25: Coverage percentage and step length evolution for 6 agents in the Rectworld environment when no active dispersion is used.

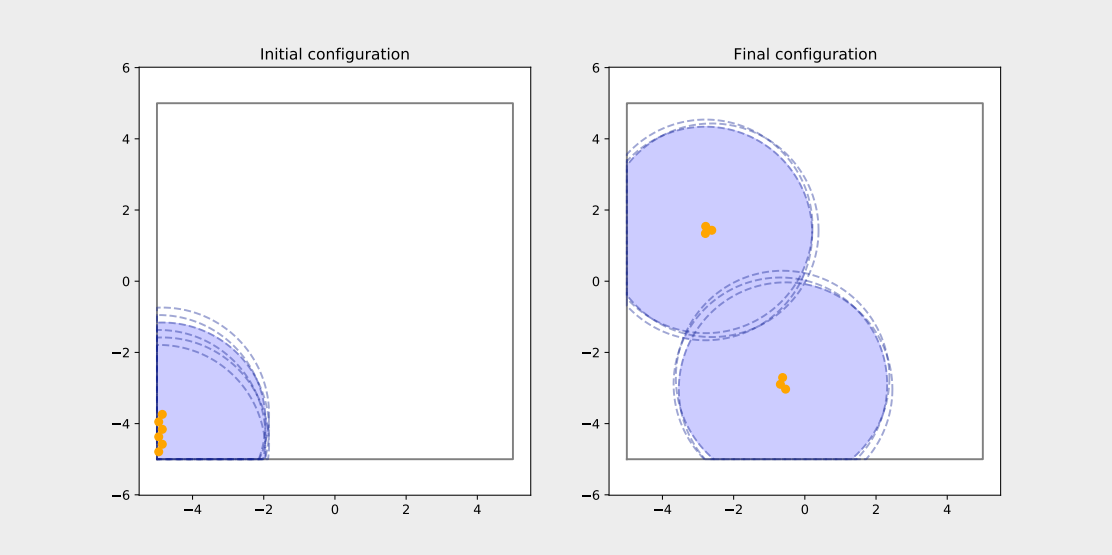
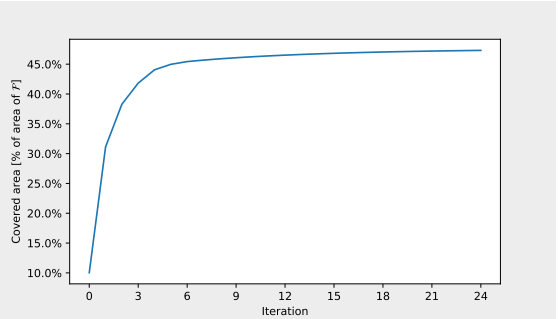
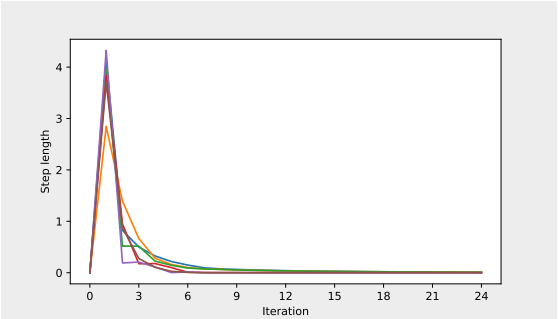


Figure 26: Initial and final configuration of 6 agents in the Rectworld environment with $k_1 = k_2 = 1$ (active dispersion).



(a) Coverage evolution for 6 agents in the Rectworld environment with $k_1 = k_1 = 1$ (active dispersion).



(b) Step length evolution for 6 agents in the Rectworld environment with $k_1 = k_1 = 1$ (active dispersion).

Figure 27: Coverage percentage and step length evolution for 6 agents in the Rectworld environment when active dispersion is used.

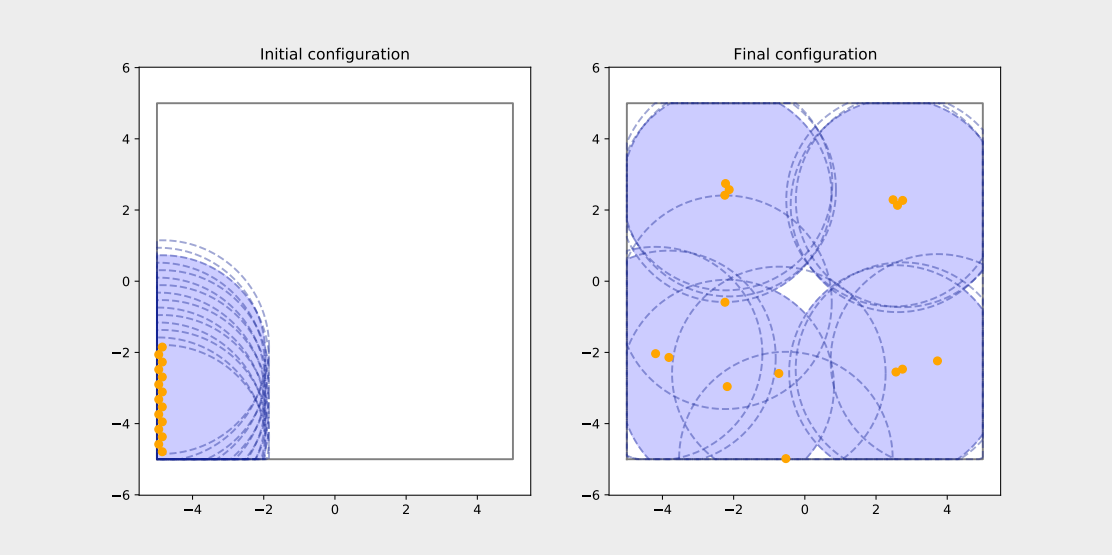
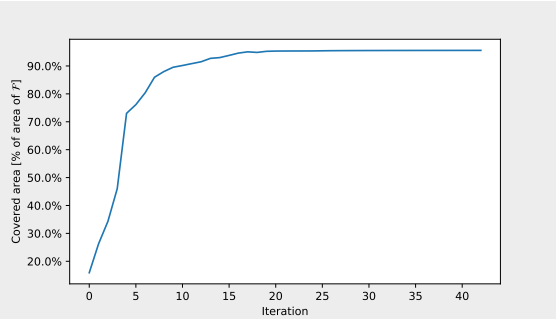
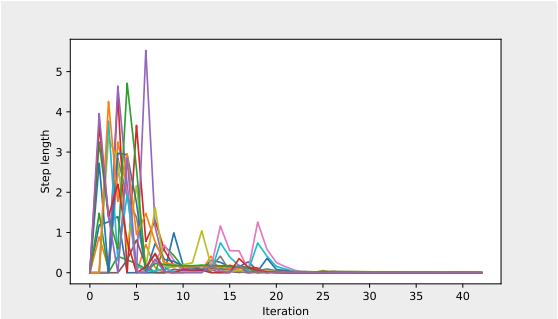


Figure 28: Initial and final configuration of 15 agents in the Rectworld environment with $k_1 = 0$ (no active dispersion).



(a) Coverage evolution for 15 agents in the Rectworld environment with $k_1 = 0$ (no active dispersion).



(b) Step length evolution for 15 agents in the Rectworld environment with $k_1 = 0$ (no active dispersion).

Figure 29: Coverage percentage and step length evolution for 15 agents in the Rectworld environment when no active dispersion is used.

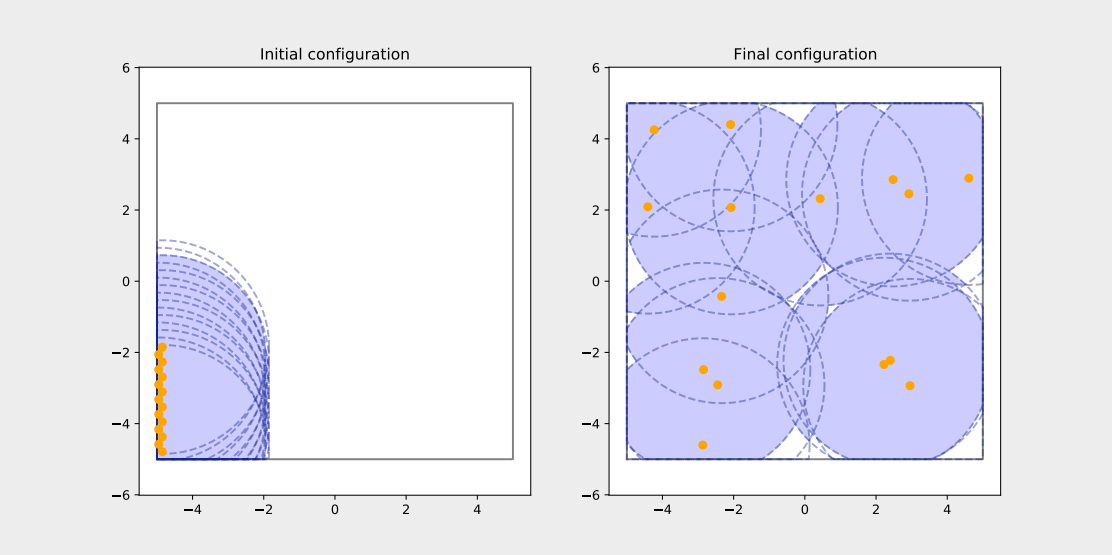
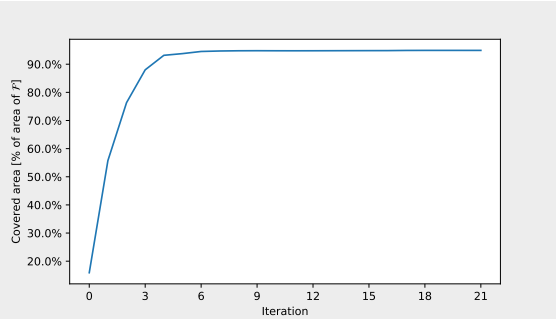
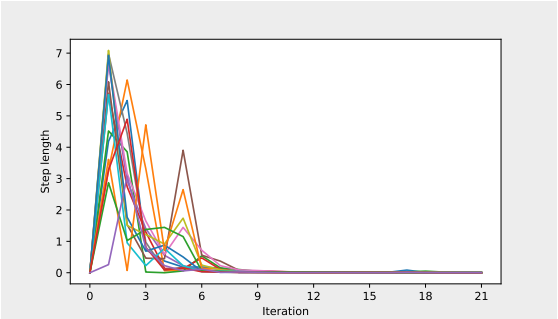


Figure 30: Initial and final configuration of 15 agents in the Rectworld environment with $k_1 = k_2 = 1$ (active dispersion).



(a) Coverage evolution for 15 agents in the Rectworld environment with $k_1 = k_1 = 1$ (active dispersion).



(b) Step length evolution for 15 agents in the Rectworld environment with $k_1 = k_1 = 1$ (active dispersion).

Figure 31: Coverage percentage and step length evolution for 15 agents in the Rectworld environment when active dispersion is used.

7.4 Complexworld

The Complexworld is constructed to evaluate the performance of Algorithm 1 for a large swarm in a larger and more demanding environment. Simulations are performed with and without active dispersion for a swarm of 50 agents. The results are shown in Figures 32 to 35.

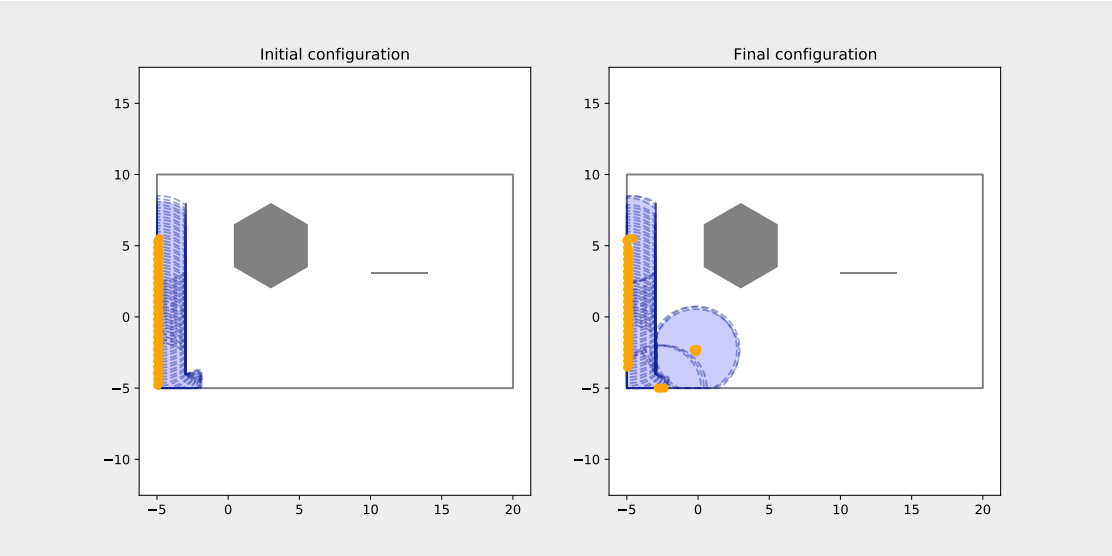
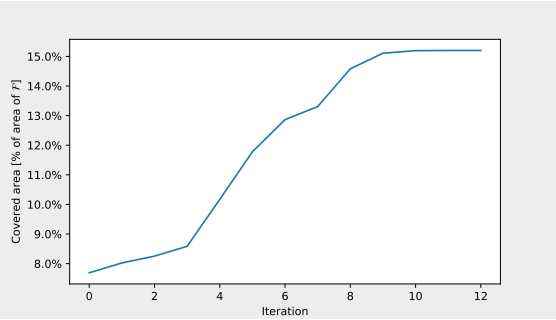
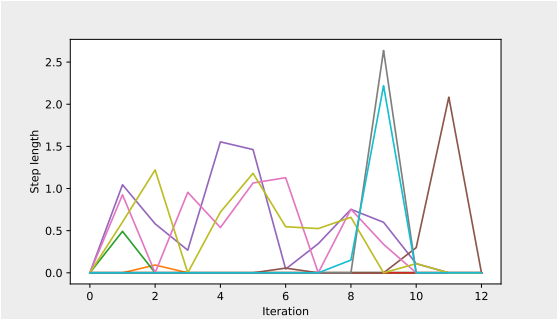


Figure 32: Initial and final configuration of 50 agents in the Complexworld environment with $k_1 = k_2 = 1$ (active dispersion).



(a) Coverage evolution for 50 agents in the Complexworld environment with $k_1 = k_2 = 1$ (active dispersion).



(b) Step length evolution for 50 agents in the Complexworld environment with $k_1 = k_2 = 1$ (active dispersion).

Figure 33: Coverage percentage and step length evolution for 50 agents in the Complexworld environment when active dispersion is used.

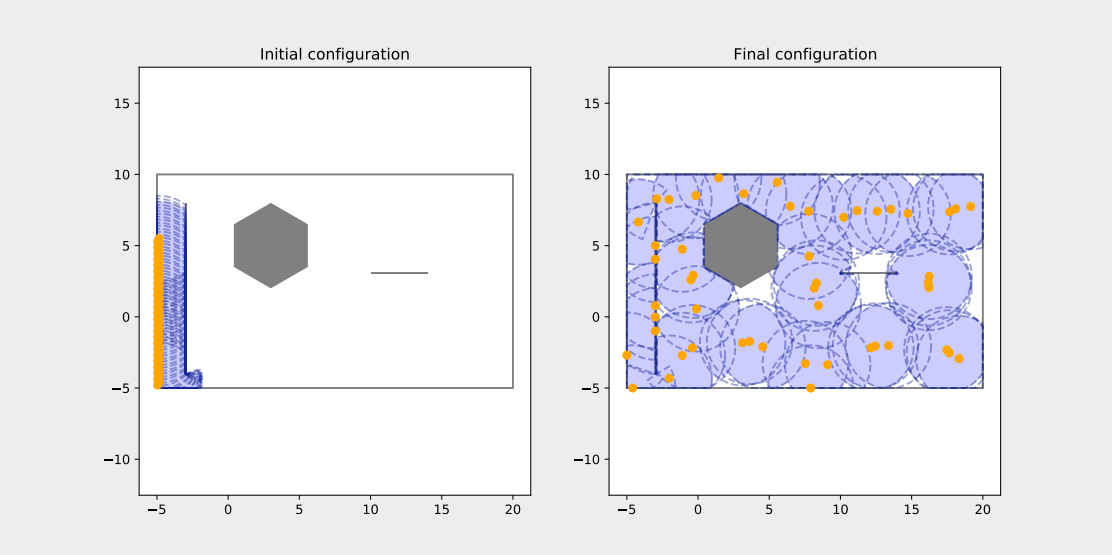
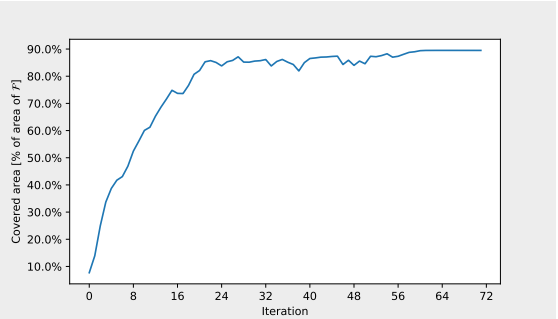
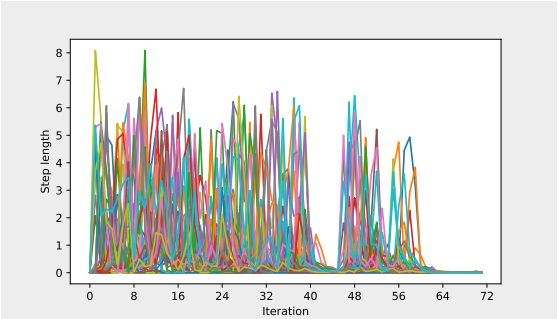


Figure 34: Initial and final configuration of 50 agents in the Complexworld environment with $k_1 = k_2 = 1$ (active dispersion).



(a) Coverage evolution for 50 agents in the Complexworld environment with $k_1 = k_2 = 1$ (active dispersion).



(b) Step length evolution for 50 agents in the Complexworld environment with $k_1 = k_2 = 1$ (active dispersion).

Figure 35: Coverage percentage and step length evolution for 50 agents in the Complexworld environment when active dispersion is used.

8 Discussion

The results obtained in Section 7 are summarized in Table 2 and Table 3.

Table 2: Simulation results for all environments without active dispersion.

Environment	3 agents		6 agents		15 agents		50 agents	
Coverage	Initial	Final	Initial	Final	Initial	Final	Initial	Final
Tinyworld	100%	100%	-	-	-	-	-	-
Tinyworld2	65.9%	71.8%	82.5%	98%	-	-	-	-
Rectworld	8%	22.1%	10%	25.5%	15%	95%	-	-
Complexworld	-	-	-	-	-	-	7.5%	15.5%

Table 3: Simulation results for all environments with active dispersion.

Environment	3 agents		6 agents		15 agents		50 agents	
Coverage	Initial	Final	Initial	Final	Initial	Final	Initial	Final
Tinyworld	100%	100%	-	-	-	-	-	-
Tinyworld2	65.9%	45.2%	82.5%	100%	-	-	-	-
Rectworld	8%	25%	10%	46%	15%	93%	-	-
Complexworld	-	-	-	-	-	-	7.5%	89%

8.1 Coverage percentage of feasible space

As the results in Table 2 show, running Algorithm 1 without active dispersion yields configurations where the final coverage is always equal to or better than the initial coverage in terms of coverage percentage. The only environment in which the coverage is not approved upon is the Tinyworld environment. This is due to the Tinyworld environment being constructed so that, in any configuration, 3 or more agents cover the entire feasible space.

When applying active dispersion Algorithm 1 returns configurations where the final coverage percentage is higher than the initial coverage percentage in all environments except for the Tinyworld2 environment. The reason for the diminishing percentage of coverage in the Tinyworld2 environment is discussed in Section 8.4.

8.2 Clustering

In all simulations, agents tend to gather in groups of three, and cluster as closely together as the constraints (34d)-(34e) allow. This behavior is especially prevalent when there are not enough agents to cover the entire feasible space, and when active dispersion is not applied. This a consequence of the area of the intersection of three circles being largest when the circle centers coincide. Considering the results in [15] such behavior is disadvantageous. The region in which the accuracy of multilateration (in presence of measurement noise) is the greatest lies in and around the convex hull of the agents. Tight clustering yields a small convex hull, meaning the region in which accurate multilateration can be performed is small.

8.3 Active dispersion as cluster prevention

In the Tinyworld environment, whose simulation results are shown in Section 7.1, the effect of active dispersion is evident (Figure 8 and Figure 10). In the case where no active dispersion is used the objective function is constant over the feasible space. Hence the initial configuration is one of infinitely many equal valued optima, and Algorithm 1 halts after one iteration. In the case where active dispersion is applied the agents spread to the corners of the feasible space as seen in Figure 10. This is due to the fact that the local probability of coverage is constant over the feasible space for all agents, hence the local optimization problem (34) is equivalent to maximizing dispersion between an agent a and its neighbors.

Simulations performed in the Rectworld environment for 15 agents exhibit some of the same behavior. The final configuration obtained when active dispersion is applied is shown in Figure 30. Comparing this to the configuration obtained in the same environment with the same number of agents (see Figure 28), it is clear that the agents are more evenly spread out in the mission space when active dispersion is used. In densely populated areas the gradient of the local probability of coverage tends to be small. This is due to the fact that in densely populated areas, the majority of the area is covered independently of an agent a covering it or not. Thus any small perturbation of an agent's position does not have a major impact on the agent's local probability of coverage. As the number of neighbors for an agent in a densely populated area is large, the dispersion term is significantly negative and has a steep gradient. This combination of movement having a small impact on the local probability of coverage and a large impact on the value of the dispersion term causes agents to move away from densely populated areas, preventing clustering. The coverage percentage is however two percent smaller when using active dispersion for a swarm of 15 agents in the Rectworld area than when only optimizing the local probability of coverage.

Active dispersion does however not prevent clustering in sparsely populated areas. This can be seen by the configurations for 3 and 6 agents in the Rectworld environment (Figure 22 and Figure 26). In these configurations, any movement from the current configuration would cause the local probability of coverage for the agent that moves to drop significantly. Furthermore, none of the agents have enough neighbors to force them to move despite the drop in local probability of coverage.

8.4 Effect of improperly weighted active dispersion

The simulations performed for a swarm of size 3 in the Tinyworld2 environment display a weakness of active dispersion. Here the swarm covers a significantly smaller percentage of the feasible space when applying active dispersion than when not. This is due to the fact that the dispersion term is weighted too heavily. When the local probability of coverage for an agent is small, i.e. $L(\mathbf{X}_{\mathcal{B}_a \cup \{a\}})$ is small for all possible values of $\mathbf{x}_a$, the value of the local objective (34a) is dominated by the dispersion term. This causes a shift in behavior. The focus is on dispersion rather than coverage, and Algorithm 1 converges to a solution that gives a smaller coverage percentage.

For a swarm of 6 agents in the Tinyworld2 environment, the case is quite different. Now active dispersion yields a configuration that gives a larger coverage percentage than when no active dispersion is applied. This result is although not a consequence of clever objective function formulation. As for three agents, the active dispersion term dominates the coverage term in (34a) due to the local probability of coverage being small no matter the value of $\mathbf{x}_a$. Hence the agents focus on maximizing dispersion rather than covered area. The solution generated by Algorithm 1 for 6 agents in the Tinyworld2 environment is identical to the solution generated for 6

agents in the Tinyworld environment when applying active dispersion. This makes it obvious that the high percentage of coverage obtained in the Tinyworld2 environment using active dispersion is not caused by the agents meticulously positioning themselves in order to maximize the local probability of coverage. Instead, they position themselves so that the maximum dispersion is obtained, and the higher percentage of coverage is obtained coincidentally.

8.5 Active dispersion forcing exploration

Figure 24 shows the configuration obtained by running Algorithm 1 for a swarm of size 6 in the Rectworld environment without active dispersion. The three agents in the bottom-left corner stay at their initial positions throughout the simulation even though the bottom left corner is covered by all 6 agents. As the minimum requirement for coverage is three agents, the swarm could clearly be utilized to give a higher percentage of coverage. Figure 36 shows the configuration of the swarm after each iteration of Algorithm 1. It is clear that throughout the iterations the visible sets of the bottom three agents are covered by those of the top three agents. Thus the local objective value for the bottom three agents is zero throughout the iterations, and any infinitesimal perturbation of their positions causes no change to the objective value, i.e. zero gradients. The SQP solver concludes that the bottom three agents are at local optima at each iteration, and no change is made to their position.

The configuration generated for a swarm of size 6 using active dispersion in the Rectworld environment (see Figure 26) does not exhibit the same behavior as the aforementioned configuration. This can be attributed to the larger initial step sizes due to active dispersion (see Figure 25b and Figure 27b) causing no agents to have their entire visible set covered by those of three or more other agents. Thus no agent is left with a constant-zero local probability of coverage, and the swarm reaches a higher percentage of coverage.

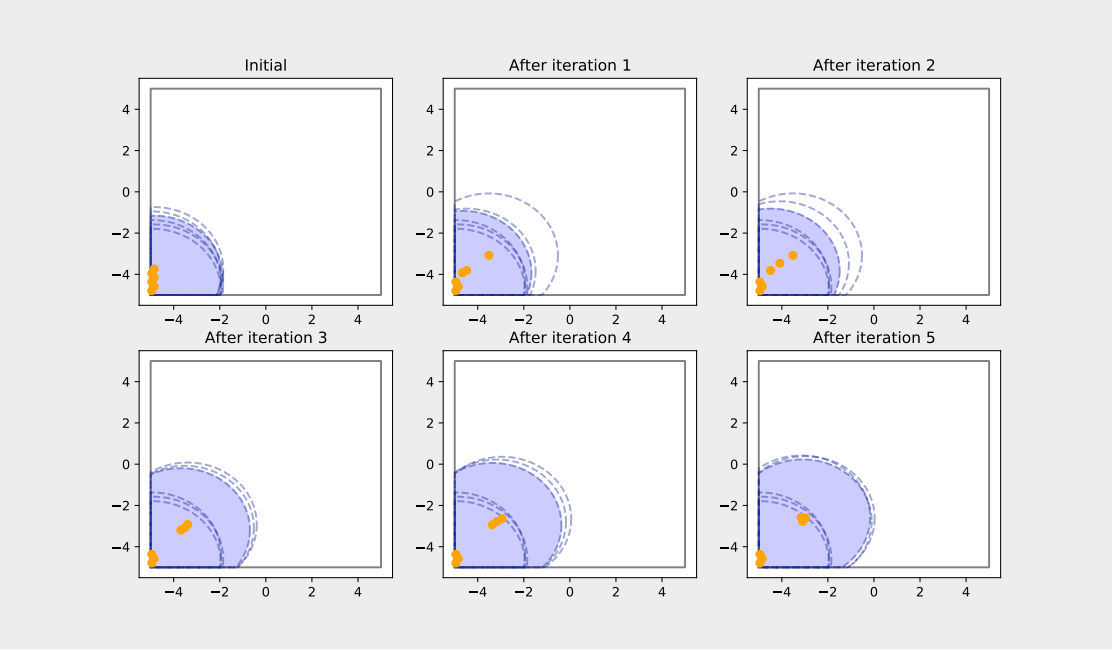


Figure 36: Intermediate configurations for 6 agents in the Rectworld environment with $k_1 = 0$ (no active dispersion).

Simulations performed in the Complexworld environment yet again show how active dispersion forces agents to explore the mission space. Without active dispersion Algorithm 1 converges quickly to a clearly sub-optimal configuration as seen in Figure 32. Only a small portion of the swarm makes advancements into the mission space. Yet again this is caused by the majority of agents having a zero-gradient local objective in every iteration of Algorithm 1 due to their visible sets being fully contained by those of at least three other agents. With active dispersion applied agents are explicitly encouraged to explore. Thus, as seen in Figure 34, agents disperse into the mission space and reach a far higher percentage of coverage than when no active dispersion is applied.

8.6 Local convergence causing sub-optimal configurations

For a swarm of 3 agents in the Rectworld environment, the final configurations obtained with and without active dispersion are shown in Figure 22 and Figure 20 respectively. The two configurations do not differ much, except that the configuration obtained when using active dispersion is translated slightly to the north-east. As shown in Figure 21b and Figure 23b the agents make larger initial steps when active dispersion is applied. This causes them to move further into the mission space. Thus less of their visual sets are clipped by the walls of the mission space, resulting in a larger percentage of covered area.

Both configurations obtained for 3 agents in the Rectworld environment exhibit the same problematic behavior. Moving the entire swarm in the north-east direction would yield a higher percentage of covered area, but Algorithm 1 converges with the swarm placed so that the area covered by the three agents is clipped by the mission space walls. This is due to Algorithm

1 optimizing the configuration of the swarm one agent at a time. In order to move the entire swarm to the north-east direction, all agents would have to move in the north-east direction, and they would have to do so one at a time. In the configuration shown in Figure 20 no single agent would increase their local objective by moving, although they would all benefit from each other moving. Due to no single agent benefiting from moving, Algorithm 1 converges to a clearly sub-optimal configuration.

Figure 26 shows the final configuration obtained for a swarm of 6 agents in the Rectworld environment using active dispersion. Again translating the entire swarm north-east would give a higher percentage of coverage, but due to all agents being at local optima of their local objective functions, Algorithm 1 halts, and yields a clearly sub-optimal configuration.

9 Future Work

Ensuring connectivity The deployed network is of little use if positional information cannot be propagated to someone outside the mission space. Due to this, effort should be put into ensuring that, in the final configuration, the network created by the agents is connected, and that at least one agent is connected with a base station at the mission space "entrance".

In reality, assuming obstacles are opaque, agents can only communicate with each other if they are within the visible sets of each other. The set of neighbors is defined so that two agents are neighbors if the distance between them is less than or equal to two times the communication range. Thus in reality, if the graph of agents is not connected, it might not be possible for an agent to determine its neighbors as there might not be any way for the information to propagate to an agent's neighbors.

Proper weighting of the active dispersion term Through simulations in the Tinyworld2 environment, it is clear that the proposed objective function (26) is prone to a poorly weighted dispersion term. The dispersion gain should not be constant, as the agents shift behavior when the local probability of coverage (23) is small. Due to this, the possibility of letting the dispersion gain vary with the local probability of coverage should be considered.

A two-phase approach could also be investigated. Phase one could cause agents to spread throughout the mission space (possibly using the potential fields approach in [11]) while ensuring connectivity of the graph spanned by the agents. In phase two agents could optimize some measure of multilateration cover (and multilateration quality).

The possibility of adapting the boosting function approach presented in [10] could also be examined. Using boosting functions there might not be a need for an explicit dispersion term, and the problem of improper weighting would be eliminated.

Prevent tight clustering Throughout the simulations, it became evident that the agents tend to form tight clusters of three agents. As multilateration yields the most accurate results for entities positioned inside or around the convex hull of agents, such behavior is undesirable. Thus maximizing the joint probability of three or more agents covering the area might not be the optimal approach when the overall goal is to deploy a network for *accurate* multilateration. Due to this more research should be made into optimal geometries between beacons used for multilateration, and alternative objective functions might need to be considered.

Avoid visible set being clipped by mission space boundary In the Rectworld environment, all final configurations suffered from the same problem: the visible sets were clipped by the mission space walls, and coordinated movements by local swarms would yield a more optimal configuration. Some coordinated moves for local swarms should be allowed to prevent the convergence to clearly sub-optimal configurations. Letting the feasible space boundaries exert some virtual force (as in [11]) could be applied to move agents away from the boundaries.

Concurrent simulation In reality, making a single agent perform optimization at any given time would demand a lot of coordination between agents, as it would have to be communicated and agreed upon which agent should perform optimization. Due to this, the possibility of letting multiple agents move at the same time should be investigated.

10 Conclusion

In this project, a novel distributed probability-based coverage optimization algorithm is proposed. The objective of which is to, without centralized control, form a network of agents who together can be used as beacons in a multilateration scheme allowing for accurate localization of entities entering the environment in which the agents are located.

The distributed objective function consists of a measure of local probability of coverage and a potential field-based measure of dispersion. The dispersion term is added so that agents are induced to spread from other agents. The algorithm is tested in four environments of different sizes and complexity. In each environment swarms of different sizes are deployed, initiated in a dense configuration, and the algorithm is applied with and without actively dispersing agents.

Simulations show that when the proposed algorithm is run without actively dispersing agents, the swarm of agents is prone to converging to configurations in which only a small part of the swarm explores the environment, leading to clearly sub-optimal coverage. This is due to the *local* probability of coverage being a constant function of an agents position in regions covered by four or more agents, leading to zero step size when performing local optimization. Applying active dispersion mitigates the problem of agents not exploring the environment. This, in some cases, results in far better coverage as the swarm is able to escape the initial dense configuration. However, in environments of smaller sizes simulations show that actively dispersing agents can overshadow the importance of covering the area, thus leading to sub-optimal configurations in which maximum dispersion is obtained rather than maximum coverage.

The results presented in this report show that in all tested environments, for all swarm sizes, the proposed algorithm yields a configuration in which the covered area is greater than or equal to the initially covered area when only optimizing the local probability of coverage. Furthermore, it is shown that in some cases adding a dispersion term inducing agents to explore the environment can drastically improve the coverage supplied by the swarm of agents. However, seen as the algorithm lacks robustness it can not yet be utilized in real-world situations. Work on the deployment problem will continue in an upcoming master's thesis. There, the problems presented in this report will be addressed, and hopefully, result in a solution that can be utilized during search-and-rescue missions.

References

[1] M.-W. Dictionary, "First responder," accessed 19 Nov. 2020. [Online]. Available: https://www.merriam-webster.com/dictionary/first%20responder

[2] M. Elbanhawi, A. Mohamed, R. Clothier, J. Palmer, M. Simic, and S. Watkins, "Enabling technologies for autonomous mav operations," *Progress in Aerospace Sciences*, vol. 91, pp. 27 – 52, 2017. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0376042116300367

[3] M. Gavhale and P. D. Saraf, "Survey on algorithms for efficient cluster formation and cluster head selection in manet," *Procedia Computer Science*, vol. 78, pp. 477 – 482, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S1877050916000934

[4] D. Anderson, "An evaluation of current and future costs for lithium-ion batteries for use in electrified vehicle powertrains," Master's project, Duke University, p. 82, 2009. [Online]. Available: https://hdl.handle.net/10161/1007

[5] J. F. M. Oudenhoven, R. J. M. Vullers, and R. van Schaijk, "A review of the present situation and future developments of micro-batteries for wireless autonomous sensor systems," *International Journal of Energy Research*, vol. 36, no. 12, pp. 1139–1150, 2012. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/er.2949

[6] C. Forster, M. Pizzoli, and D. Scaramuzza, "Svo: Fast semi-direct monocular visual odometry," in *2014 IEEE International Conference on Robotics and Automation (ICRA)*, 2014, pp. 15–22.

[7] M. Faessler, F. Fontana, C. Forster, E. Mueggler, M. Pizzoli, and D. Scaramuzza, "Autonomous, vision-based flight and live dense 3d mapping with a quadrotor micro aerial vehicle," *Journal of Field Robotics*, vol. 33, no. 4, pp. 431–450, 2016. [Online]. Available: https://onlinelibrary.wiley.com/doi/abs/10.1002/rob.21581

[8] L. Bayndr, "A review of swarm robotics tasks," *Neurocomputing*, vol. 172, pp. 292 – 321, 2016. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0925231215010486

[9] M. Zhong and C. G. Cassandras, "Distributed coverage control and data collection with mobile sensor networks," in *49th IEEE Conference on Decision and Control (CDC)*, 2010, pp. 5604–5609.

[10] X. Sun, C. G. Cassandras, and K. Gokbayrak, "Escaping local optima in a class of multi-agent distributed optimization problems: A boosting function approach," 2014.

[11] A. Howard, M. J. Matarić, and G. S. Sukhatme, "Mobile sensor network deployment using potential fields: A distributed, scalable solution to the area coverage problem," in *Distributed Autonomous Robotic Systems 5*, H. Asama, T. Arai, T. Fukuda, and T. Hasegawa, Eds. Tokyo: Springer Japan, 2002, pp. 299–308.

[12] C. Wolff, "Multilateration," last visited on 13.11.2020. [Online]. Available: https://www.radartutorial.eu/02.basics/rp52.en.html

[13] A. Savvides, C.-C. Han, and M. B. Strivastava, "Dynamic fine-grained localization in ad-hoc networks of sensors," in *Proceedings of the 7th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '01. New York,

NY, USA: Association for Computing Machinery, 2001, p. 166179. [Online]. Available: https://doi.org/10.1145/381677.381693

[14] F. Domingo-Perez, J. Lzaro, A. Wieser, E. Gorostiza, D. Salido-Monz, and . De-La-Llana-Calvo, "Sensor placement determination for range-difference positioning using evolutionary multi-objective optimization," *Expert Systems with Applications*, vol. 47, pp. 95–105, 11 2015.

[15] R. Kaune, "Accuracy studies for tdoa and toa localization," in *2012 15th International Conference on Information Fusion*, 2012, pp. 408–415.

[16] E. W. Weisstein, "Simple polygon. From MathWorld–A Wolfram Web Resource," last visited on 10/11/2020. [Online]. Available: https://mathworld.wolfram.com/SimplePolygon.html

[17] Y. H. Wang, "On the number of successes in independent trials," *Statistica Sinica*, vol. 3, no. 2, pp. 295–312, 1993. [Online]. Available: http://www.jstor.org/stable/24304959

[18] C. Perwass, "Geometric algebra with applications in engineering," vol. 4, 01 2009.

[19] J. Nocedal and S. J. Wright, *Numerical Optimization*, 2nd ed. New York, NY, USA: Springer, 2006.

[20] D. Kraft, *A software package for sequential quadratic programming*, ser. Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt Köln: Forschungsbericht. Wiss. Berichtswesen d. DFVLR, 1988. [Online]. Available: https://books.google.no/books?id=4rKaGwAACAAJ

[21] G. Van Rossum and F. L. Drake, *Python 3 Reference Manual*. Scotts Valley, CA: CreateSpace, 2009.

[22] P. Virtanen, R. Gommers, T. E. Oliphant, M. Haberland, T. Reddy, D. Cournapeau, E. Burovski, P. Peterson, W. Weckesser, J. Bright, S. J. van der Walt, M. Brett, J. Wilson, K. J. Millman, N. Mayorov, A. R. J. Nelson, E. Jones, R. Kern, E. Larson, C. J. Carey, İ. Polat, Y. Feng, E. W. Moore, J. VanderPlas, D. Laxalde, J. Perktold, R. Cimrman, I. Henriksen, E. A. Quintero, C. R. Harris, A. M. Archibald, A. H. Ribeiro, F. Pedregosa, P. van Mulbregt, and SciPy 1.0 Contributors, "SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python," *Nature Methods*, vol. 17, pp. 261–272, 2020.

[23] M. Berdal, "Prosjekoppgave." [Online]. Available: https://github.com/mBerdal/ProsjektOppgave

Appendices

A Derivation of the probability of coverage on distributed form

Using (13) the probability of coverage (14) is expanded:

$$\Phi^{3^+}(\mathbf{X}_{\mathcal{N}}, \mathbf{y}) = 1 - \Phi^2(\mathbf{X}_{\mathcal{N}}, \mathbf{y}) - \Phi^1(\mathbf{X}_{\mathcal{N}}, \mathbf{y}) - \Phi^0(\mathbf{X}_{\mathcal{N}}, \mathbf{y}) \tag{41}$$

The swarm, $\mathcal{N}$, is partitioned into two disjoint sets: $\{a\}$ and $\mathcal{N} \setminus \{a\}$. Using this (41) is written as:

$$\begin{aligned}
\Phi^{3^+}(\mathbf{X}_{\mathcal{N}}, \mathbf{y}) &= 1 \\
&\quad - \big(1 - \hat{p}(\mathbf{x}_a, \mathbf{y})\big) \prod_{k \in \mathcal{N}\setminus\{a\}} \big(1 - \hat{p}(\mathbf{x}_k, \mathbf{y})\big) \\
&\quad - \hat{p}(\mathbf{x}_a, \mathbf{y}) \prod_{k \in \mathcal{N}\setminus\{a\}} \big(1 - \hat{p}(\mathbf{x}_k, \mathbf{y})\big) \\
&\quad - \big(1 - \hat{p}(\mathbf{x}_a, \mathbf{y})\big) \sum_{j \in \mathcal{N}\setminus\{a\}} \hat{p}(\mathbf{x}_j, \mathbf{y}) \prod_{k \in \mathcal{N}\setminus\{a\}\setminus\{j\}} \big(1 - \hat{p}(\mathbf{x}_k, \mathbf{y})\big) \\
&\quad - \hat{p}(\mathbf{x}_a, \mathbf{y}) \sum_{j \in \mathcal{N}\setminus\{a\}} \hat{p}(\mathbf{x}_j, \mathbf{y}) \prod_{k \in \mathcal{N}\setminus\{a\}\setminus\{j\}} \big(1 - \hat{p}(\mathbf{x}_k, \mathbf{y})\big) \\
&\quad - \big(1 - \hat{p}(\mathbf{x}_a, \mathbf{y})\big) \sum_{\mathcal{A} \in \mathrm{Comb}(\mathcal{N}\setminus\{a\}, 2)} \prod_{j \in \mathcal{A}} \hat{p}(\mathbf{x}_j, \mathbf{y}) \prod_{k \in \mathcal{N}\setminus\{a\}\setminus\mathcal{A}} \big(1 - \hat{p}(\mathbf{x}_k, \mathbf{y})\big) \\
&= 1 \\
&\quad - \prod_{k \in \mathcal{N}\setminus\{a\}} \big(1 - \hat{p}(\mathbf{x}_k, \mathbf{y})\big) \\
&\quad - \sum_{j \in \mathcal{N}\setminus\{a\}} \hat{p}(\mathbf{x}_j, \mathbf{y}) \prod_{k \in \mathcal{N}\setminus\{a\}\setminus\{j\}} \big(1 - \hat{p}(\mathbf{x}_k, \mathbf{y})\big) \\
&\quad - \big(1 - \hat{p}(\mathbf{x}_a, \mathbf{y})\big) \sum_{\mathcal{A} \in \mathrm{Comb}(\mathcal{N}\setminus\{a\}, 2)} \prod_{j \in \mathcal{A}} \hat{p}(\mathbf{x}_j, \mathbf{y}) \prod_{k \in \mathcal{N}\setminus\{a\}\setminus\mathcal{A}} \big(1 - \hat{p}(\mathbf{x}_k, \mathbf{y})\big)
\end{aligned} \tag{42}$$

Applying (12) to (42) yields:

$$\begin{aligned}
\Phi^{3^+}(\mathbf{X}_{\mathcal{N}}, \mathbf{y}) &= 1 - \Phi^0(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y}) - \Phi^1(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y}) - \Phi^2(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y})\big(1 - \hat{p}(\mathbf{x}_a, \mathbf{y})\big) \\
&= \Phi^{3^+}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y}) + \Phi^2(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y})\hat{p}(\mathbf{x}_a, \mathbf{y})
\end{aligned} \tag{43}$$

Rewriting (43) yields:

$$\begin{aligned}
\Phi^{3^+}(\mathbf{X}_{\mathcal{N}}, \mathbf{y}) &= \Phi^{3^+}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y}) - \hat{p}(\mathbf{x}_a, \mathbf{y})\Phi^{3^+}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y}) + \hat{p}(\mathbf{x}_a, \mathbf{y})\Phi^{2^+}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y}) \\
&= (1 - \hat{p}(\mathbf{x}_a, \mathbf{y}))\Phi^{3^+}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y}) + \hat{p}(\mathbf{x}_a, \mathbf{y})\Phi^{2^+}(\mathbf{X}_{\mathcal{N}\setminus\{a\}}, \mathbf{y})
\end{aligned} \tag{44}$$