ARSOn

A Robotic Search Optimization

J. Hulshof

MSc Thesis

TU Delft Delft University of Technology

Delft Center for Systems and Control

ARSOn
A Robotic Search Optimization

MSc Thesis

J. Hulshof

June 7, 2013

Faculty of Mechanical, Maritime and Materials Engineering (3mE) · Delft University of Technology

Almende BV Rotterdam, FireSwarm Project

University of Technology Delft, Delft Centre for System and Control

Abstract

Robotic search is a very active field of research, and especially search with multiple robots is of high interest. A swarm of robots equipped with sensors could be used in a variety of useful settings such as border patrol, monitoring water quality of the ocean with underwater robots, or – in case of the FireSwarm project – for finding dune fires with Unmanned Aerial Vehicles (UAVs). This thesis was assigned by the research company Almende B.V. and is related to the FireSwarm project. The main idea behind projects like FireSwarm is that a large group of cheap UAVs with cheap less reliable sensors can search more effectively than one expensive UAV with more reliable sensors.

Therefore, the main focus in this thesis is on the development of efficient robotic search strategies for different sensor representations. The robotic search problem that is defined in this thesis is the problem of finding an important point (a fire) in a predefined search area as fast as possible with autonomous UAVs. Herein, a clear distinction is made between search with perfect (deterministic) sensors and search with more realistic and imperfectly (stochastically) modeled sensors. The modeling of the fire sensors is very general in order to be able to represent any kind of (fire) sensor. This is achieved by, instead of qualitatively modeling the effect of each factor on the fire sensor, modeling the sensor as a simple stochastic process.

Two main conclusions can be drawn from the tests performed in this thesis. First, the optimal strategy for search with a single UAV, with perfect deterministic sensors, is shown to be a predefined sweeping path. Furthermore, the methods proposed in this thesis for predefining a search path for multiple UAVs with deterministic sensors resulted in near-optimal performance. Secondly, it is concluded that for increasingly stochastic sensors, the predefined sweeping path is easily outperformed by the simple greedy strategy proposed in this thesis.

So, because of the general representation of the search problem, the results found in this thesis cannot only contribute to the development of fast efficient algorithms for the FireSwarm project but also to other real-world (robotic) search problems.

Table of Contents

Chapter 1

Introduction

Robotic search is a very active field of research, and especially search with multiple robots is of high interest. A swarm of robots equipped with sensors could be used in a variety of useful settings – such as border patrol, monitoring water quality of the ocean with underwater robots, or in case of the FireSwarm project – for finding dune fires with Unmanned Aerial Vehicles (UAVs). This thesis is assigned by the research company Almende B.V. and related to the FireSwarm project. The goal of this thesis is to find theoretical answers for the development of efficient search algorithms in order to aid the practical development of algorithms for FireSwarm and future robot search related projects at Almende.

Figure 1-1: Picture of the UAV used in FireSwarm

The FireSwarm project is a collaboration between the fire department of Bergen, Delft University of Technology (MAVlab), University of Groningen, and the companies DevLab (Salland Electronics), EagleVision, and Almende B.V. The main idea behind the FireSwarm project is that a swarm of cheap UAVs with cheap less reliable sensors can search more effectively than one expensive UAV with more reliable sensors.

The objective of the FireSwarm project is to control a swarm of autonomous UAVs such that a fire is found as quickly as possible. More specifically, ten UAVs will be deployed in a predefined square search area of 5 by 5 kilometers to search for one or more starting fires that have a footprint of approximately 2 by 2 meters. The UAVs used in FireSwarm are

electrically powered airplanes, see Figure 1-1. The UAVs are equipped with an autopilot that can be used for waypoint navigation, communication equipment to send data to other UAVs and sensors to detect fire. The sensors that are used for the detection of fire are a combination of cameras and infrared heat sensors.

Each UAV is equipped with a downward facing camera, which is used to capture images from the ground. This image is processed and analyzed by the fire detection software. The software will return a statement whether there is fire detected or not. The probability or reversely the uncertainty of the fire detection is caused by the following aspects:

- The position of the UAV is determined using the Global Positioning System (GPS) and is therefore only an estimation.
- The velocity of the UAV depends on the wind.
- The roll of the UAV is being constantly adjusted by the autopilot, causing the wings to wiggle, which changes the projection of the image of the ground especially in relationship to the position of the UAV.
- The UAV can fly at different altitudes, which also changes the camera image.
- The fire detection software uses a series of techniques that have each a certain probability of classifying the fire and may result in false positives and negatives.

1-1 Problem Statement

In related research robotic search problems are mostly defined for a specific application with specific sensors and dynamics. This has resulted in a vast variety of solutions for very specific problems. However, the theoretical questions that arise from the general (robotic) search problem remain unsolved. Therefore, the work in this thesis is aimed at the more fundamental questions in the field of robotic search that are also present in FireSwarm.

One of the most important parts of robotic search is the sensor; so this is where the main part of the research is aimed at. The fire detection in the FireSwarm project is the result of a series of complex algorithms that are fed with sensor information that depends on a large number of factors, including the actual movement, orientation, and position of the UAV. Instead of using a very detailed model to qualitatively describe the influence of all factors on the fire detection sensor, specifically for FireSwarm, the sensors are modeled by a simple and more general stochastic process. This general representation of the sensors can be compared to the way the sky is searched with a searchlight, where only objects in the lit area can be seen with some probability and no measurement can be taken of the object outside of the boundary of the light beam, see Figure 1-2.

So the main question that is to be solved in this thesis is: “What is the most efficient way to search some predefined search area such that a fire is found as quickly as possible with some predefined certainty”.

Figure 1-2: An Auxiliary Territorial Service girl crew works a searchlight near London, on January 19, 1943, trying to find German bombers for the anti-aircraft guns to hit. (AP Photo).

1-2 Overview of the Thesis

This thesis starts with an overview of related work in Chapter 2. In that chapter different approaches to coverage, exploration, and search are discussed and compared. After the discussion of related work the research of this thesis starts.

The main focus in this thesis is on the sensor representation and its effect on the optimal search strategy. A clear distinction is therefore made between search with perfect (deterministic) sensors and search with more realistic and imperfectly (stochastically) modeled sensors. Furthermore, for both sensor representations, the problem is first solved for the single UAV case and later extended to multiple UAVs. This last distinction is made in order to simplify the problem at first so we can see what makes the general problem hard to solve.

An overview of the research problems can be found in Figure 1-3.

Chapter 3 addresses Problem 1, i.e., robotic search with a single UAV with perfect deterministic sensors and Problem 2, i.e., robotic search with multiple UAVs with deterministic sensors. In this chapter a method will be given that optimally solves the deterministic search problem with a single UAV. This chapter also proposes methods for the multiple UAV case with deterministic sensors. These proposed methods approach the lower boundary of the minimum path length under certain conditions.

Chapter 4 continues with Problem 3: the problem of robotic search with a single UAV with stochastically modeled sensors. In this chapter the optimal method for the deterministic case will be tested with stochastic sensors and compared with new methods that were suggested in this thesis in order to solve the stochastic case. Furthermore, the last problem, Problem 4 involving the multi-robot case with stochastic sensors, will be addressed briefly.

General conclusions are drawn in Chapter 5 and recommendation are given there for future work on the subject of robotic search with stochastically modeled sensors.

Note that in this thesis the general problem of finding an important point in some search area with any kind of robot is referred to as the Single or Multi-Robot Search problem, whereas the more specific problem in this thesis is referred to as the search for fire with single or

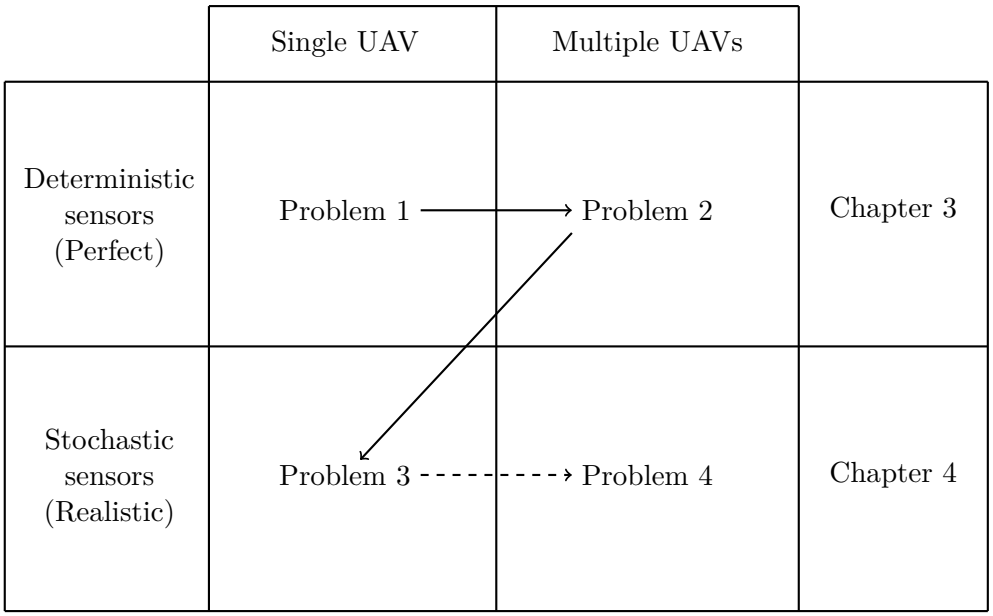


Figure 1-3: A schematic overview of problems discussed in this thesis.

multiple UAVs. So in this thesis an UAV is a special case of robot and a fire is a special case of an important point.

Chapter 2

Related Work

2-1 Introduction

This chapter focusses on different methods for coverage, search, and exploration. Four common methods are described in this chapter. These methods are: frontier-based exploration, exact cellular decomposition, policy optimization, and path planning Optimization. Each of these methods is very different from the other ones, so an overview is given of what is exactly optimized and how the overall control scheme of each method works. This chapter is concluded with the preferred approach for the search objective that is proposed in the FireSwarm project.

2-2 Frontier-Based Exploration

The first approach is very intuitive and asks the following general question [1]:

"Given what is known about the world, where should you move to gain as much new information as possible".

This can be formalized by (2-1), when exploration is formulated as coverage. In order to do this, it is assumed that the exploration is carried out with sensors that measure perfectly, deterministically, and uniformly over the entire sensor area. Deterministic sensors mean, in the context of this thesis, that when a certain area is sensed multiple times the measurement is exactly the same every time it is measured on that particular location. This formulation of exploration is given by

$$\underset{x}{\operatorname{argmin}} \ |A_k(x) \cap \bigcup_{m=1}^{k-1} A_m| \tag{2-1}$$

where the sensor area $A_k \subset \mathbb{R}^2$ is the predefined 2-dimensional area that can be measured on discrete time step k from position $x \in \mathbb{R}^2$ and $|S|$ is the surface of area S. So by solving (2-1) the overlapping area, of the current sensor area and the union of the previously sensed areas, is minimized. Therefore maximizing the newly explored or covered area.

An attempt to solve (2-1) resulted in the frontier-based exploration heuristic that was introduced in [1]. The frontier-based exploration is formulated as:

"To gain the most new information about the world, move to the closest point at the boundary between the known area and the unknown area."

In this way when the robot moves to the frontier, defined as the boundary between the explored and the unexplored area, new information is obtained. In Figure 2-1 the frontier-based exploration strategy is visualized. From this figure it can be seen that by consecutive movements of a mobile sensor to the frontier, the frontier will be pushed forward and the explored area will expand.

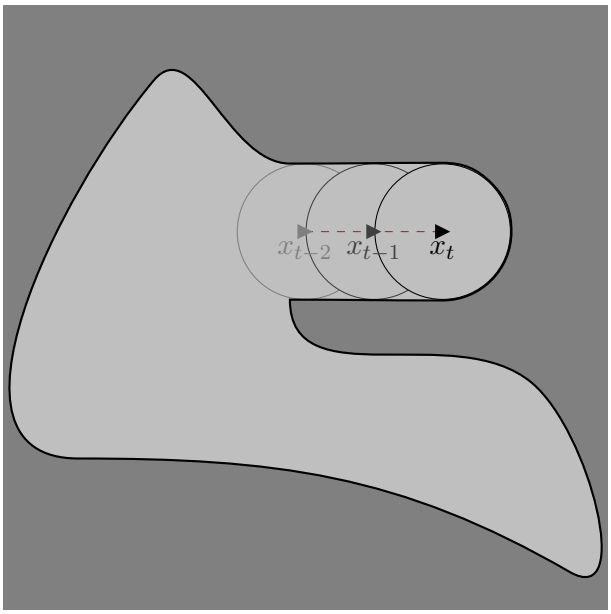


Figure 2-1: Three robot locations at three consecutive time steps of a robot its path while carrying out frontier-based exploration. Positions of the robot are marked with triangles, circles mark the sensor range, the thick black line is the frontier between the covered area (light gray) and uncovered area (dark gray)

In Figure 2-1 a schematic overview is depicted of a frontier-based Exploration. In the first block a representation of the world is used as an input, then (2-1) is optimized. The next position is used as a reference for the low-level position control that determines the actual position of the robot and that is based on the dynamics of the robot. When the actual robot coordinates are known, the world image is updated.

An example of frontier-based exploration of a multi-robot system is enforced by the constraint that the future location has to be on the frontier, such that the gain of new information is guaranteed [2]. This heuristic for frontier-based exploration can be formalized by:

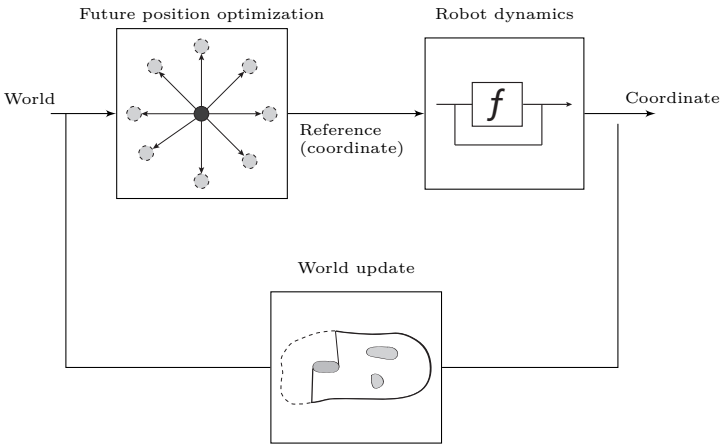


Figure 2-2: Schematic overview of frontier-based coverage. The first block represents some optimization of the future position, i.e. maximizing information gain or minimizing the distance to a position that results in the gain of new information. The optimal position is used as a reference for the position controller that uses the robot dynamics to control the actual position of the robot. At the new location information is gained and used to update the world map. The updated map will then again be used for the next position optimization.

$$x_{k+1} = \underset{x \in X_{k+1}}{\operatorname{argmin}} \quad \|x - x_k\| \tag{2-2}$$

where the function $\|p_1 - p_2\|$ returns the Euclidean distance between the two points p_1 and p_2. The set X_{k+1} is defined as the set of 8 points that can be found by moving in one of 8 directions $\{\text{North}, \text{North-West}, \text{West}, \cdots, \text{North-East}\}$. The first crossing with the exploration frontier or some maximum distance away from the current position in the given direction, is then the point that is included in the set X_{k+1}. So by finding the argument of the minimization of (2-2) the shortest path to the exploration frontier is chosen. This method can therefore easily be adjusted to additional objectives by adding constraints as long as there will still remain feasible solutions. Figure 2-3 is taken from [2] and depicts an example of future robot positions.

The approach of frontier-based exploration proves to be an effective heuristic to explore unmapped territory, even if there is no knowledge of the world or even of the dimensions of the search area. However, frontier-based exploration is not necessarily the optimal way to cover an area or even to maximize information gain, especially when the coverage is considered over several time steps. In other words a frontier-based approach only considers the optimization of the position at the next time step and therefore does not consider the total time frame of the run.

2-3 Exact Cellular Decomposition

When the dimensions of the search area are known and the sensors and position control are deterministic it is possible to find the optimal path that covers a certain search area with

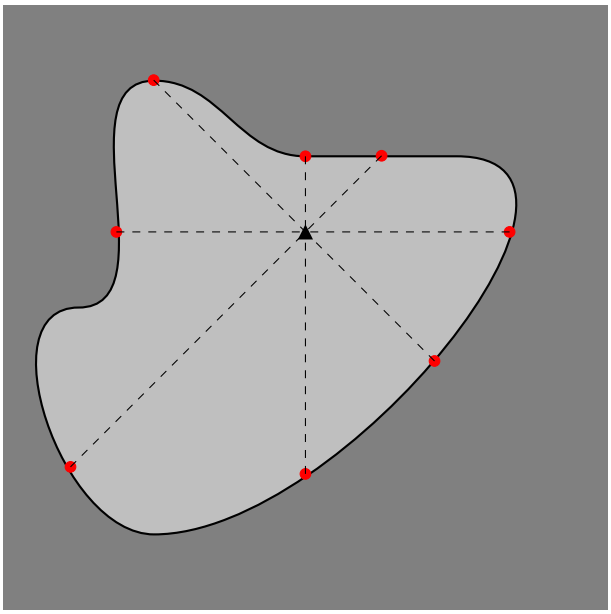


Figure 2-3: Possible future positions based on current position are given by the red dots. The dark gray area represents the unknown area and the light gray the known area.

the shortest possible total path [3, 4]. The definition for optimal coverage that is proposed is the shortest path that covers the whole area. When looking at the case where the traveled distance during one time step is constant this objective can be defined as:

$$\underset{x_1,\cdots,x_N \in \mathbb{R}^2}{\arg\min} \left\{ N : \bigcup_{n=1}^{N} A(x_n) \supseteq S \right\} \tag{2-3}$$

This is a minimization of the number of points N that it takes to cover the entire search area $S \subset \mathbb{R}^2$ by a single robot with a predefined sensor area $A(x) \subset \mathbb{R}^2$ that is a function of the position x, i.e., a circular area with the center at position x. In other words the number of measurements that it takes to measure the whole search space, is minimized.

For a simple rectangularly or even trapezoidally shaped area the optimal path for total coverage by a single robot is easily found in a boustrophedon (literally: "the way of the ox") or ox-like path (see Figure 2-5). The ox-like path is a path that resembles the way a field is plowed by an ox. However, when obstacles are present and multiple robots are used the problem gets complex and cannot just be covered optimally by a ox-like path. To make it less complex the problem can be decomposed into three subproblems [3, 4]:

- the decomposition of the search area in to trapezoidally shaped cells
- the coverage of trapezoidally shaped cells (which has already been solved optimally with the ox-like path, see Figure 2-5)
- the path between the cells

The above decomposition shifts the optimization problem from optimizing the measurement locations to the minimization of the path between the cells that results after the decomposition of the search area.

Exact cellular decomposition methods decompose the free area F into a set of cells denoted by C, the union is a proper superset of the original free area. The free area F is given by:

$$F \supset S \setminus \bigcup_{i=1}^{k} O_i \tag{2-4}$$

$$\tag{2-5}$$

where O_i, $i \in \{1, \cdots, k\}$ are the areas of the obstacles in the search area. The operator $\setminus$ indicates the set difference. So F is the total search area S except for the area that is covered by the set of obstacles. The cellular decomposition is chosen in such a way that the individual cells can be optimized by a simple ox-like path, see Figure 2-5.

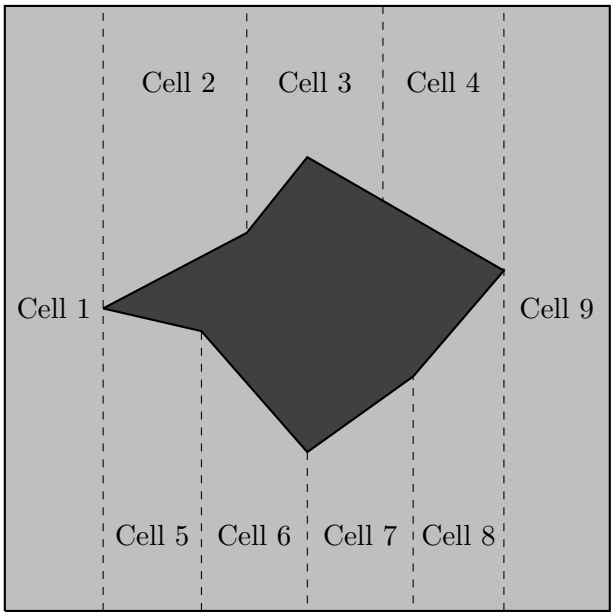


Figure 2-4: Exact cellular decomposition of a rectangular area in nine trapezoidal shapes. The light gray area is to be explored, the dark gray area is an object. The shape of the cells are chosen so they can be covered optimally by a Boustrophedon path as is displayed in Figure 2-5. This figure is inspired by [3].

The same decomposition of the coverage problem with perfect sensors is shown in the schematic block scheme in Figure 2-6. An image of the world is used as an input to decompose the search area into cells. An optimal path is planned, which is the shortest path that visits all cells. Then, for every cell, an optimal coverage path is planned, this path is a tuple of coordinates that is used as a reference for the low-level controller of the robot.

The obvious assumption can be made that with more than one single robot the coverage of the area can be faster [4]. The search area is then decomposed in the same way as was

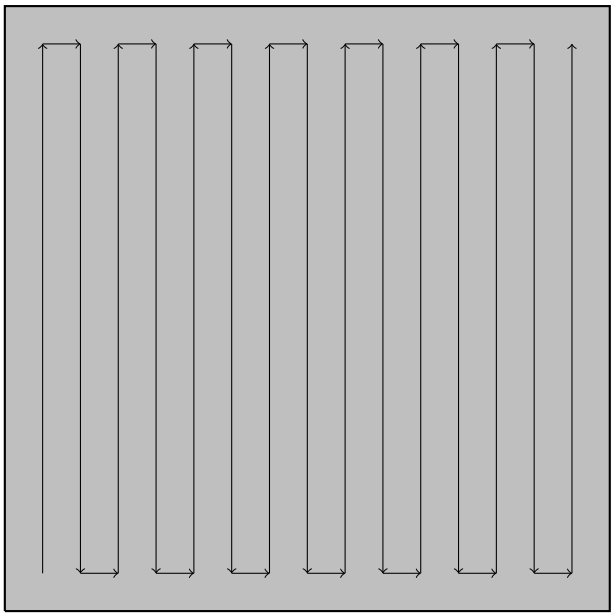


Figure 2-5: Boustrophedon path, (literally: "the way of the ox"). The path that is shown is a known optimal solution for the coverage of a rectangular area. This figure is inspired by [3].

proposed in the single robot cellular decomposition approaches [3] but with multiple robots. Just as in the single robot case the coverage problem can be decomposed in to the same three subproblems, the decomposition of the search area in to cells, the coverage of these individual cells, and the path between those cells. Although a Multi-Robot approach is proposed that optimally covers the individual cells, [4], it does not find the optimal, shortest path between cells.

In [5] a decentralized Multi-Robot approach is proposed that uses cellular decomposition to solve the coverage optimization problem. At every time step, based on the positions of the robots, the search space is partitioned into Voronoi cells (see Figure 2-7), each robot has its own cell.

A Voronoi tessellation is a means of dividing an area into a number of cells depending on a set of reference points. The Voronoi cell of a particular reference point (i.e. the position of a robot) represents the space of all points that are closer to this given reference point than it is to any of the other reference points in the set [6].

The coverage of the cell is determined by calculating the optimal orientation δ_m^* for each robot $m \in \{1, \cdots, M\}$ by maximizing an objective function. This optimization is given by:

$$\delta_m^* = \underset{\delta_m}{\operatorname{argmax}} \, \mathcal{H}_{\text{discover},m}(x_m, \delta_m) \tag{2-6}$$

where the position of robot m is denoted by $x_m \in \mathbb{R}^2$, x_m.

The objective function calculates the performance by integrating the product of the performance function f and the density function $\phi(q)$ over all the frontiers, ∂S_m. The frontier

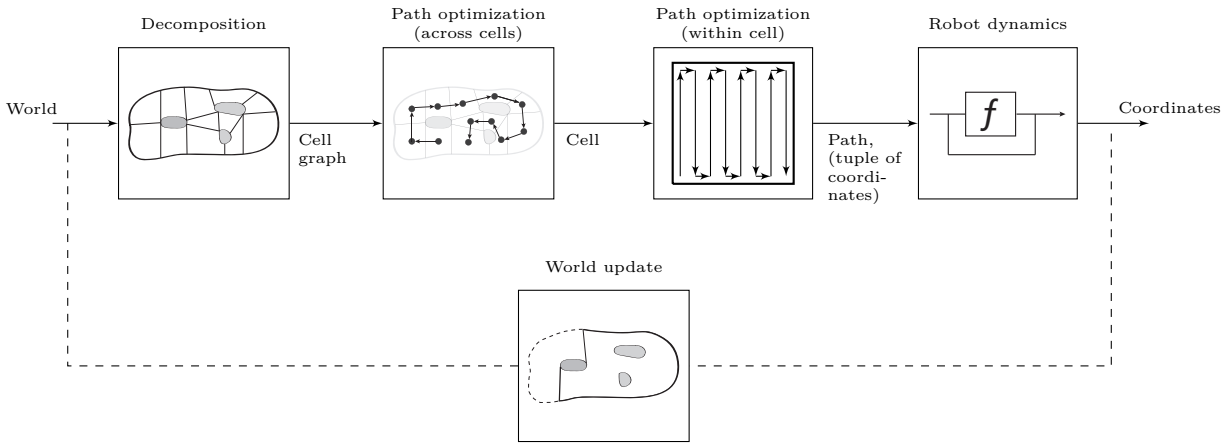


Figure 2-6: Schematic of a Cellular Decomposition coverage method. This method needs a total model of the world or search area that is to be covered. Then this world is decomposed into cells, the path to move from cell to cell is optimized and finally the path within the cell is optimized. The optimized path is a tuple of coordinates that is used as a reference for the low-level position controller of the robot. As an extension a feedback loop, displayed by the dashed line, can be made to update the world and correct for any errors during the run.

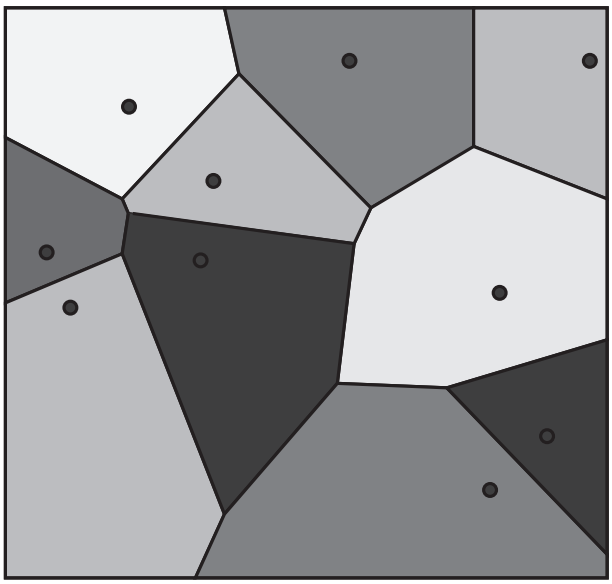


Figure 2-7: A Voronoi decomposition of 10 sites (black dots) in Euclidean space. A Voronoi cell of a particular site (i.e. the position of a robot) represents the space of all points that are closer to the corresponding site than to any of the other sites.

is defined as the boundary between covered and uncovered area, The density function $\phi(q)$ encodes any information gain at the point $q \in \partial S_m$, so q is a point on the frontier ∂S_m.

The objective function is given by:

$$\mathcal{H}_{\text{discover},m}(x_m, \delta m) = \int_{\partial S_m} f(x_m, \delta_m, q)\phi(q)dq \tag{2-7}$$

The performance function f is given by:

$$f(x_m, \delta_m, q) = \exp\left(-\frac{\alpha^2(x_m, \delta_m, 1)}{2\theta^2}\right) \exp\left(-\frac{\|q - x_i\|_2^2}{2\sigma^2}\right) \tag{2-8}$$

where σ and θ are the standard deviations of the Gaussians. The performance function $f(x_m, \delta_m, q)$ consists of a product of two components: the angular component and the distance component.

The angle α is given by:

$$\alpha(x_i, \delta_i, q) = \sphericalangle(q - p_i) - \delta_i + \begin{cases} 2\pi & \text{if} \quad \gamma < \pi, \\ -2\pi & \text{if} \quad \gamma > \pi, \\ 0 & \text{else.} \end{cases} \tag{2-9}$$

The symbol $\sphericalangle$ denotes the spherical angle. This approach is a combination of exact cellular decomposition (Voronoi) and a frontier-based objective function that tries to maximize information gain at every time step for each robot. The Voronoi decomposition can be obtained locally by each robot as long as the robot network is fully connected, which enables position sharing of all the robots in the system.

So the methods based on an exact cellular decomposition of the search space propose an effective heuristic to cover a certain area. However, all methods proposed here assume some special case of the coverage area, e.g. the perfect decomposition of the free area into trapezoids. Therefore, the heuristics that are given by these methods are limited to only very specific cases of the coverage problem.

2-4 Policy Optimization

When dealing with an unknown environment and when a path has to be covered over and over again, as for example in cleaning tasks, self-learning techniques can be useful. A practical example of this is a household robot with a finite state machine that uses neural networks to learn over a number of training runs in order to derive the optimal policy for covering a room with obstacles [7]. This approach only updates its policy after each run, so in the first run there will be no learning. Furthermore, the policy that is generated is the optimization of an heuristic that leads to the optimal coverage of one specific search area and the specific unknown dynamics of the used system.

Another self-learning approach uses neural networks more directly, where a neuron in the network directly corresponds to a position in the real world. The neural network is trained

in order to find the best path for total coverage [8]. This algorithm adjusts already on-line during the first run instead of only at the end of each trial. The neural network approach, proposed in [8], discretizes the search area into a grid of 30 by 30 positions. The actions that can be taken from each position are also discretized in to a set of actions.

The neural network has one neuron for every position and a link from each position to the neurons that are in the same neighborhood, see Figure 2-8.

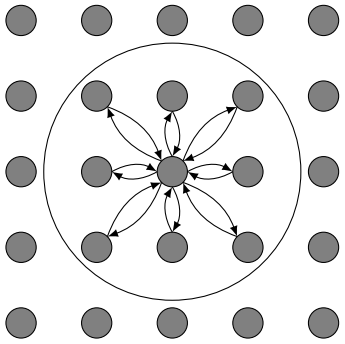


Figure 2-8: Discretized actions and positions represented by a neural network. The arrows represent actions of the robot and links in the neural network. The circles represent positions of the robot and neurons in the neural network. This figure is inspired by [8].

Furthermore, the proposed neural network approach is capable of planning coverage paths with multiple robots when treating other robots as obstacles.

The schematic of this approach is displayed in Figure 2-9. The methods for policy optimization need some prior policy, often randomly generated, to give the first actions. The actions are directly imposed on the dynamics of the robots and this will give the next state of the robot, which can be some path or a single location that results in new information about the world. This new information is used to build a map of the world. This world map can be used as an assessment in order to see how effective the current policy is. Then the policy can be adjusted and the cycle starts over again, depending on the method the policy can be assessed as frequent as every time step [8] or at the end of the task [7].

In a static environment, where the Markov property holds, both methods should eventually, after a number of executions, converge to the optimal policy that results in optimal coverage. An advantage of the methods proposed in this section is that the dynamics of the robot or even the interaction between the world and the robot do not need to be known, or in other words the methods can learn in which state the robot will end up given some action and the current state. However, a disadvantage of the optimization of the policy is that it takes a high number of executions of the policy to converge. Therefore, policy optimization methods discussed here are inherently not focused on the optimization of coverage in the first run but try to find a general policy that works optimally in similar situations where the system can be described accurately by the same discrete states. Alternatively, policy optimization could also be used to plan a path off-line a priori to the execution in the real world. However, this would obviously require an accurate (dynamic) model of the system.

So the resulting policy can be seen as a self-learned heuristic solution that can be applied to situations that are similar. Based on the choice of the discrete states of the systems the

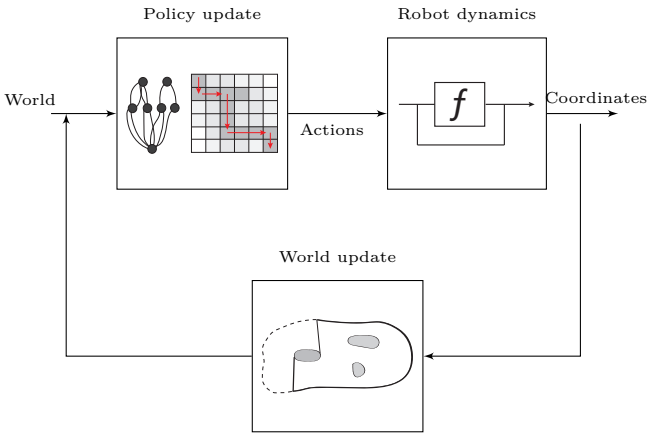


Figure 2-9: Schematic of the policy optimization scheme. The methods for policy optimization need some prior policy, often randomly generated, to give the first actions. The actions are directly imposed on the dynamics of the robots and this will give the next state of the robot, which can be some path or a single location that results in new information about the world. This new information is used to build a map of the world. This world map can be used as an assesment in order to see how effective the current policy is. Then the policy can be adjusted and the cycle starts over again.

policy can represent a heuristic for only the next step or even the total path to take given the current state.

2-5 Path Planning Optimization

Another way to optimize an objective such as coverage or search is by path optimization. Path Optimization tries to optimize some fitness function that depends on the planned path and is designed towards some objective, in the case of this thesis, as search or coverage. The path can be represented as a single point in a high-dimensional space where for each way-point in the path a coordinate is represented in two dimensions. This gives the following definition for a path by a coordinate in high-dimensional space:

$$p \in \mathbf{R}^{N \times 2} \tag{2-10}$$

where p is the coordinate that represents a possible path and N is the number of points that form the path.

Path planning optimization tries to find the best solution, which is a coordinate as is defined by (2-10), in this continuous high-dimensional solution space. One could use Particle Swarm Optimization (PSO) to optimize such a high-dimensional function.

In [9], a cooperative path planning approach is proposed that uses PSO to plan the optimal, shortest path for all rescue searchers in order to find a lost person as fast as possible. The proposed method uses a fitness function that assesses the total path of all the rescue searchers

in the search party such that the length of the combined search path is minimized. PSO is used as an optimization technique in [9] because it can optimize a continuous high-dimensional function and can cope with local optima. However, the convergence cannot always be proven when the fitness function is non-convex.

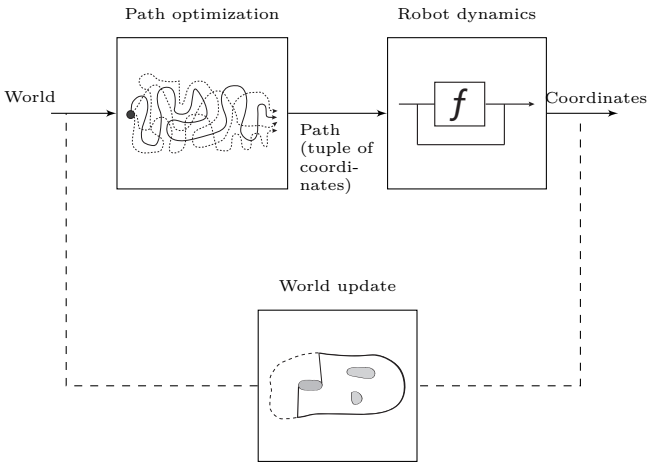


Figure 2-10: Schematic of the path planning optimization scheme. The optimization of a path uses an (in)complete map of the world as an input to assess the fitness of a path. The optimized path is used as an input for the low-level position controller of the robot that actually carries out the planned path. Depending on the method the path can be recalculated using an updated world map.

2-6 Conclusions

Simple heuristic approaches could give a solution for the search problem of a dynamic unknown environment with non-perfect sensors and position control. The disadvantage of heuristics such as frontier-based exploration is that, at best, the objective is only optimized over the next time step. The behavior that emerges from these simple heuristics is very hard or even impossible to predict and will be very unlikely to complete the task optimally.

The heuristic method of exact cellular decomposition can offer, for some simplified cases of the coverage problem, a heuristic that results in optimal coverage. However, the cellular decomposition methods assume a static world where sensors behave deterministically. The advantage of these methods is that they offer solutions that can be calculated in advance. Another approach to the optimization of the path length to ensure total coverage, is by using policy optimization off-line on a model of the system. The advantage of using policy optimization is that it is able to generate an optimal solution for any arbitrary lay-out of the area that is to be covered.

Furthermore, policy optimization could also be used on-line to optimize a search path when the world and the dynamics of the system are not known. The on-line implementation of policy optimization could be used to optimize every next step or to optimize the entire path that is needed to cover some area. In the first case a policy is optimized after taking a number of steps and results in the optimal policy for taking one step, given the current state. In other

words we could say that the resulting policy is a self-learned heuristic method for planning the next step. The second case optimizes the entire path that ensures full coverage and will therefore need several runs, i.e. the execution of entire paths, to converge to the optimal policy.

Finally, path planning optimization uses PSO to optimize a high-dimensional fitness function that depends on each waypoint of the path. However, it can only be used to optimize the path based on the information that is known about the world at the time of the optimization. The advantage of this method is that it can plan a path that consists of continuous coordinates based on a fitness function that can describe any objective as long as it can be described as a function that depends on the coordinates of the path. The main disadvantage is that the optimization of the fitness function gets increasingly more computationally expensive when the number of waypoints in the path increases.

Chapter 3

Coverage (Deterministic)

3-1 Introduction

In this chapter It is shown that the search problem, as was described in the Introduction of this thesis, can be formulated as a coverage problem. This simplification can be made when perfect deterministic sensors are assumed. Furthermore, the coverage problem is decomposed into two parts. First, in Section 3-2, the search area is decomposed into a minimum number of observations and this optimal set of locations will be given. Secondly the optimization of the path that joins these observation points, is discussed in Section 3-3. This section will give the optimal path for the single robot dynamic coverage problem and the results of four coverage methods that pose near-optimal solutions for the multi-robot dynamic coverage problem.

3-2 Static Coverage

3-2-1 Problem Decription

When search is assumed with perfect deterministic sensors, the task of finding an interesting point, such as a fire, is reduced by just looking everywhere in the search space at least once. A perfect deterministic sensor is, in the context of this thesis, a sensor that can perfectly detect whether a circular area with radius r around the current position (x, y) of the Unmanned Aerial Vehicle (UAV) is occupied by a fire. So now the problem is to find the smallest possible set of observation positions that, when combined, can observe the whole search area. A schematic representation of the problem is displayed in Figure 3-1. Some additional definitions and assumptions are given by:

- The search area is a unity square in 2-dimensional Euclidean space that has its corners in the points $(0, 0)$, $(0, 1)$ $(1, 1)$, and $(1, 0)$, $\in \mathbb{R}^2$.
- The position of each way-point $W \in \{W_0, \cdots, W_N\}$ is a point $(x, y) \in \mathbb{R}^2$.

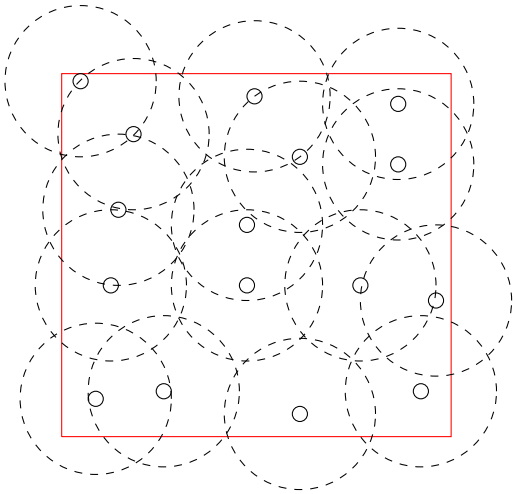


Figure 3-1: A schematic representation of a sub-optimal solution to the static coverage problem. The open dots mark the way-point of the UAV during an observation. The circle that surrounds that open dot, marks the observed area. The red box marks the search area.

- The area that is observed from each way-point W_n is a circular area with radius r with its center in W_n.

The formulation of the minimization problem is given by:

$$\underset{\{W_0,\cdots,W_N\}}{\arg\min} \left\{ N | \ \forall \ W \in [0,1]^2 \ \exists \ W_n \in \{W_0,\cdots,W_N\} : \ \|W_n - W\| \leq r \right\} \tag{3-1}$$

What this formula basically states is that the optimal set of way-points is the smallest set of way-points, with size N, to satisfy the constraint that for any possible way-point W that lays inside the unity square, there exists a way-point W_n, which is a member of the set of way-points $\{W_0,\cdots,W_N\}$ that is not further away from W than a distance of r.

3-2-2 Optimal Solution

The coverage of rectangular shapes by equally sized circles is proven to be optimally covered by a hexagonal formation, given that the radius of the circles goes to zero [10]. A representation of this hexagonal formation can be seen in Figure 3-2 and the coordinates of the grid points are, for arbitrary rectangular grids of size $A \times B$, defined by:

$$W_{\text{hexa}}(a,b) = \begin{cases} \left(\sqrt{3}r(a-1) + \frac{\sqrt{3}r}{2} \quad , \frac{3}{2}(b-1) + \frac{1}{2}\right) & \text{if } b \text{ is odd} \\ \left(\sqrt{3}r(a-1) \qquad\quad , \frac{3}{2}(b-1) + \frac{1}{2}\right) & \text{otherwise} \end{cases} , \in \mathbf{R}^2 \tag{3-2}$$

where $a \in \{1,\cdots,A\}$ and $b \in \{1,\cdots,B\}$.

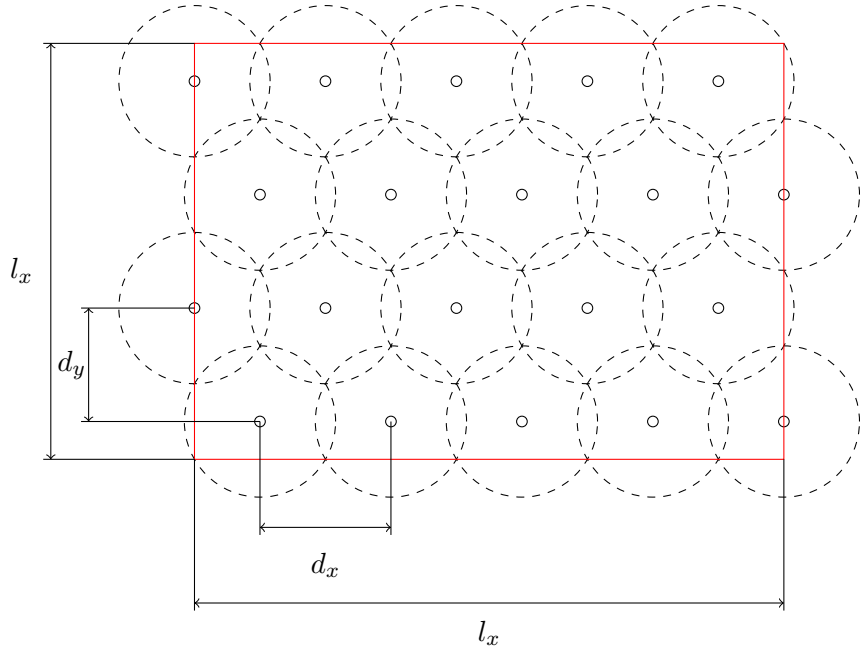


Figure 3-2: A hexagonal grid of circles that is the optimal coverage of rectangular search areas, given that the radius of the circles goes to zero. The distances d_x and d_y are given by $d_x = \sqrt{3}r$ and $d_y = \frac{3r}{2}$, where r is the radius of the circles and the sensor range of the UAVs. The dimensions of the search area are given by l_x and l_y.

The condition that the radius of the circles has to go to zero, can also be formulated as that the number of circles has to be infinite. So practically this means that for the hexagonal grid heuristic to be close to optimal the number of circles N has to be very large. When N becomes larger the overlap ratio:

$$\frac{O_{\max} - O_{\text{hexa}}}{O_{\max}} \tag{3-3}$$

converges to $1 - \frac{3}{2\pi}\sqrt{3} \approx 0.173$, where $O_{\max}$ is the maximum combined area of all the N circles when there would me no overlap and O_{hexa} is the rectangular area that is covered by the hexagonal grid heuristic. To arrive at this expression the areas $O_{\max}$ and O_{hexa} are computed as:

$$O_{\max} = Nr^2 = AB\pi r^2 \tag{3-4}$$
$$O_{\text{hexa}} = l_x l_y \tag{3-5}$$
$$\tag{3-6}$$

where l_x and l_y are given by:

$$l_x = (A-1)d_x + \frac{1}{2}d_x \tag{3-7}$$
$$l_y = (B-1)d_y + r \tag{3-8}$$
$$\tag{3-9}$$

Combining (3-5) with (3-7) and (3-8) will give:

$$O_{\text{hexa}} = \left((A-1)d_x + \frac{1}{2}d_x\right)\left((B-1)d_y + r\right) \tag{3-10}$$

$$\tag{3-11}$$

Applying $A = B$ and $r = 1$ to (3-4) and (3-5) yields:

$$O_{\max} = A^2\pi \tag{3-12}$$

$$O_{\text{hex}} = \left((A-1)\sqrt{3} + \frac{\sqrt{3}}{2}\right)\left((A-1)\frac{3}{2} + 1\right) \tag{3-13}$$

$$= \left(6A^2 - 5A + 1\right)\frac{\sqrt{3}}{4} \tag{3-14}$$

$$\tag{3-15}$$

Now the limit of the overlap ratio can be expressed for $A = B \to \infty$ as:

$$\lim_{A\to\infty} \frac{O_{\max} - O_{\text{hex}}}{O_{\max}} = \lim_{A\to\infty} \frac{A^2\pi - \left(6A^2 - 5A + 1\right)\frac{\sqrt{3}}{4}}{A^2\pi} \tag{3-16}$$

$$\tag{3-17}$$

$$= \frac{A^2\pi - A^2\frac{6}{4}\sqrt{3}}{A^2\pi} = 1 - \frac{3}{2\pi}\sqrt{3} \approx 0.173 \tag{3-18}$$

When the ratio of overlap is plotted it can be seen that for smaller number of circles the overlap quickly explodes and the heuristic is no longer a good solution, see Figure 3-3.

So the hexagonal grid gives a heuristic for the optimal coverage of any arbitrary rectangle by equally sized circles as long as the number of circles is very high and the radius of the circles is very small compared to the width and height of the rectangular covered area.

3-3 Dynamic Coverage

3-3-1 Problem Description

The solution to the minimization problem (3-1) will yield a set of N coordinates for a given search area with dimension l_x, l_y, and the observation radius of the sensor r. These coordinates ensure, once visited, that every point in a square of size 1×1 is observed (covered). The next problem will be the problem of optimizing the path. The optimal path would be the path that minimizes the total distance traveled while visiting every point $W_n \in \{W_1, \cdots, W_N\}$ at least once. To better represent the problem it is defined as a graph. The vertices V are given by the set of way-points $\{W_1, \cdots, W_N\}$, the edges E in the graph represent the distances

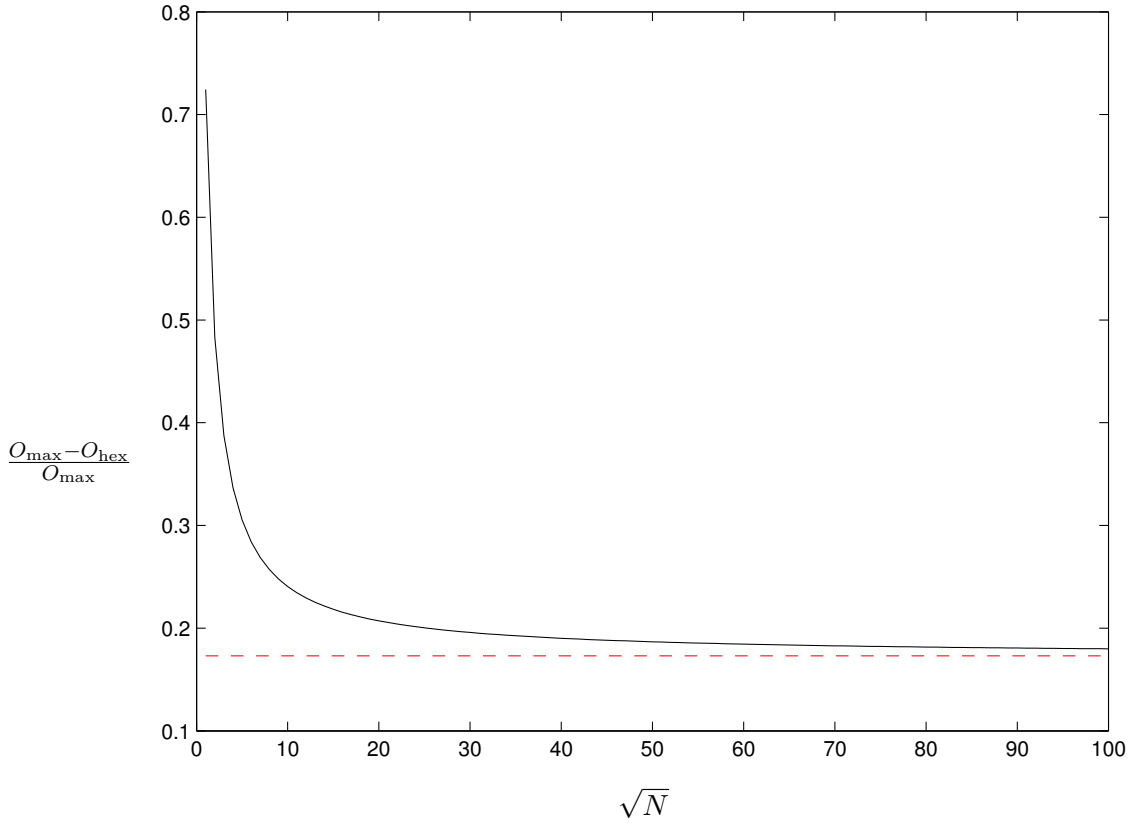


Figure 3-3: A plot of the overlap ratio: $\frac{O_{\max}-O_{\text{hex}}}{O_{\max}}$ as a function of the number of circles N, shown by the black curve. The dashed line gives the limit that the ratio converges to.

between the way-points in 2-dimensional euclidean space. So the edges represent paths that can be traveled by the UAV. Since it is possible to travel from any way-point to any other waypoint, the graph is fully connected. This fully connected graph is given in Figure 3-4, in this figure all possible paths are plotted between any pair of two waypoints.

Traveling Salesman Problem

The problem of finding the shortest path that visits all points (except for the starting point) of a given set of points, is called the Traveling Salesman Problem. Since the traveling sales man problem is known to be NP-hard for an arbitrary set of points it would not be wise to try and find its solution in this thesis. It is more efficient to try and exploit the structure of the graph and its properties.

Lower Bound

First the lower bound of the length of the path is investigated. The hexagonal grid that is displayed in Figure 3-2 yields a regular structure where each point, except for those on the

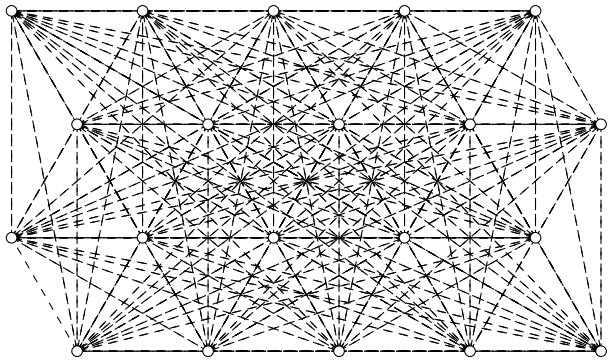


Figure 3-4: A fully connected graph were every path between any set of two way-points is plotted as a dashed line. The waypoints are displayed as small open circles and lay on the hexagonal grid that is defined by 3-2

border of the graph, is closest to six other grid points at a distance of $\sqrt{3}r$. The shortest distance between two points in the hexagonal grid is $\sqrt{3}r$ and at least $N-1$ edges are needed to connect all vertices to each other plus one additional vertex is needed to connect the last visited vertex to the first one. So the lower bound of the problem of finding the shortest path is: $N\sqrt{3}r$.

T-graph

So in order to find an optimal solution, a path could be selected that only uses N edges of length $\sqrt{3}r$. This means that all edges that have higher values than $\sqrt{3}r$ can be disregarded and the fully connected graph will be reduced to the induced subgraph with only edges that represent distances of $\sqrt{3}r$. The graph that only features edges of length $\sqrt{3}r$ and has vertices that are given by a hexagonal grid, Figure 3-2, is called a T-graph.

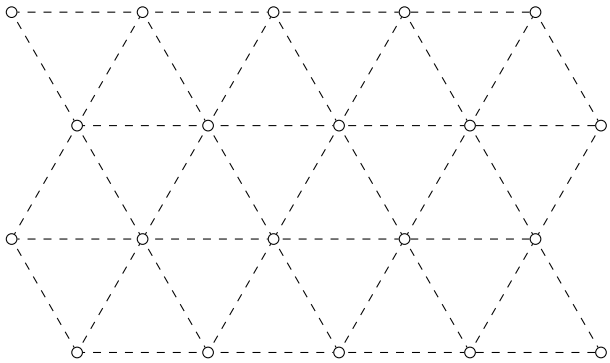


Figure 3-5: Rectangularly shaped T-graph of size 5×4. The edges represent possible paths of length $\sqrt{3}r$

In case a graph has only edges of the same length, the path that is the solution to the traveling salesman problem is called a Hamiltonian cycle. Moreover, since a Hamiltonian cycle is always a simple cycle, the start vertex can be chosen arbitrarily from all the vertices in the graph.

The *Hamiltonian Theorem* [11] states that every finite, 2-connected, linearly convex T-graph

has a Hamiltonian cycle. There is only one exception which is a D-graph that is shown in Figure 3-6).

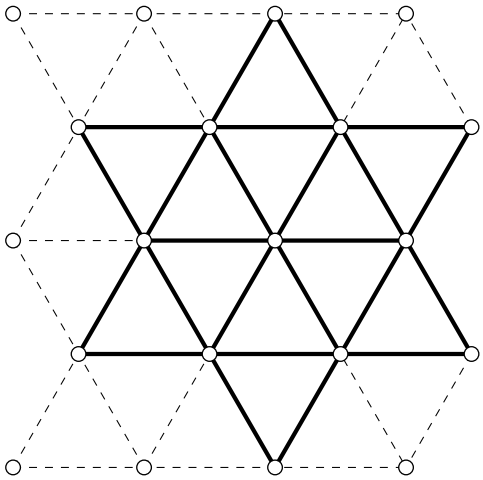


Figure 3-6: The D-graph is the only 2-connected T-graph that has not one possible Hamiltonian Cycle.

From the above Hamiltonian Theoremcan be concluded that every rectangularly shaped T-graph of size $A \times B$ where $A \geq 2$ and $B \geq 2$ has a Hamiltonian cycle, since such a graph is always 2-connected, linearly convex, and not isomorphic to the D-graph 3-6. In other words, a perfect path of length $N\sqrt{3}r$ exists that is the shortest possible path to connect all vertices. However, this insight only provides the proof that there exists an optimal path, but it does not give a solution to the problem.

3-3-2 Methods

Single Robot

In order to come up with a heuristic method that can find the Hamiltonian subgraph of any rectangular T-graph, the regularity in the structure of the graph is exploited. One solution to construct a subgraph that is the Hamiltonian cycle for any possible rectangular T-graph of size $A \times B$, where $A \geq 2$ and $B \geq 2$, is given by the following heuristic:

- IF A is even AND B is even THEN
  - Construct a graph of size $2 \times B$ as displayed in Figure 3-7a.
  - Extend the above graph to size $A \times B$ as displayed in Figure 3-7b.
- IF A is even AND B is odd THEN
  - Construct a graph of size $3 \times B$ as displayed in Figure 3-7c.
  - Extend the above graph to size $A \times B$ as displayed in Figure 3-7d.
- IF A is odd AND B is even THEN

  - Construct a graph of size $2 \times B$ as displayed in Figure 3-7e.
  - Extend the above graph to size $A \times B$ as displayed in Figure 3-7f.

- IF A is odd AND B is odd THEN
  - Construct a graph of size $3 \times B$ as displayed in Figure 3-7g.
  - Extend the above graph to size $A \times B$ as displayed in Figure 3-7h.

This heuristic yields a subgraph that is a Hamiltonian Cycle for any rectangular T-graph of size $A \times B$, where $A \geq 2$ and $B \geq 2$. So for the single robot coverage problem this gives one solution to the shortest path trough a set of waypoints that is yielded from a hexagonal grid decomposition.

Multi-Robot

The multi-robot problem is an extension of the single robot problem. Where the single robot case could be compared to a traveling salesman problem, the multi-robot case can be compared with the multi-traveling salesmen problem. So the problem is now extended to finding M Hamiltonian cycles, one for each of the M UAVs, that together visit every vertex in the rectangular T-graph only once. This rectangular T-graph is the same graph that was already defined for the single robot case.

Where the single robot problem has a clear solution under the given assumptions, the multi-robot version is not so straightforward. If the case is considered where the start point of every individual UAV can be chosen freely, the problem reduces to M single robot problems, where M is the number of UAVs. This decomposition can be made when we assume that the full graph can be split in M isomorphic subgraphs that satisfy $A \geq 2$ and $B \geq 2$.

However, since in the practical case of FireSwarm all UAVs are likely to be deployed from the same place, the constraint of a collective starting point will be introduced. So all UAVs start and end their paths in the same vertex in the lower left corner at position $W_{\text{hexa}}(1, 1)$. (This is often referred to as a Multi Traveling Sales Men problem with single depot.)

So this means that there is no longer a perfect Hamiltonian cycle since the first vertex will already be visited at least twice by all of the M UAVs. Nevertheless three different heuristic methods are proposed to solve the multi-robot scenario with a common starting point. The methods are named: the Boxes method, the Flower method, and the Lanes method. All methods were chosen to be efficient, in terms of path length, and to be able to cover all vertices in the graph at least once, and also start and return in the same vertex at $W_{\text{hexa}}(1, 1)$.

In order to compare the different methods the total path length of all UAVs together is expressed by the number of edges $E_{\text{tot}}(M, N)$, which is a function of M, the number of UAVs, and N, the total number of vertices in the graph. Furthermore, for simplicity, the dimensions of the search area that is to be covered by the different methods is assumed to be square and $l_x = l_y = 1$.

Boxes

The first method is the Boxes method. This method decomposes the square search area in M subareas of equal dimensions, where M is the number of UAVs, see Figure 3-8.

The width of each subarea is $c = \frac{1}{M}$ in the x-direction and 1 in the y-direction. Each of these subareas translates into a T-graph that has A_c vertices in the x-direction:

$$A_{\mathrm{c}} = \frac{A}{M} = \frac{1}{d_x M} = \frac{1}{\sqrt{3} r M} \tag{3-19}$$

and B_{c} of vertices in the y-direction:

$$B_{\mathrm{c}} = B = \frac{1}{d_y} = \frac{1}{\frac{3}{2} r} \tag{3-20}$$

The total number of way-points can be expressed by:

$$N = AB = \frac{1}{\sqrt{3}\frac{3}{2} r^2} \tag{3-21}$$

The value for the radius of the sensor is thus given by:

$$r = \sqrt{\frac{1}{\sqrt{3}\frac{3}{2} N}} = \frac{1}{\sqrt[4]{3}\sqrt{\frac{3}{2}}\sqrt{N}} \tag{3-22}$$

$$\tag{3-23}$$

Combining (3-19) and (3-20) with (3-21) yields:

$$A_c = \sqrt[4]{\frac{3}{4}} \frac{\sqrt{N}}{M} \tag{3-24}$$

$$B_c = \sqrt[4]{\frac{4}{3}} \sqrt{N} \tag{3-25}$$

So after decomposition the original graph is divided into M subgraphs, which are all rectangular T-graphs that can be optimally covered by each of the M UAVs.

Following the Hamiltonian Theorem, each of these areas can be covered optimally as long as $A_c \geq 2$ and $B_c \geq 2$, and the solution is given by the heuristic for single robot coverage, which has been discussed in Section 3-3-2. The only wasted edges are those that form the paths

from the starting point to the closest vertex in the corresponding subareas. This loss is given by:

$$E_{B_{\text{loss}}}(N,M) = 2A_c \sum_{m=1}^{M} (m-1) \tag{3-26}$$

$$= 2\frac{1}{M}\sqrt[4]{\frac{3}{4}}\sqrt{N}\left(\frac{M^2+M}{2} - M\right) \tag{3-27}$$

$$= \sqrt[4]{\frac{3}{4}}\sqrt{N}\Big(M-1\Big) \tag{3-28}$$

So this will result in the total number of edges of the path that covers all way-points:

$$E_B(N,M) = N + E_{B_{\text{loss}}}(N,M) \tag{3-29}$$

$$= N + \sqrt[4]{\frac{3}{4}}\sqrt{N}\Big(M-1\Big) \tag{3-30}$$

It has to be noted that this equation can result in a number of edges that is not a natural number. Furthermore, it is assumed that the number of vertices in horizontal direction of the total T-graph that corresponds to the total search area divides neatly by the number of UAVs, M.

Flower

The Flower method also decomposes the square in M equally sized subareas, see Figure 3-9. The subareas are shaped as triangles that have one point, the starting point in $W_{\text{hexa}}(1,1)$, in common. In order to simplify the problem it is assumed that the number of UAVs, M, is even, such that we can assume that the decomposition of the search area below the diagonal is identical to the decomposition above the diagonal, where the diagonal is defined as the line through $(0,0)$ and $(1,1)$. In this section we will only discuss the method that covers the area that lies below the diagonal. Furthermore, we will assume that the method to cover the search area above the diagonal is identical, when it is mirrored in the diagonal, to the method to cover the bottom part.

The M subareas, in which the square is decomposed, yields T-graphs that do not necessarily have a Hamiltonian cycle. This is because in order for the Hamiltonian Theorem to hold, the graphs have to satisfy the requirement of being 2-connected. So, when the number of UAVs M increases, c will decrease, which will result in increasingly narrower triangular areas. This will result in the subareas to become too narrow such that the T-graphs that cover them will become less than 2-connected, see Figure 3-10 that shows the first part of a possible subarea close to the starting point $W_{\text{hexa}}(1,1)$, where the corresponding T-graphs becomes too narrow such that is no longer 2-connected. The red lines in Figure 3-10 are an example of the two edges that could be cut to disconnect the graph, to show that indeed the graph is not 2-connected anymore.

Figure 3-11 shows, in red, the assumed minimum size of the T-graph that has to fit between the upper and lower boundary anywhere in the subarea in order for the corresponding T-graph to be at least 2-connected. Since the vertical distance between the upper and the lower

boundary diverges, in the positive horizontal direction, a distance u can be defined that is the minimum distance between the upper and lower boundary, such that the part of the subarea that is to the right of the point of this minimum distance u will always have a corresponding T-graph that is at least 2-connected.

The distance u depends on the angles between the horizontal axis and the lines that mark lower and upper boundary of the search area. From Figure 3-9 it can be seen that the subarea with the largest angles is the one that has the diagonal of the search space as the upper boundary of the subarea. Note that only the search area below the diagonal is discussed.

So as can be seen from Figure 3-11 the distance u depends on the angle α between the lower boundary of the subarea to the horizontal axis. The maximum value of α is a function of the number of UAVs M and is given by:

$$\alpha(M) = \tan^{-1}\left(\frac{1 - \frac{1}{\frac{M}{2}}}{1}\right) \tag{3-31}$$

$$= \tan^{-1}\left(1 - \frac{2}{M}\right) \tag{3-32}$$

So now the maximum value u can be found by using the expression (3-32) as is given by:

$$u(M) = \sqrt{3}r\tan(\alpha_{\max}) + 3r \tag{3-33}$$

$$= \sqrt{3}r\left(1 - \frac{2}{M}\right) + 3r \tag{3-34}$$

$$= \sqrt{3}r - \frac{2\sqrt{3}r}{M} + 3r \tag{3-35}$$

We can reshape the subareas such that the points inside the subarea can be covered optimally. This is done by adding M equally sized triangles as is displayed in dark gray in Figure 3-12. The base and height of the triangle are equal to $u(M)$. So the number of edges needed to cover these triangles are equal to the number of vertices inside the triangular area and is given by:

$$E_{F_{\mathrm{A}}}(M) = \frac{M}{2}\frac{1}{2}\frac{u}{d_x}\frac{u}{d_y} \tag{3-36}$$

$$= \frac{M}{4}\frac{u}{\sqrt{3}}\frac{u}{\left(\frac{3}{2}\right)} \tag{3-37}$$

$$= \frac{Mu^2}{6\sqrt{3}r^2} \tag{3-38}$$

$$\tag{3-39}$$

Furthermore, to get to the start of the redefined subareas the UAVs have to travel to their individual starting points that are indicated in Figure 3-12 by white dots. Because these

paths are also T-graphs, they have a different amount of vertices in the horizontal direction than they have in the vertical direction. Therefore, the calculation consists of two parts. First the number of edges in horizontal direction, for all UAVs combined, is given by:

$$E_{F_{\mathrm{x}}}(M) = \frac{2u}{d_x} \sum_{m=1}^{\frac{M}{2}} (m-1) \tag{3-40}$$

$$= \frac{2u}{\sqrt{3}r} \left(-\frac{M}{2} + \frac{\frac{M}{2}\left(\frac{M}{2}+1\right)}{2} \right) \tag{3-41}$$

$$= \frac{u}{\sqrt{3}r} \left(\frac{M^2}{4} + \frac{M}{2} - M \right) \tag{3-42}$$

$$= \frac{u}{\sqrt{3}r} \left(\frac{M^2}{4} - \frac{M}{2} \right) \tag{3-43}$$

Secondly the number of edges in the vertical direction, for all combined UAVs, is given by:

$$E_{F_{\mathrm{y}}}(M) = 2\frac{u}{d_y} \sum_{m=1}^{\frac{M}{2}} (m-1) \tag{3-44}$$

$$= 2\frac{u}{\left(\frac{3}{2}\right)r} \sum_{m=1}^{\frac{M}{2}} (m-1) \tag{3-45}$$

$$= \frac{4u}{3r} \left(-\frac{M}{2} + \frac{\frac{M}{2}\left(\frac{M}{2}+1\right)}{2} \right) \tag{3-46}$$

$$= \frac{u}{3r} \left(\frac{M^2}{2} - M \right) \tag{3-47}$$

$$\tag{3-48}$$

So the total the number of edges that is needed to cover all N vertices by means of the Flower method is:

$$E_F(N, M) = N + 2\Big(E_{F_{\mathrm{x}}}(M) + E_{F_{\mathrm{y}}}(M) + E_{F_{\mathrm{A}}}(M) \Big) \tag{3-49}$$

$$= N + \frac{2Mu^2}{6\sqrt{3}r^2} + \frac{2u}{\sqrt{3}r}\left(\frac{M^2}{4} - \frac{M}{2} \right) + \frac{2u}{3r}\left(\frac{M^2}{2} - M \right) \tag{3-50}$$

$$= N + \left(\frac{u^2}{3\sqrt{3}r^2} - \frac{u}{\sqrt{3}r} - \frac{2u}{3r} \right) M + \left(\frac{u}{2\sqrt{3}r} + \frac{u}{3r} \right) M^2 \tag{3-51}$$

$$\tag{3-52}$$

For the Flower method it also has to be noted that the number of edges given by (3-49) can be rational. A more important note is that quickly the number of UAVs becomes so big that the subarea become so narrow that $c < u$. So in order for the equation to be accurate r needs to be sufficiently small such that $c > u$.

Lanes

The Lanes method is inspired by the single robot path. The Lanes method for the multi-robot case is just a simple extension from the method that optimally solves the single robot coverage problem. A line formation is formed and the sweep pattern is carried out by the separate UAVs. The turning at the end of a lane is done in such a way that the differences in path length between UAVs, after taking a right corner, are cancelled out by the consecutive left corner. The paths are also chosen in such a way that they do not cross each other and do not have common points. Because the UAV all start in the same point $W_{\text{hexa}}(1,1)$ also with this method there is a suboptimal area, see Figure 3-13.

The number of edges that is needed on top of the lower bound of N is given by:

$$E_{L_{\text{loss}}}(M) = M(M-1) + 2\sum_{m=1}^{M}(m-1) = 2M^2 - 2M \tag{3-53}$$

So this results in the total number of edges for this method:

$$E_L(M) = N - 2M + 2M^2 \tag{3-54}$$

What has to be noted for the Lanes method is that it is assumed that the number of vertices always needly divides by two times the number of UAVs.

3-3-3 Test Setups and Results

The additional number of edges that is needed on top of the lower bound N for each heuristic method is given in the table below:

Method	Function	
Boxes	$E_{B_{\text{loss}}}(N,M)$	$= \sqrt[4]{\frac{3}{4}}\sqrt{N}(M-1)$
Flower	$E_{F_{\text{loss}}}(M)$	$= \left(\frac{u^2}{3\sqrt{3}r^2} - \frac{u}{\sqrt{3}r} - \frac{2u}{3r}\right)M + \left(\frac{u}{2\sqrt{3}r} + \frac{u}{3r}\right)M^2$
Lanes	$E_{L_{\text{loss}}}(M)$	$= 2M^2 - 2M$

where $u = \sqrt{3}r - \frac{2\sqrt{3}r}{M} + 3r$ and the start- and end-point of the path is $W_{\text{hexa}}(1,1)$ for every UAV.

To get a better idea of the method that produces the solution that is closest to the lower bound N, a plot was made in the range of $M \in [1,100]$ and $\sqrt{N} \in [1,1000]$. For each combination of M and $\sqrt{N}$ the number of required edges is calculated for each method and compared to find the best performing method for that particular case. A pixel is plotted in the respective color of the method that needs the least amount of edges, Figure 3-14.

Furthermore, a plot is given of the ratio between the additional edges that are needed on top of the lower bound N, see Figure 3-15. In other words this ratio gives a measure of relative influence of the sub-optimally covered areas on the total path length. For each combination of $\sqrt{N}$ and M the best performing method was used.

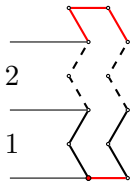
3-4 Conclusions

The problem of search with a single UAV with perfect deterministic sensors with a circular sensor area was decomposed in a static coverage problem and a dynamic path planning problem. The static coverage problem can be solved optimally by using a hexagonal grid heuristic for the location of the observation points. The lower bound of the length of the path was found to be $\sqrt{3}M$ where M is the number of UAVs. The heuristic method for the path planning problem was shown to reach the lower bound.

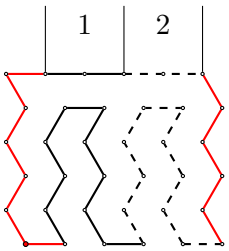
The extension of the single UAV search problem to the multiple UAV case used the same optimal set of observation points that resulted after the first decomposition of the single UAV case. Furthermore, the additional constraint of a collective start and end point for all UAVs was introduced which makes it impossible to reach the same lower bound as for the single UAV case.

The methods proposed in this thesis to solve the search problem with multiple UAV with deterministic sensors were compared in terms of path length. It can be concluded that the performance of the Flower method is near-optimal when the number of observation points is very high compared to the number of UAVs. The Boxes method seems to perform best when the number of UAV becomes close to the root of the total number of observation points. However, these results have to be seen has indications due to the noted simplifications that were applied. A way to truly verify the results is by using algorithms to carry out each method exactly, although this is not expected to change the general conclusions drawn here.

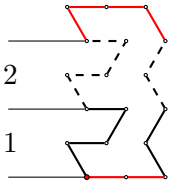
So this shows that it is possible to come up with heuristic methods for planning paths that can cover a rectangular search area close to optimality. In other words this means that an almost perfect search path can be determined a priori when the sensors of the UAVs are considered as perfect and deterministic.



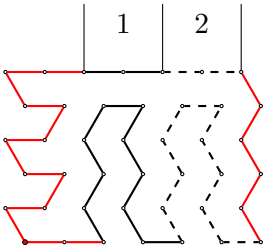
(a) Vertical extension for grids where A and B are both even



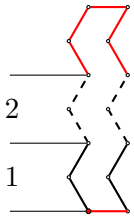
(b) Horizontal extension for grids where A and B are both even



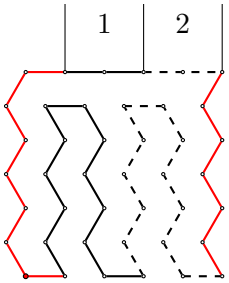
(c) Vertical extension for grids where A is odd and B is even



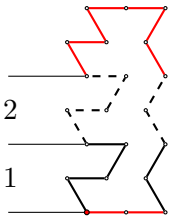
(d) Horizontal extension for grids where A is odd and B is even



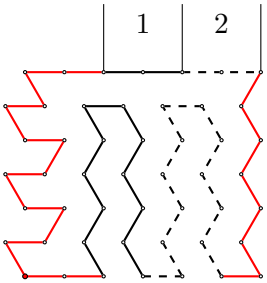
(e) Vertical extension for grids where A is even and B is odd



(f) Horizontal extension for grids where A is even and B is odd



(g) Vertical extension for grids where A and B are both odd



(h) Horizontal extension for grids where A and B are both odd

Figure 3-7: The four different cases and their methods of extending them in the vertical direction and horizontal direction, where the black solid lines and black dashed lines show the repetitive structure in the graph. For all four cases two out of an infinite number of identical extensions are shown in vertical and horizontal direction, with the first extension depicted by the graph with solid black lines and the second extensions with the dashed black lines. Furthermore, the extensions are also shown by the numbers 1 and 2, respectively for the first and second extension.The conditions per method, in terms of the size of the hexagonal grid (A and B), are given in the caption of each subfigure.

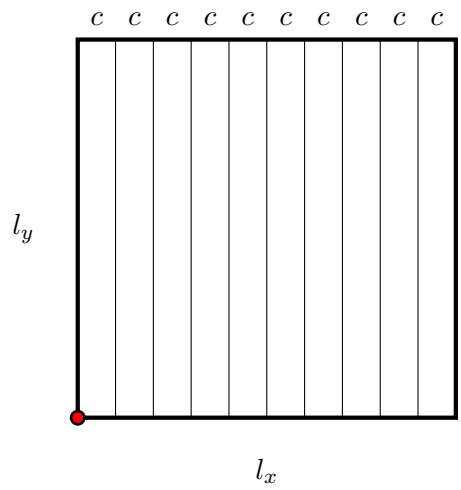


Figure 3-8: The Boxes decomposition of a, $l_x = 1$ by $l_y = 1$, square in M equally sized rectangles with $c = \frac{1}{M}$. Every rectangle is assigned to one of the M UAVs

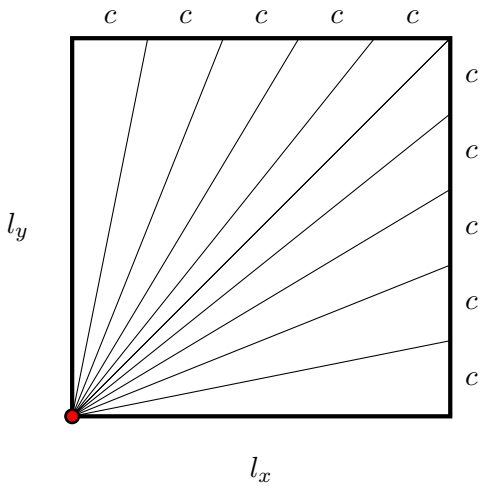


Figure 3-9: The Flower decomposition of a, $l_x = 1$ by $l_y = 1$, square in M equally sized triangles. $c = \frac{1}{M}$. Every triangle is assigned to one of the M UAVs, where $M \in \{2, 4, 6, \cdots, 2k\}, k \in \mathbb{N}^+$

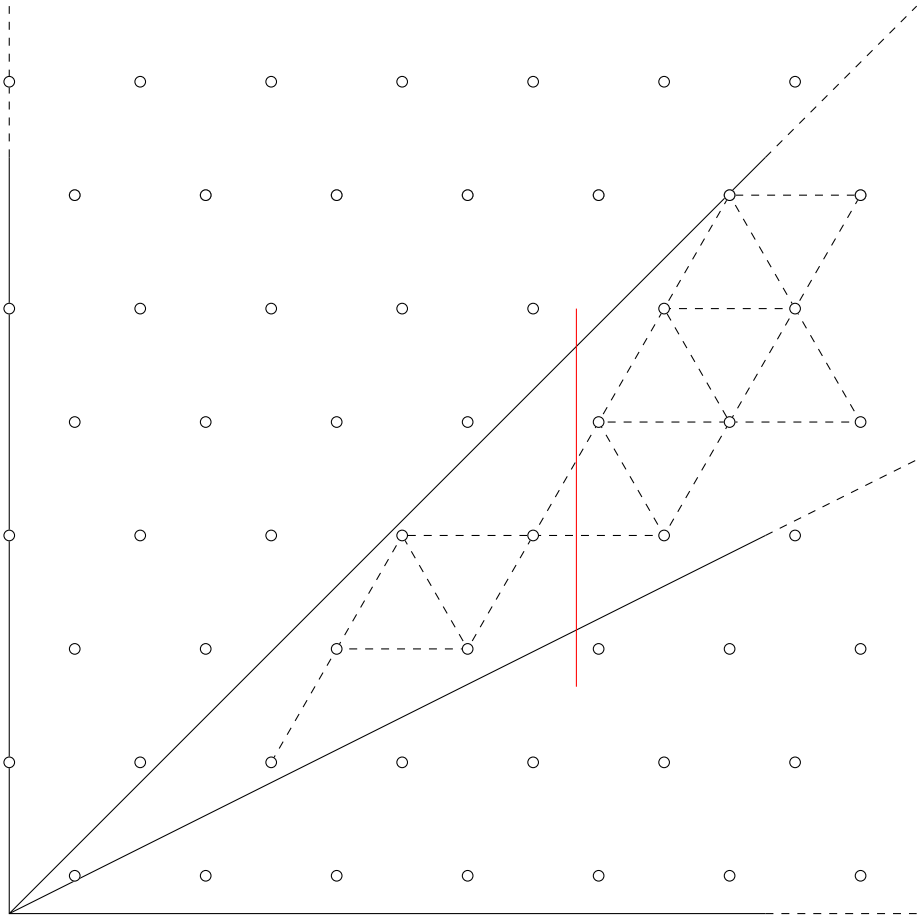


Figure 3-10: A part of a possible subarea is shown were the corresponding T-graph is less than 2-connected as can be shown by the removal of the two edges by cutting along the red line.

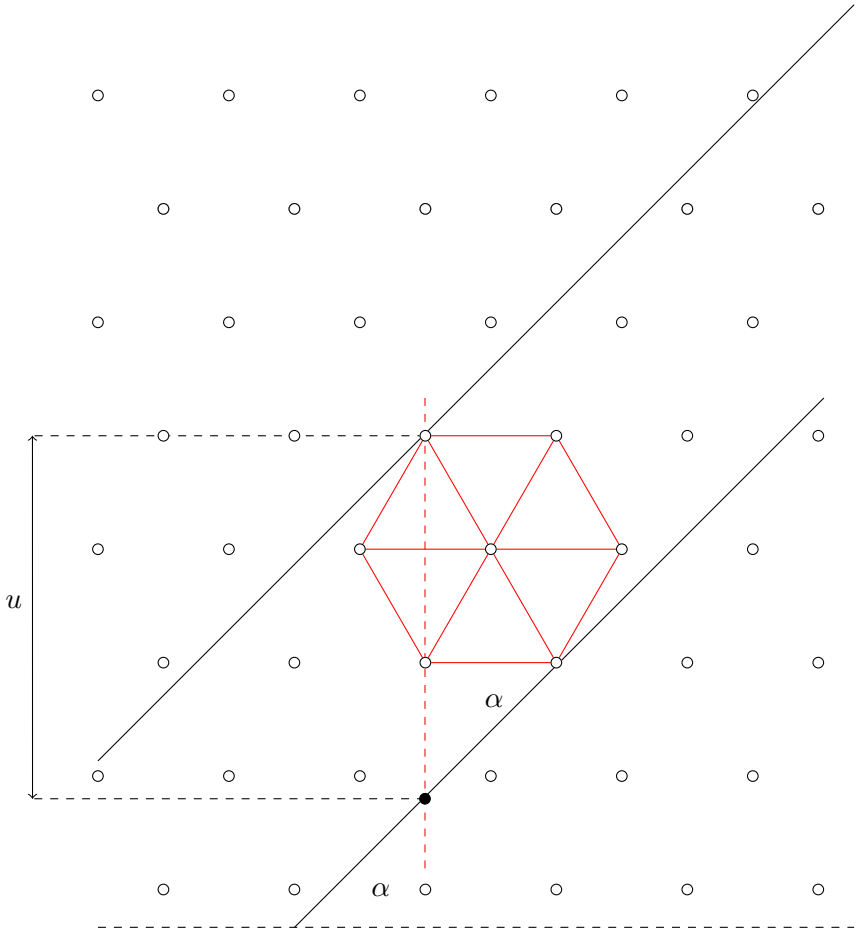


Figure 3-11: The graph depicted in red shows the assumed minimum size of the T-graph that ensures that the part of the subgraph that is to the right of the red dashed line is at least 2-connected. The distance u shows the minimum distance between the upper and lower boundary of the subarea such that it can still fit the red graph. The angle α is the angle between the horizontal axis and the lower black line that marks the lower boundary of the subarea.

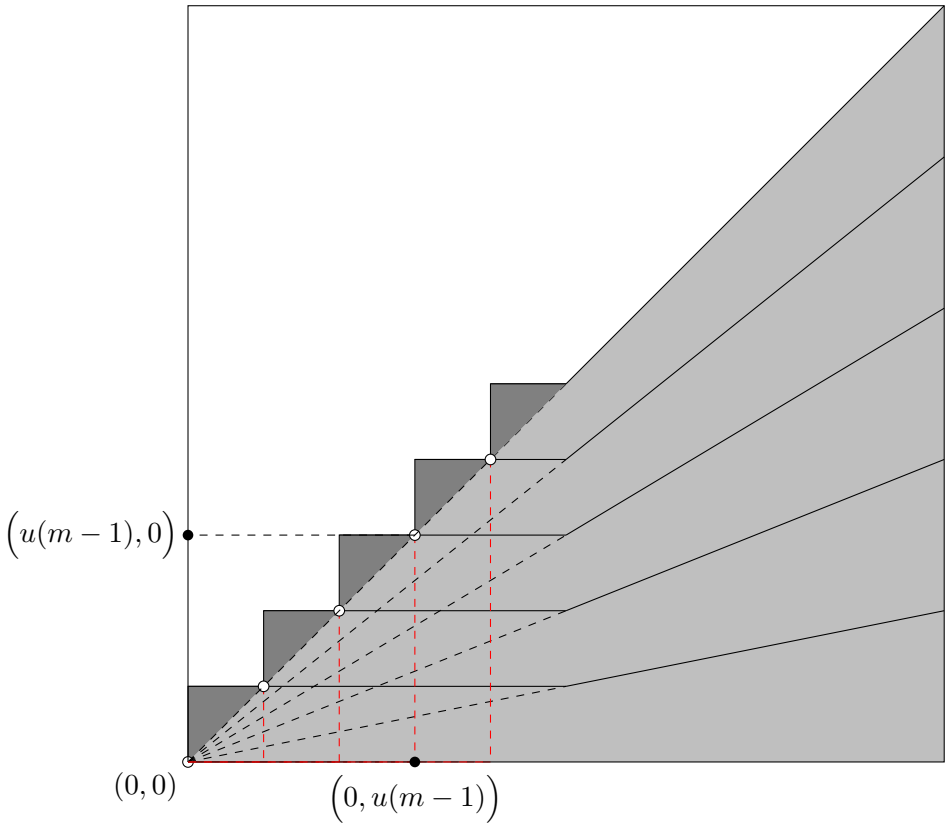


Figure 3-12: The decomposition of the bottom half of the 1 by 1 square in $\frac{M}{2}$ areas (light gray). The triangles (dark gray) indicate areas wherein the vertices are covered twice. The white dots indicate the start and end point of the individual areas for each of the $\frac{M}{2}$ UAV. The dashed (red) lines indicate the paths that are needed to travel between the collective starting point in $W_{\text{hexa}}(1,1)$ to the individual start points in $\Big(u(m-1), u(m-1)\Big)$, where $m \in \{1, \cdots, \frac{M}{2}\}$

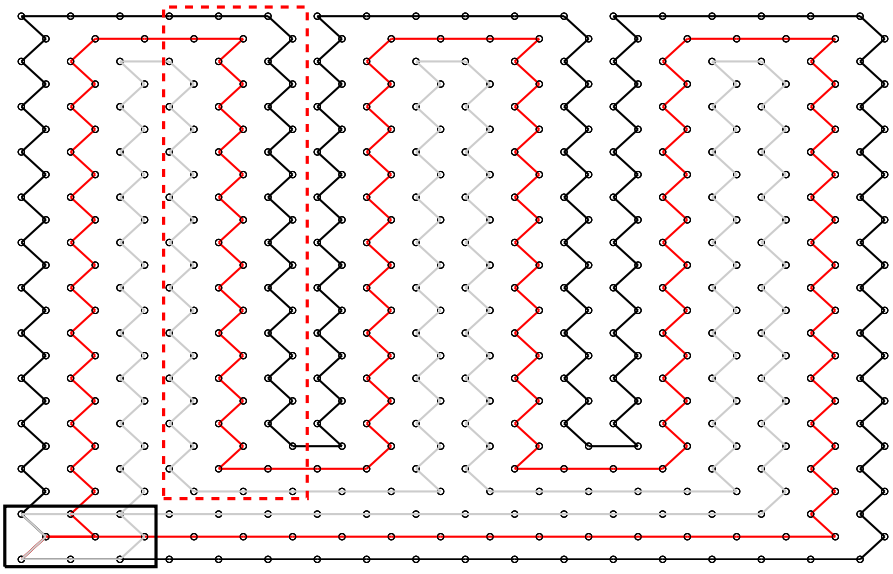


Figure 3-13: This figure displays the paths of $M = 3$ UAV (one color for any of the UAV) that cover a total of $N = 450$ points. The Lanes method uses an sweep path with multiple UAV. It can be seen that the path length of a left turn that is followed by a right turn and vice-versa is of the same length for all UAV (dashed box). The area marked with the solid box is the sub optimally covered area, which results in $Nl(M) = 2M^2 - 2M$ extra edges that are needed on top of the lower bound of N edges.

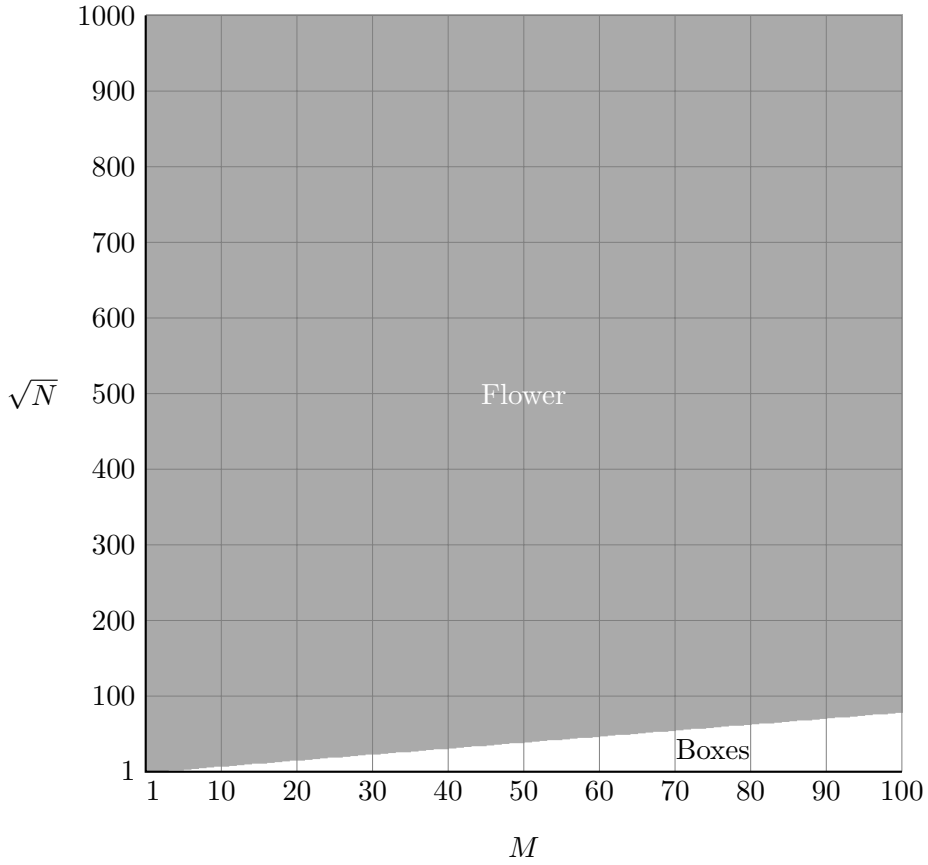


Figure 3-14: Overview of the best methods in range $M = [0, 100], \sqrt{N} = [0, 1000]$.The areas where Flowers, and Boxes method perform best are displayed by the colors white and gray respectively. Note that the Lanes method did not perform best at any of the tested combinations of M and $\sqrt{N}$.

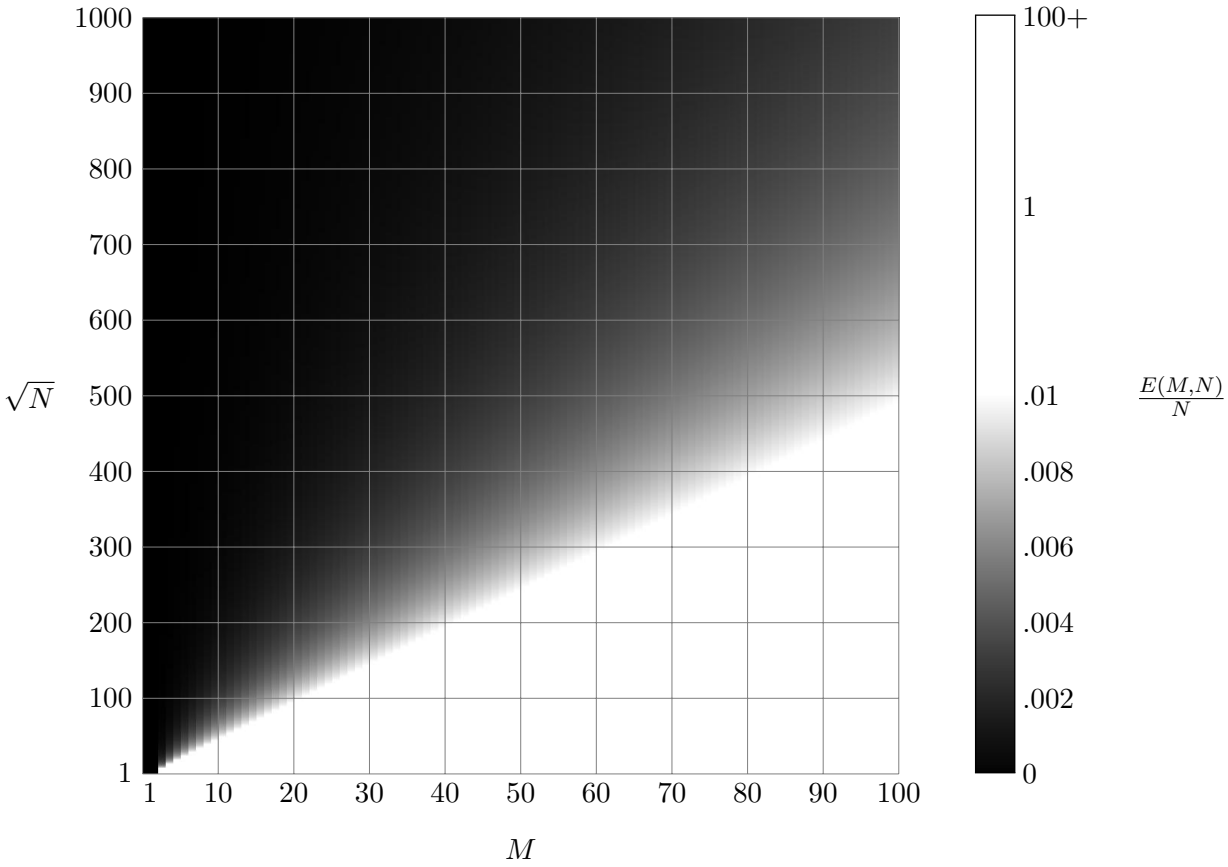


Figure 3-15: Overview of the ratio $\frac{E_{\text{loss}}(M,N)}{N}$. This is the ratio between the additional edges and the lower bound on the number of edges for that particular number of points N, the darker the color the lower the ratio. Black indicates near optimum performance. Notice that for each combination of $\sqrt{N}$ and M the best performing heuristic method was used.

Chapter 4

Search (Stochastic)

4-1 Introduction

In Chapter 3 the deterministic case has been discussed where the sensor reading is always perfect and coverage is the optimal strategy. In this chapter the case will be discussed where a sensor reading is described by a stochastic process. This is done by first giving a problem description of the stochastic problem. Secondly, a framework is proposed that can run the five different algorithms that are also proposed in this thesis. Thirdly, a number of test setups is considered to study the effect of some important parameters and their effect on the different proposed algorithms. The results of the tested algorithms are discussed and this chapter is closed by the conclusions.

4-2 Problem Description

In this section the problem description for the stochastic case will be given by three aspects of the problem. First the position control by waypoint navigation will be formulated. Secondly, the model will be given that is used to sample at those waypoints, thirdly and lastly, it is explained how the hypotheses are tested. These three aspects of the problem are depicted in Figure 4-1.

4-2-1 Navigation

The search area is divided into $A \times B$ grid cells. All valid positions of the Unmanned Aerial Vehicle (UAV) are given by the following points:

$$W_{(a,b)} = \left(\frac{(a - \frac{1}{2})l_x}{A}, \frac{(b - \frac{1}{2})l_y}{B} \right) \in \mathbb{R}^2 \tag{4-1}$$

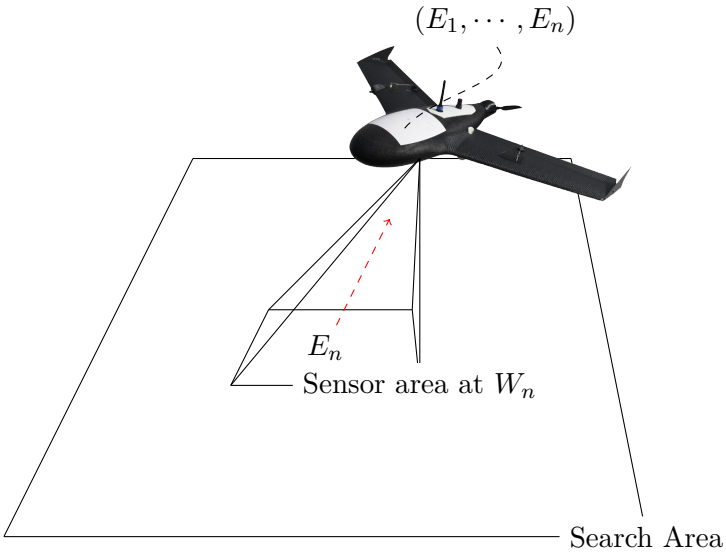


Figure 4-1: A schematic representation of the situation where evidence E_n is collected at some sensor area belonging to waypoint W_n inside the search area. The evidence is stored onboard of the UAV, where n indicates the waypoint index.

for $a \in \{1, \cdots, A\}$ and $b \in \{1, \cdots, B\}$ where l_x and l_y are the width and height of the rectangular 2-dimensional search area. This means that the only coordinate at which the UAV is allowed to plan a waypoint is at one of the coordinates that is given by (4-1), which correspond to the centers of the $A \times B$ cells. In Figure 4-2 a representation can be found for a search area with 5×5 grid cells, where, as an example, three valid waypoints are plotted inside the search area.

For the sake of simplicity the dynamics of the UAV are simplified by assuming that the velocity of the airplane is constant and the path between two waypoints is always a straight line except for the case were the UAV plans the next waypoint in the same location. In this case the UAV takes a circular path with the same constant velocity v, when the path of the circle has a constant length of d_{circle}.

Furthermore, we assume the sensed area to be rectangular instead of circular. This allows for easier representation and simulation of the problem. The dimensions of the grid cells are chosen in such a way that the grid cell coincides with the sensed area in case the UAV is located at one of the valid waypoints, at the center of the grid cell.

4-2-2 Sampling

A sample that is taken by the sensor from one of the valid waypoints given by (4-1) can either be positive or negative. A positive sample indicates evidence that a fire is present inside the waypoint its corresponding cell, whereas a negative sample indicates no fire inside this cell. The stochastic process that generates the evidence can be described by:

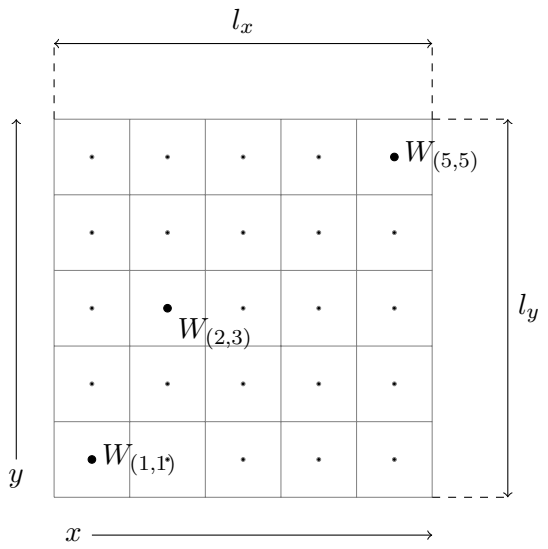


Figure 4-2: The grid world representation of the search space where $A = B = 5$. All valid waypoints according to (4-1) are marked with grey dots. As examples the positions of three valid waypoints $W_{(1,1)} = (\frac{1}{10}l_x, \frac{1}{10}l_y)$, $W_{(2,3)} = (\frac{3}{10}l_x, \frac{1}{2}l_y)$, and $W_{(A,B)} = (\frac{9}{10}l_x, \frac{9}{10}l_y)$ are marked with thick black dots. From the center of each square grid cell the occupancy of that cell can be sampled according to the probabilities given in (4-2)

$$P(E = \text{Positive}|H = \text{Fire})_{W_n} = P_{\text{TP}} \tag{4-2}$$
$$P(E = \text{Negative}|H = \text{Fire})_{W_n} = P_{\text{FN}} = 1 - P_{\text{TP}} \tag{4-3}$$
$$P(E = \text{Positive}|H = \text{No fire})_{W_n} = P_{\text{FP}} \tag{4-4}$$
$$P(E = \text{Negative}|H = \text{No fire})_{W_n} = P_{\text{TN}} = 1 - P_{\text{FP}} \tag{4-5}$$

where $E_n \in \{\text{Positive, Negative}\}$ indicates the evidence collected at W_n, which is the nth waypoint that is visited. The above probabilities are, respectively, the probability of a true positive (TP), a false negative (FN), a false positive (FP), and a true negative (TN).

So the collection of evidence at the current waypoint W_n is given by a standard straightforward sample from a uniform distribution:

$$E_n(a,b) = \begin{cases} \text{Empty} & \text{if } W_n \text{ is not inside cell } (a,b) \\ \text{Positive} & \text{else if } (r_n \leq P_{\text{TP}} \text{ AND } W_n = W_{\text{fire}}) \text{ OR } (r_n \leq P_{\text{FP}} \text{ AND } W_n \neq W_{\text{fire}}) \\ \text{Negative} & \text{else} \end{cases} \tag{4-6}$$

where W_{fire} is the waypoint that corresponds to the center of the grid cell that actually contains a fire and r_n is a uniformly distributed random value between 0 and 1 generated at the time of arrival of the UAV at waypoint W_n.

The storage of evidence on board of the UAV is defined by:

$$\mathcal{E}_n(a,b) = (E_1(a,b), \cdots, E_n(a,b)) \tag{4-7}$$

which means that every time new evidence is collected at the nth waypoint the whole history of evidence $\mathcal{E}_n(a,b)$ for any cell in the grid, so for all $a \in \{1, \cdots, A\}$ and all $b \in \{1, \cdots, B\}$, is accessible by the UAV.

4-2-3 Testing of hypotheses

Hypothesis testing is used to model the probability to find fire in a certain grid cell. The hypothesis that the grid cell that corresponds to the current valid waypoint is occupied by a fire, is tested by basic Bayesian inference:

$$P(H|E) = \frac{P(E|H)P(H)}{P(E)} \tag{4-8}$$

where H is the hypothesis and E is the evidence. The probability $P(H|E)$ is the posterior and $P(H)$ is the prior. The calculation of the posterior can be done by the recursive form of the Bayesian inference rule:

$$P(H|E_n) = \frac{P(E_0|H)}{P(E_n)} P(H|E_{n-1}) \tag{4-9}$$

When the expression (4-7) for the collected evidence is put in (4-10) the following expression is yielded that gives the probability of the hypothesis that a fire is present in cell (a,b) when the UAV has just measured at the nth waypoint in the path:

$$P(H|\mathcal{E}_n)_{a,b} = \frac{P(\mathcal{E}_0|H)}{P(\mathcal{E}_n)} P(H|\mathcal{E}_{n-1})_{a,b} \tag{4-10}$$

Here, we use the prior $P(H|\mathcal{E}_{n-1})_{a,b}$, which is the probability of cell (a,b) being occupied by a fire given previously collected evidence at cell (a,b) and we multiply it by the factor $\frac{P(E_0|H)}{P(E_n)}$. The outcomes of the hypotheses that were generated by the update rule in (4-10) are listed in the occupancy grid matrix O.

The occupancy grid O has the same size $A \times B$ as the waypoint grid and lists for every cell $O_{a,b}(n)$ the outcome of the tested hypothesis that the cell corresponding to waypoint $W_{(a,b)}$ is occupied with fire given all the evidence $\mathcal{E}_n$ collected up to the moment the nth waypoint is visited. So $O_{a,b}(n)$ is given by:

$$O_{a,b}(n) = P(H = \text{fire}|\mathcal{E}_n)_{a,b} \tag{4-11}$$

4-2-4 Objective

The objective will be to find a strategy that will give the path $\mathcal{W} = (W_1, \cdots, W_N)$ that contains a tuple of waypoints such that:

$$\text{minimize} \quad \sum_{n=2}^{N} \|W_n - W_{n-1}\| \frac{1}{v} \tag{4-12}$$

$$\text{subject to } O_{W_{\text{fire}}} \geq p_{\text{sure}} \tag{4-13}$$

where all waypoints in $\mathcal{W}$ are valid given by (4-1), $W_1 = W_{(1,1)}$, v is the constant velocity of the UAV, and $\|W_n - W_{n-1}\|$ indicates the 2-norm or in other words the distance between two points in 2-dimensional space. The occupancy value $O_{W_{\text{fire}}}$ is the probability of the hypothesis being true that the actual occupancy grid cell W_{fire} is occupied with a fire, given the accumulated evidence collected at all the waypoints in the path up to waypoint W_n. The probability p_{sure} represents the required certainty for, which it is assumed that a fire is detected.

4-3 Algorithms

So now that the model of the UAV is defined different strategies can be tested. The framework for one UAV is given by:

1. Start in the corner and set $n = 1$, such that $W_1 = (1, 1)$.
2. Chose the next coordinate, thus increment n, based on one of the strategies: Psychic, Random, Coverage, Greedy, or Planned, where the values for the tested hypothesis can be used as input (see Sections 4-3-1, 4-3-2, 4-3-3, 4-3-4, and, 4-3-5).
3. Go to the next coordinate of the path with constant velocity v.
4. Sample the current location and test the hypotheses by basic Bayesian inference of all cells and update the occupancy matrix $O_{W_n}(n)$.
5. Go back to (2) if no value of matrix $O(n) \geq p_{\text{sure}}$.

In this thesis the problem of robotic search with stochastic sensors and multiple UAVs is only tested briefly. Therefore, the implementation with multiple UAVs will be done by a simple extension of the single UAV framework by executing them for each UAV simultaneously. In order to add some basic cooperation between UAVs, the sampled evidence at step 4 is not only added to the UAVs its own memory but also to all UAVs in the swarm.

Note that the position of the fire is modeled by the occupancy of the area that corresponds to the waypoint W_{fire} that is also a valid waypoint. Furthermore, the path $\mathcal{W}$ that is calculated by this algorithm will contain N waypoints, where the number of waypoints is not fixed nor known a priori.

4-3-1 Psychic

A way to find the lower bound of the problem described in Section 4-2, is to assume W_{fire} is known and go and stay there until enough evidence is collected such that $O_{W_{\text{fire}}} \geq p_{\text{sure}}$.

So the Psychic algorithm is described by:

$$W_n = W_{\text{fire}} \tag{4-14}$$

4-3-2 Random

The Random algorithm creates a random walk by randomly selecting one out of four valid actions: move one step to the North, South, East, or West.

A definition of the Random algorithm is given by:

$$W_{n+1} = W_n + \Delta_n \tag{4-15}$$

where Δ represents the displacement from the current position to the next position. The definition of Δ is given by:

$$\Delta_n = \begin{cases} (0,1) & \text{if rand}(1,4) = 1 \\ (0,-1) & ,, \quad ,, \quad = 2 \\ (1,0) & ,, \quad ,, \quad = 3 \\ (-1,0) & ,, \quad ,, \quad = 4 \end{cases} \tag{4-16}$$

where $\text{rand}(R_1, R_2)$, with R_1 and R_2 integers, draws a random, uniformly distributed integer value from the set $\{R_1, R_1 + 1, \cdots, R_2\}$ at time n. To ensure that no coordinate is selected that is outside the search area the following rule is applied: when W_{n+1} does not satisfy (4-1) $W_{n+1} = W_n$.

4-3-3 Coverage

The Coverage algorithm returns a preprogrammed path that results in a ox-like walk, see Section 2-3. An example of this path is given by Figure 4-3.

$$W_n = W_{\text{coverage}}(n) \tag{4-17}$$

where $W_{\text{coverage}}(n)$ is always a valid waypoint.

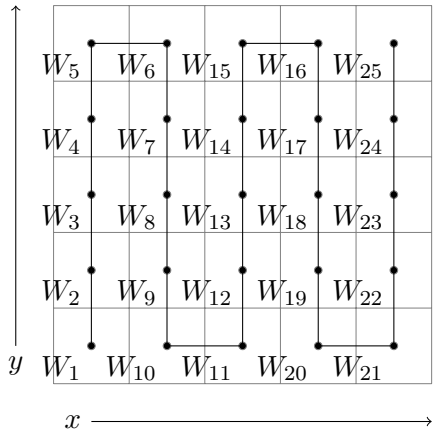


Figure 4-3: An example of an oxlike coverage path on a $A \times B$ grid where $A = 5$, $B = 5$ grid. When the last waypoint is reached the path is returned in reverse, back to the first waypoint, where the path is repeated.

4-3-4 Greedy

The Greedy algorithm returns the waypoint that maximizes the fitness function:

$$f_{\text{fitness}}(W) = \frac{l_{\max} - \|W - W_{n-1}\|}{l_{\max}} \cdot O_W \tag{4-18}$$

where $l_{\max} = \sqrt{l_x^2 + l_y^2}$. This fitness function is a quantification of how fit, or good, a possible next waypoint is. In this case the fitness is determined by the product of two parts. The first part is a normalized distance between the current waypoint and the next waypoint. The second part is the probability of the grid cell that corresponds to the next waypoint to occupy a fire. So now the Greedy algorithm finds the best next position by optimizing:

$$W_n = \arg\max_W \left\{ \frac{l_{max} - \|W - W_{n-1}\|}{l_{\max}} \cdot O_W \middle| O_W \leq p_{\text{sure}} \right\} \tag{4-19}$$

where condition $O_W \leq p_{\text{sure}}$ ensures that a cell occupied by fire that is already found is excluded from the search. Note that the optimization can be carried out by simply calculating the fitness for any of the valid waypoints, since the number of possibilities is only $A \cdot B$.

4-3-5 Planned

The Planned strategy is similar to the Greedy strategy. However, the Planned algorithm considers and plans more than only one waypoint in advance. An example of a possible planned path is given in Figure 4-4. It can be seen in Figure 4-4 that waypoints that form

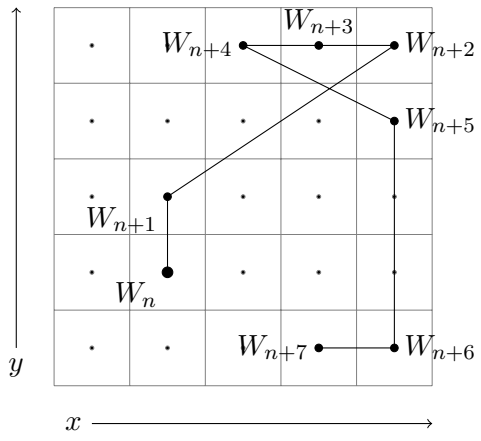


Figure 4-4: An example of a planned path of length 7 on a $A \times B$ grid where $A = 5$, $B = 5$ grid.

the path can be planned anywhere in the search area as long as it is a valid waypoint as is defined earlier in (4-1).

Similarly as for the Greedy strategy the fitness function for the Planned strategy calculates a fitness value by taking the normalized product of the length of the planned path and the sum of the values in the occupancy grid that correspond to the waypoints in the path. The fitness function is given by:

$$f_{\text{fitness}}\Big((W_n, W_{n+1}, \cdots, W_{n+L})\Big) \tag{4-20}$$

$$= \frac{l_{\max} \cdot L - \sum_{l=1}^{L} \|W_{n+l} - W_{n+l-1}\|}{l_{\max} \cdot L} \cdot \frac{\sum_{l=1}^{L} O_{W_{n+l}}(n)}{L} \tag{4-21}$$

where W_n is the current waypoint and $(W_{n+1}, \cdots, W_{n+L})$ is the tuple of waypoints that form the path of length L that is planned when the UAV is at W_n.

Although we now have defined a fitness function that quantifies how good a path of length L is, finding this path amongst the large number of possible paths is difficult. This is because the number of possible paths, even when the set of possible waypoints is finite, quickly explodes with a larger number of planned waypoints.

In order to find the best path without having to calculate the fitness of every possible path a Particle Swarm Optimization (PSO) approach is implemented. The PSO algorithm uses standard position and velocity update rules [12] for the control of the particle swarm in the solution space:

$$v_{i,d}(k+1) = w v_{i,d}(k) + c_1 r_1(k)\Big(p_{\text{p}_{i,d}} - x_{i,d}(k)\Big) + c_2 r_2(k)\Big(p_{\text{g}_d} - x_{i,d}(k)\Big) \tag{4-22}$$

$$x_{i,d}(k+1) = x_{i,d}(k) + v_{i,d}(k+1) \tag{4-23}$$

Here, $v_{i,d}(k)$ and $x_{i,d}(k)$ are the velocity and the position of particle $i \in \{1, \cdots, M\}$ in dimension $d \in \{1, \cdots, D\}$ at iteration step k. The symbols w, c_1 and c_2 are constants, r_1 and

r_2 are uniformly distributed random numbers between 0 and 1. The particle position $p_{\mathrm{p}_{i,d}}$ is the best position found so far by the particle. The global best position p_{g_d} is the position that has the best solution found so far by any of the particles in the swarm. The best position is the position where the highest fitness is achieved according to (4-20), the personal and global best position are updated at every iteration step.

The positions of the particle in the particle swarm are positions in the solution space of the fitness function. This allows for the representation of all the separate waypoints that form the path of the UAV in the real world, by a single point in the D-dimensional solution space, where $D = 2L$. The representation of the variables of the fitness function that are represented by the position of the particle is defined by:

$$f\Big(x_{i,1}, \cdots, x_{i,D}\Big) = f\Big(W_{n+1}, \cdots, W_{n+L}\Big) \tag{4-24}$$

So by this definition of the PSO algorithm a path can be optimized of L waypoints from the current waypoint W_n of the UAV given the collection of evidence at that time. Since discrete waypoint navigation is used to navigate the UAVs and the PSO defined here operates in a continuous solution space the waypoints in the path expressed by the particles in the swarm are rounded towards the nearest valid waypoint. This is done every time just before the fitness value is evaluated and also before determining the final solution that is returned by the PSO algorithm. So the path that is returned by the PSO algorithm is the global best position that is found, which is then, for each of the waypoints in the path, rounded towards the nearest valid waypoint.

4-3-6 Addition: Future Evidence Estimation

When each individual UAV in a swarm with multiple UAVs calculates its own next waypoint or waypoints (in case of the planned algorithm) it can happen that already another UAV has planned to visit that point. Even more a similar effect could also happen to a single UAV that carries out the planned algorithm. It could plan a path that visits one waypoint more than once.

In both of the above cases the assumed probability of fire is most likely going to change before it is visited as planned. The problem that arises from this is that the points that have a relatively high probability for fire compared to other cells will be planned multiple times by some algorithms, such as Greedy or Planned, even when the probability is still very low in an absolute sense.

So in order to correct for this false value of probability, the future probability should be estimated. The suggested approach in this thesis is to estimate the most likely evidence that will be collected in the future based on planned waypoints of other UAVs or other points in the path of one single UAV.

This estimated evidence $\mathcal{E}'$ is added to the real evidence $\mathcal{E}$ and is used in the fitness function to calculate a new occupancy matrix O' that depends on the planned waypoints of all the UAVs in the swarm. So when the fitness is calculated the algorithm will now use a different occupancy matrix O' that depends on the waypoints that will be visited in the future. However, since

this evidence is to be collected in the future it can only be estimated, the estimated future evidence $E_l^{'}(a,b)$ is given by:

$$E^{'}_{k_{(a,b)}}(a,b) = \begin{cases} \text{Positive} & \text{if } k_{(a,b)} \leq \text{round}(O_{W_n} \cdot K_{(a,b)}) \\ \text{Negative} & \text{else} \end{cases} \tag{4-25}$$

where $k_{(a,b)} \in \{1, \cdots, K_{(a,b)}\}$ and $K_{(a,b)}$ is the total number of visits at cell (a,b) when all the planned waypoints of all other UAVs that are planned at the moment that the waypoint W_n is visited, would be carried out. The total tuple of evidence that is estimated to be collected in cell (a,b) in the future is given by:

$$\mathcal{E}^{'}(a,b) = \left(E_1^{'}(a,b), \cdots, E^{'}_{K_{(a,b)}}(a,b) \right) \tag{4-26}$$

So now we have an estimate of what the most likely evidence is that will be collected over some given future. In the same way as the normal occupancy matrix O is used we can use the alternative occupancy matrix $O^{'}$ that depends on estimated future evidence to plan waypoints. Note that the addition of future evidence estimation that is described in this section is only meant as an addition to the greedy algorithm in case of multiple UAVs or for the planned algorithm with both single and multiple UAVs.

4-4 Test Setup and Results

In order to find out what the influence is of different parameters, a number of experimental scenarios was implemented. In this section the influence of the following parameters is tested:

- Sensor Probabilities: $P_{\text{TP}}, P_{\text{FP}}$
- Initial Fire Probability: $P(H|\mathcal{E}_0)$
- Search Area (Grid Size): $A \times B$
- Number of UAVs: M
- Number of Planned Waypoints: L

In order to make the testing of all scenarios more consistent, first a basic initial setup of the parameters is given for each of the above experiments, and then for each experiment one of the parameters is varied to study the effect on the search time for different algorithms. The parameters are listed in the table below:

The number of grid cells, $A \times B = 100$, is kept small in order to keep the run time of one simulation low so that a large number of runs can be simulated that are needed for a statistical analysis. The dimension of the width of the square grid cell $d_{\text{cell}} = 100$ m is chosen so it corresponds to the sensor range of the UAV. The distance of the circle that has to

Parameter	Value [unit]	Description
A, B	10 [-]	Number of cells in the grid in x and y direction.
d_{cell}	100 [m]	The width of the square grid cell.
v	22 [m/s]	Cruising velocity of the UAVs.
d_{circle}	100 [m]	Length of the circular path that the UAV needs to take when: $W_n = W_{n-1}$.
M	1 [-]	Number of UAVs.
n_{fires}	1 [-]	Number of fires.
p_{sure}	0.99 [-]	Required probability for which the fire is assumed to be detected.

Table 4-1: Standard parameters that are used for the test setups.

be flown in other to check a waypoint in the grid twice in a row is equal to 100 meters. Furthermore, the constant velocity v is based on the actual cruising velocity of the UAV used in FireSwarm.

As the performance measure for all the experiments the total search time t_{total} is used:

$$t_{\text{total}} = \frac{1}{v} \sum_{n=2}^{N} \|W_n - W_{n-1}\| \tag{4-27}$$

with $W_1 = (1, 1)$.

4-4-1 Sensor Probabilities: $P_{\text{TP}}, P_{\text{FP}}$

First the influence of different values for $P_{\text{TP}}, P_{\text{FP}}$ is tested. The following values and combinations of $P_{\text{TP}} \in \{1, 0.9, 0.75, 0.5\}$, $P_{\text{FP}} \in \{0, 0.05, 0.10\}$ are tested.

The results of the four strategies, Psychic, Coverage, Random, and Greedy, are shown in Figure 4-5:

It can be clearly seen that when the sensor probabilities become worse, further away from the deterministic case ($P_{\text{TP}} = 1$ and $P_{\text{FP}} = 0$), the difference of the search time becomes larger between the Greedy and Coverage algorithm. This effect can be clearly seen when we look at the second row where $P_{\text{TP}} = 0.9$ and we can see the difference grow between the Coverage and the Greedy algorithm for larger values of P_{FP} (False Positive). Although the same effect can be seen for the probability P_{TP} (True Positive), the effect is weaker and for $P_{\text{FP}} = 0$ not even distinguishable. Furthermore, when sensors probabilities become worse, the Coverage algorithm is not only outperformed by the Greedy algorithm, but there is also no longer a significant difference between the Coverage algorithm and the Random algorithm, i.e. the case were $P_{\text{TP}} = 0.5$ and $P_{\text{FP}} = 0.15$.

The above described effect can be explained by the following: When a sensor is deterministic only one sample is needed at every valid waypoint in the search area to know the occupancy by fire of the entire grid. Therefore, the best strategy is to plan the shortest path that visits all waypoints exactly once. The optimal way to do this is with the coverage algorithm described

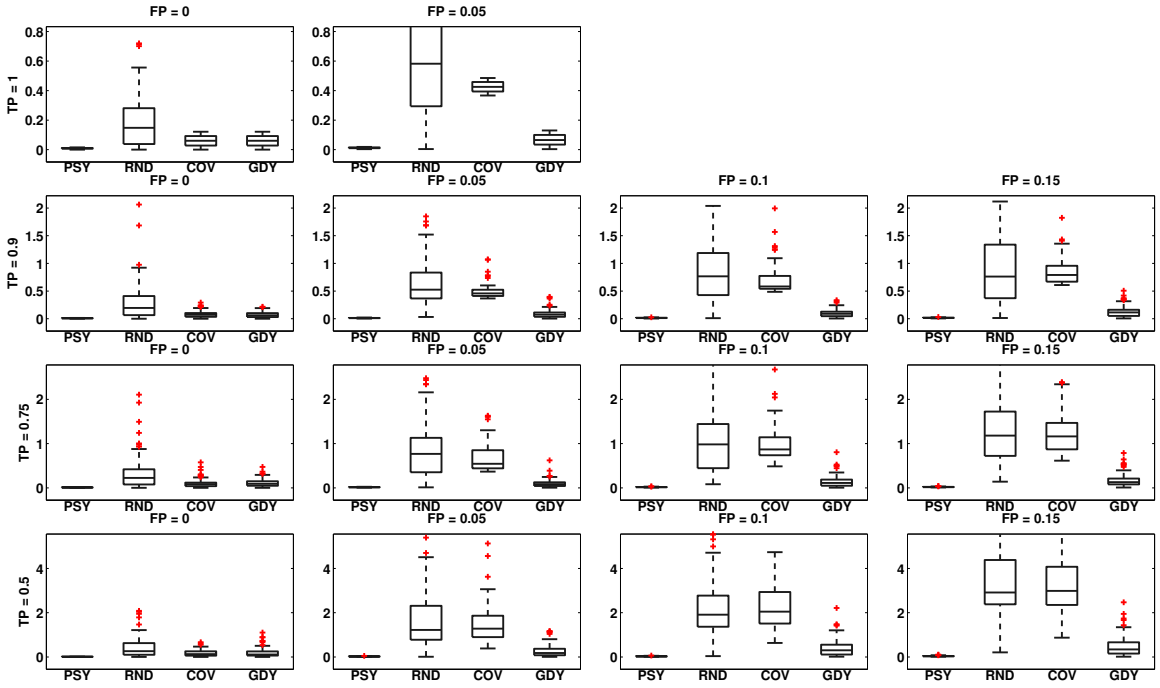


Figure 4-5: Boxplots of 100 runs for each of the four strategies (Psychic, Random, Coverage, Greedy) and 14 configurations of $P_{\text{TP}} \in \{1, 0.9, 0.75, 0.5\}$ and $P_{\text{FP}} \in \{0, 0.05, 0.1, 0.15\}$. On the vertical axis the total search time is displayed in hours. The center line in the boxes mark the median, the box itself encases everything between q_1 (25th percentile) and q_3 (75th percentile). The outliers are displayed as (red) plus signs; a result is an outlier when it is below $q_1 - 1.5(q_3 - q_1)$ or above $q_3 + 1.5(q_3 - q_1)$

in Chapter 3. However, when the sensor readings become increasingly unsure, more evidence is needed to determine whether a cell is occupied by fire. Furthermore, the probabilities of the fire occupancy of the individual cells will become increasingly different, so some areas will become more interesting to revisit and others less. This means that the outcome of the sensor readings determine how interesting it is to take another measurement in that area. This indicates that the optimal path is highly dependent on the collected samples, so a coverage path that samples all cells uniformly is no longer the optimal path. Instead the path should be recalculated dynamically based on the latest update of the found evidence. What the tested Greedy algorithm does, is find a point that is the best trade-off between distance to the current position and possibility to find fire in the area that can be measured from that point. So therefore the simple Greedy algorithm easily outperforms the Coverage algorithm.

4-4-2 Initial Fire Probability: $P(H|\mathcal{E}_0)$

When the first hypothesis in a simulation is tested, an initial probability is required a priori, to be able to update it. This is the probability that a cell is occupied based on no collected evidence. In the previous simulations this value was set to $\frac{n_{\text{fires}}}{A \cdot B}$, presuming that the change for fire in each cell is the same a priori to the search. To see what the influence is of $P(H|\mathcal{E}_0)$ on the total search time, simulations were run with:

$P(H|\mathcal{E}_0) \in \{0.001, 0.005, 0.01, 0.02, 0.05, 0.07, 0.1, 0.2, 0, 5\}$

Figure 4-6 show the results of the above proposed test.

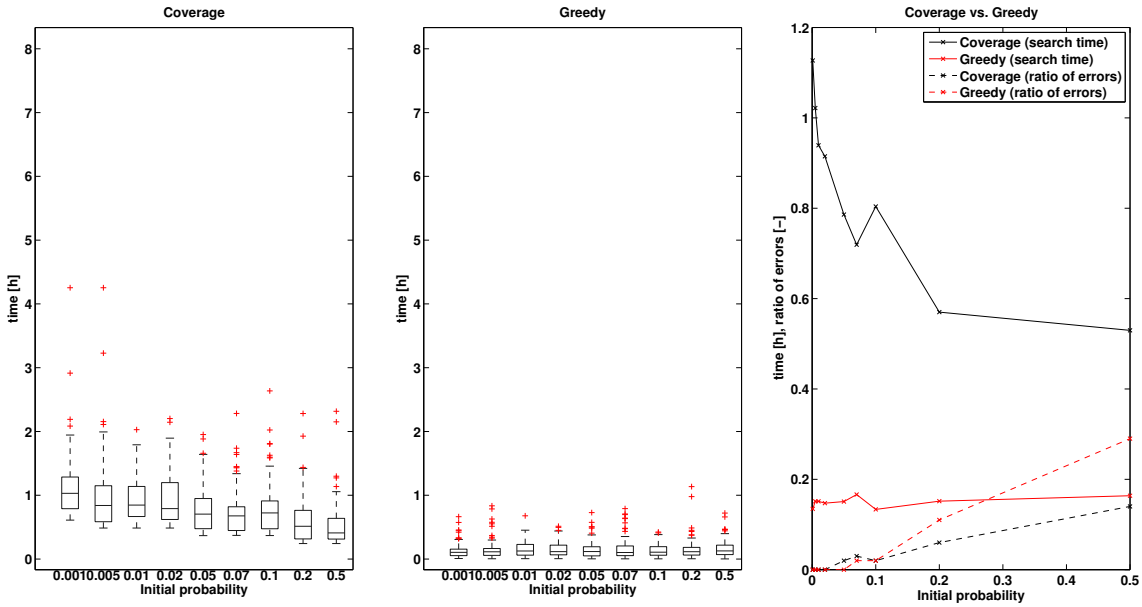


Figure 4-6: The two graphs show boxplots of 100 runs for different values of $P(H|\mathcal{E}_0)$ with the Coverage algorithm and the Greedy algorithm. In the third graph the average results of the total search time are plotted against the different values of the initial occupancy probability $P(H|\mathcal{E}_0)$. The solid lines show the average time for a run as a function of the initial fire probability. The dashed lines indicate the ratio between the number of errors that were produced and the number of runs. (N.B. there can be multiple errors in the same run)

The results in Figure 4-6 show that the increase of the initial occupancy probability value results in a decrease in search time for the Coverage algorithm, whereas no effect can be seen for the total search time when the Greedy algorithm is used.

However, for both algorithms, a rapid increase of the ratio of errors can be seen for larger values of the initial probability. The ratio of errors gives the number of errors summed over all runs divided by the number of runs, where the number of errors within one run can be larger than one.

So for both the Greedy strategy and the Coverage strategy a value of $\frac{1}{A \cdot B} = 0.01$ looks like a good choice for the initial occupancy probability to ensure practically no errors.

4-4-3 Search Area (Grid Size): $A \times B$

In order to test the influence of the dimensions of the search area, simulations are run for different values of A and B while keeping the dimensions of the cell and the velocity the same. The variations in grid size are given by: $A = B \in \{2, 3, 4, 5, 6, 7, 8, 9, 10, 12, 15, 20\}$.

The first two subfigures in Figure 4-7 show boxplots of the total search time for different sizes of the search area with in the first subfigure the Coverage algorithm and in the second

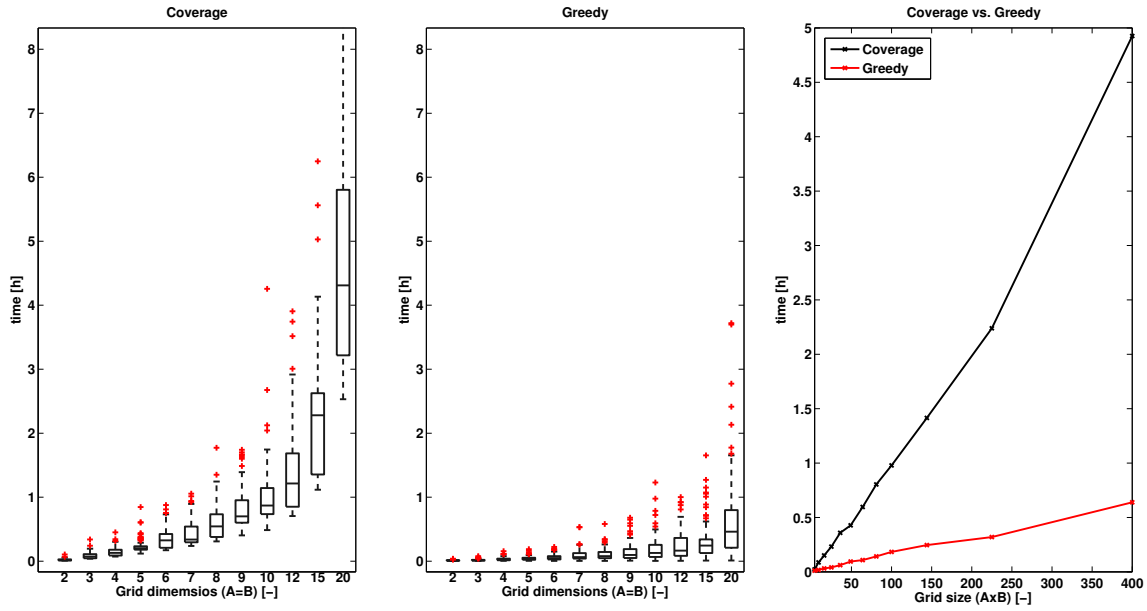


Figure 4-7: The first two graphs show boxplots of the results of increasing number of grid cells. Each of the two graphs shows the results of the strategies: Greedy and Coverage. The third graph shows the average total search time plotted against the number of grid cells for each of the two strategies.

subfigure the Greedy algorithm. The third subfigure shows the average value of the runs per algorithm as a function of the size of the search area, $A \cdot B$. It can be seen that the dependency of the search time on the size of the search area is linear.

4-4-4 Number of UAVs: M

The number of UAVs is tested for the Greedy algorithm only, since the Coverage algorithm does not have any interaction between the UAVs. Therefore, a multiple UAV setup will, for the Coverage algorithm, only result in a smaller number of grid cells that need to be searched by one UAV. Since the influence of the number of grid cells was already tested, the Coverage algorithm is not tested for the number of UAVs.

The results show that for each of the two versions of the Greedy algorithm the decrease in search time is initially very big with the increase in UAVs. However, when the number of UAVs becomes bigger, the decrease in search time per additional UAV becomes smaller.

What is worth noticing is that the normal Greedy algorithm performs worse than when the search area would be split in M equally sized subareas that are to be covered individually by one of the M UAVs. The theoretical results of this scenario are plotted as a dashed line in Figure 4-8 to represent the case where we could indeed perfectly divide the total number of cells in the search area amongst the individual UAVs. From this we can see that the Greedy algorithm with estimated future evidence performs almost as good as the theoretical

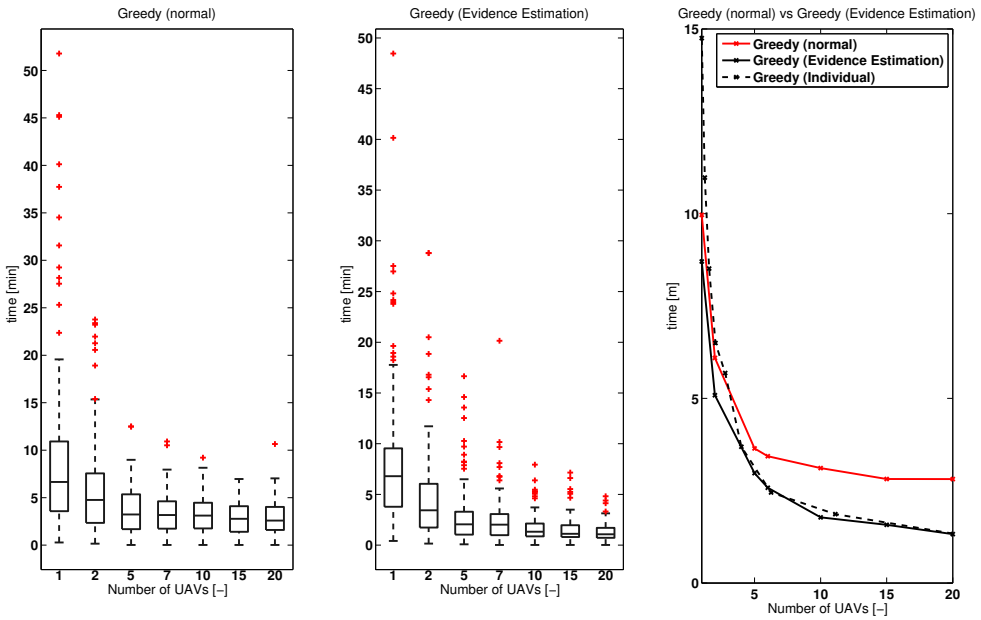


Figure 4-8: The first two graphs show boxplots of the results of varying the number of UAVs for two different algorithms: Greedy (normal) and Greedy (with Evidence Estimation). The third graph shows the average total search time plotted against the number of UAVs for the two algorithms and for the theoretical individual case. The theoretical individual case is based on a perfect division of the grid in M individual subareas that are searched as if it would be a single UAV search. The data that was used was taken from the grid size test that is displayed in Figure 4-7.

case of the individual Greedy algorithm. The data that was used to calculate the theoretical individual case was taken from the earlier performed grid size test, see Section 4-4-3.

So in other words the performance of the combined cooperating UAVs does not result in better performance than the sum of the individual UAVs. However, since the number of grid cells cannot perfectly be divided and also because of the overlapping in the corner the real resulting search time will be slightly longer for the individual Greedy strategy. So, although the Greedy algorithm with the additional future evidence estimation could match or even improve on the performance of the individual Greedy algorithm, future research is needed to further improve the cooperation between UAVs.

4-4-5 Number of Planned Waypoints: L

The influence of the number of waypoints in the path, planned by the Planned algorithm, is also tested. The test uses the following settings for the number of waypoints: $L \in \{2, 5, 7, 10\}$. The results are plotted in Figure 4-9 that shows the total search time for the normal Planned algorithm and the Planned algorithm that incorporates estimated future evidence.

The results show for both varieties of the Planned algorithm an increase in total search time for an increasing number of waypoints in the path. This is of course the opposite effect that

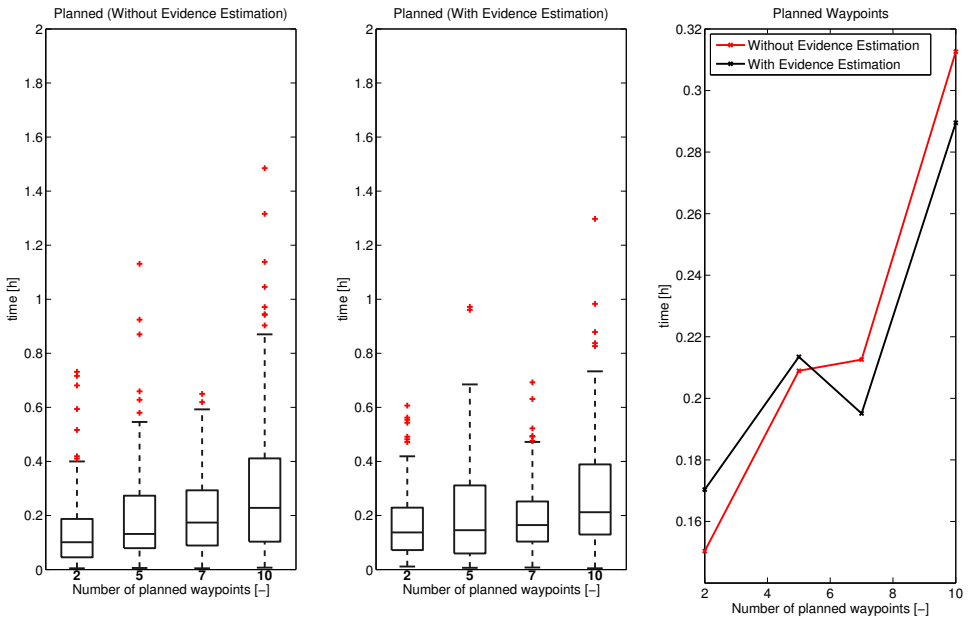


Figure 4-9: The first two graphs show boxplots of the results of different number of waypoints in the planned path for two different algorithms: Planned (without Evidence Estimation) and Planned (with Evidence Estimation). The third graph shows the average total search time plotted against the number of waypoints for the two algorithms.

was aimed for with the Planned algorithm. Even more the results for both varieties of the Planned algorithm are also worse than the results for the standard Greedy algorithm.

This can be explained in two ways. The first one is based on the intuitive notion that it is usually not advantageous or at least very hard to use a future planning algorithm when the world behaves highly stochastically. The second explanation is that the fitness function is very hard to optimize with PSO since there is no functional gradient available in the solution space since a fire can only be measured inside the corresponding grid cell. Furthermore, the fitness function is also not smooth and not continuous which makes it highly convex and therefore difficult for PSO to optimize it.

4-5 Conclusions

The three main conclusions of this chapter are given in this section. First it is concluded that for UAVs with perfect deterministic sensors the optimal strategy to find fire as quickly as possible is a preprogrammed coverage path. This concurs with what was found in Chapter 3. However, when the sensors of the UAV behave more stochastically the Greedy strategy easily outperforms the Coverage strategy and the Coverage strategy performs almost as bad as a Random walk. This effect is most clearly seen when the probability of a false positive increases, whereas the effect is smaller but still noticeable for decreasing probability of a true positive.

Secondly, the addition of the future evidence estimation to the standard Greedy algorithm, proposed in this thesis, resulted in an improvement in terms of total search time. However, this attempt to implement successful cooperation with multiple UAVs did only result in a marginal improvement compared to the sum of the same number of individually operating UAVs. Future research is needed to see whether the cooperation could further be improved.

Thirdly, the performance of the Planned strategy that was implemented to improve on the Greedy algorithm performed worse in terms of total search time. Furthermore, the results suggested that an increase in the number of planned waypoints results in an increase instead of a decrease in total search time. This effect could be caused by the nature of the problem, since it is hard or maybe even impossible to plan in a highly stochastic world. Alternatively, it could be caused by using PSO as an optimization technique since the fitness function does not show a functional gradient and is not smooth and not continuos.

Chapter 5

Conclusions and Recommendations

5-1 Conclusions

This thesis work was focused on finding efficient strategies for a general robotic search problem. Although the problem in this thesis is inspired by the practical problem of the search for fire with a swarm of Unmanned Aerial Vehicles (UAVs) it is defined very generally. The general representation of the problem is done by modeling the sensors by a simple stochastic sampling process and the modeling of the low-level control by point to point waypoint navigation. This general representation makes the results in this thesis applicable to a broad range of real-word problems as is discussed further in Section 5-2.

In this section the main conclusions of this thesis will be discussed. Chapter 3 showed that the search with deterministic sensors can be described as a coverage problem. The coverage problem was decomposed into two problems: the static and the dynamic coverage problem. The static coverage problem is the problem of finding the minimum set of observation points such that the sum of all observations at this set of points covers the total search area. The dynamic problem is the problem of finding the shortest path that visits every observation point in the set of observation points that results after the first decomposition of the coverage problem.

A lower bound of the length of the path was found by combining the proof that the hexagonal grid heuristic provides the optimal placement for the observation points [10] and the Hamiltonian theorem that states that there exists a Hamiltonian cycle in every rectangular T-graph [11]. It was shown for the single UAV case that the lower bound of the path length can be reached by the heuristic method posed in this thesis. The methods proposed in this thesis for the multiple UAV case show near-optimal performance when the number of observation waypoints is assumed far greater than the number of robots.

However, representing the sensors stochastically showed that the coverage methods, which performed optimally in the single robot stochastic case, are easily out-performed by a simple greedy algorithm that was proposed in this thesis. This led to the conclusion that when the sensors used for search behave increasingly stochastically, i.e. more uncertain, the optimal

strategy for search shifts from a pre-planned coverage sweep to a dynamic algorithm that can adapt to new sensor readings. An attempt to improve on the simple greedy algorithm by a planned strategy that uses Particle Swarm Optimization (PSO) to plan a path with multiple waypoints was unsuccessful. However, it is too soon to dismiss the approach of path planning optimized with PSO; so future research is needed. A brief attempt to implement the greedy algorithm on the multiple UAV case was only partially successful. It did only result in a marginal improvement in search time compared to the performance of the sum of the same number of individually operating UAVs.

5-2 Recommendations

Some recommendations can be given for future research. The recommendations are listed and grouped in three groups in increasing order of required research time.

First some recommendations are proposed that directly result from the work in this thesis. For instance the performance of the path planning algorithm could be improved by reconsidering the optimization by PSO. The PSO could be improved by smoothening the fitness function so it is smooth over the entire solution space and shows a gradient. This could be achieved by e.g. a Gaussian representation of the fire probability over the entire search area instead of only a uniformly distributed value inside a grid cell. Although this adjustment cannot add new information it could result in better performance of the PSO algorithm because PSO algorithms are known to perform better with a smooth and continuous fitness function. Additionally, the results of the greedy and planned method could improve when the representation of the sensed area is enhanced from only one grid cell to a number of smaller grid cells. This could result in an actual functional gradient in the search area near a fire, which will probably result in better performance of the algorithms. Furthermore, this multi-cell representation of the sensed area will also result in a more accurate representation compared to the real sensor because it allows for a distribution of the sensor readings within the sensor area that can represent both uncertainties of the sensors as well as the uncertainties of the Global Positioning System (GPS).

Secondly some recommendations are given that are a little further away from the work in this thesis. An extension could be made to include human judgement in the (fire) search algorithms by letting the operator also provide evidence. This evidence could be given a priori, an example of this could be the negative evidence for some subarea that contains water such as a lake or a river or positive evidence for places with dry grass. Furthermore, the human operator could also be put directly into the loop by providing additional evidence to the UAVs based on the judgement of the camera images that are captured by the UAVs. Another recommendation could be to reconsider the representation of the path by a tuple of waypoints and instead represent it as positions at fixed time steps. In other words representing the system as a discrete-time model predictive control problem. This will also allow for an easier incorporation of the dynamics of the UAVs to improve the planned path, i.e. avoid sharp cornering, which will improve the performance for real systems.

Thirdly some recommendations can be made to look at implementations in other fields of research. One could think of implementing a similar greedy algorithm to find leaks in pipelines with autonomous submarines or to look for accidents on highways with UAVs. A more

abstract implementation of the same search strategies could be in fault detection in complex production processes where a faulty product can be caused by a vast number of problems in different parts of the production process. An efficient search method could be used to select what tests need to be taken in order to find the problem as quickly as possible, where all available tests have different stochastic outcomes and test durations. A similar application could be found in diagnostic medicine where the cause of the symptoms of a patient needs to be found as quickly as possible (or with the lowest cost) by selecting a series of different diagnostic procedures. Another application in medicine is the search for tumors with a full-body MRI scanner. The problem with a full-body MRI scanner is that it is impossible to scan and evaluate every part of the body at the highest possible resolution, so a search algorithm could propose an efficient solution to scan the body by deciding to zoom in on some parts of the body based on previous evidence or by deciding to skip over some other parts. The previous evidence can be added a priori when it is for example based on historical data or it can be a result of different observation during the process of scanning.

So it becomes clear that the way that the robotic search problem is represented in this thesis is very general and can therefore represent a broad number of real-world problems in various fields. The work in this thesis contributes to the development of fast and efficient algorithms to solve these kinds of real-world problems.

Bibliography

[1] B. Yamauchi, "A Frontier-Based Approach for Autonomous Exploration," in *Computational Intelligence in Robotics and Automation*, pp. 146–151, IEEE, 1997.

[2] A. Doniec, N. Bouraqadi, M. Defoort, S. Stinckwich, *et al.*, "Distributed constraint reasoning applied to multi-robot exploration," in *International Conference On Tools With Artificial Intelligence*, pp. 159–166, IEEE, 2009.

[3] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon cellular decomposition," in *International Conference on Field and Service Robotics*, vol. 12, Citeseer, 1997.

[4] D. Latimer IV, S. Srinivasa, V. Lee-Shue, S. Sonne, H. Choset, and A. Hurst, "Towards Sensor Based Coverage with Robot Teams," in *Proceedings International Conference on Robotics and Automation*, vol. 1, pp. 961–967, IEEE, 2002.

[5] A. Haumann, K. Listmann, and V. Willert, "DisCoverage: A new Paradigm for Multi-Robot Exploration," in *Internation Conference on Robotics and Automation*, pp. 929–934, IEEE, 2010.

[6] A. Okabe, B. Boots, K. Sugihara, and S. N. Chiu, *Spatial tessellations: concepts and applications of Voronoi diagrams*, vol. 501. Wiley, 2009.

[7] P. Tse, S. Lang, K. Leung, and H. Sze, "Design of a Navigation System for a Household Mobile Robot Using Neural Networks," in *Neural Networks Proceedings*, vol. 3, pp. 2151–2156, IEEE, 1998.

[8] S. Yang and C. Luo, "A Neural Network Approach to Complete Coverage Path Planning," *Transactions on Systems, Man, and Cybernetics*, vol. 34, no. 1, pp. 718–724, 2004.

[9] R. Pitre, "Modified Particle Swarm Optimization for Search Missions," in *Symposium on System Theory*, pp. 362–365, IEEE, 2008.

[10] R. Kershner, "The Number of Circles Covering a Set," *American Journal of Mathematics*, vol. 61, no. 3, pp. 665–671, 1939.

[11] V. S. Gordon, Y. L. Orlovich, and F. Werner, "Hamiltonian Properties of Triangular Grid Graphs," *Discrete Mathematics*, vol. 308, no. 24, pp. 6166–6188, 2008.

[12] D. Bratton and J. Kennedy, "Defining a Standard for Particle Swarm Optimization," in *Swarm Intelligence Symposium*, pp. 120–127, IEEE, 2007.

Glossary

List of Acronyms

PSO	Particle Swarm Optimization
GPS	Global Positioning System
UAV	Unmanned Aerial Vehicle