CHALMERS

Hexacopter Strategies for Wilderness Search and Rescue Missions

Master's Thesis in Engineering Physics

Gustaf Gulliksson

Department of Signals and Systems
Chalmers University of Technology
Göteborg, Sweden 2014

Report No. EX052/2014

THESIS FOR THE DEGREE IN MASTER OF SCIENCE

Hexacopter Strategies for Wilderness Search and Rescue Missions

Gustaf Gulliksson

Department of Signals and Systems
CHALMERS UNIVERSITY OF TECHNOLOGY
Göteborg, Sweden 2014

Hexacopter Strategies for Wilderness Search and Rescue Missions

Gustaf Gulliksson

Master of Science Thesis in cooperation with Combine AB
Report No. EX052/2014
Department of Signals and Systems
Chalmers University of Technology
SE-412 96 Göteborg
Sweden
Telephone: + 46 (0)31-772 1000

Cover:
The hexacopter, developed and built at Combine

Chalmers Reproservice
Göteborg, Sweden 2014

Hexacopter Strategies for Wilderness Search and Rescue Missions
Gustaf Gulliksson
Department of Signals and Systems
Chalmers University of Technology

Abstract

Rescue missions for missing people requires a lot of resources, are difficult to organize and can pose a danger to rescue teams. Time is often a critical parameter and therefore it is important that the search can be performed efficiently. An alternative to today's traditional search by foot can be the use of unmanned aerial vehicles, UAVs. This report describes search algorithms developed in order to optimize the search with several UAVs as well as a simulation platform to test the algorithms.

A model has been developed where the search area is divided into smaller sub-areas, which are prioritized according to how likely it is to find someone in that area. This is estimated from the terrain and external factors such as the position where the person was last seen and if any other traces have been found. The search path, which gives the order in which all the sub-areas are scanned, has to be optimized to minimize the expected search time. The developed optimization algorithm is able to find solutions that are within 11% of the optimal solution.

Sammanfattning

Sökarbeten efter försvunna personer kräver mycket resurser, är svåra att organisera och kan även innebära fara för sökarbetarna. Tiden är ofta kritiskt vid den här typen av uppdrag och det är därför viktigt att sökningen kan ske på ett så effektivt sätt som möjligt. Ett alternativ till dagens traditionella sökning till fots kan vara att använda obemannade luftfarkoster, UAV:er, för att effektivisera sökningen. Denna rapport beskriver dels sökalgoritmer som utvecklats i syfte att optimera sökning med flera farkoster tillsammans, samt även en simuleringsplattform för utvärdering av algoritmerna.

Modellen som utvecklades bygger på att sökområdet delas upp i mindre områden, vilka prioriteras efter hur troligt det är att finna någon inom det området. Detta uppskattas genom att dels väga in hur terrängen ser ut samt att ta hänsyn till andra yttre omständigheter så som vart personen senast sågs eller om något annat spår hittats. Sökvägen, som anger i vilken ordning alla mindre områden ska avsökas, måste optimeras på en sådant sätt att den förväntade söktiden blir så liten som möjligt. Den utvecklade optimeringsalgoritmen lyckades hitta lösningar som är inom 11% från den optimala lösningen.

Acknowledgments

First I would like to thank my examiner at Chalmers, Prof. Martin Fabian, for his guideline and help.

I also would like to thank my supervisors at Combine, Erik Silfverberg and Thomas Hedberg, for giving me the opportunity to do this project and for all your valuable assistance.

My grateful thanks are also extended to all other employees at Combine who have supported me during this project.

Gustaf Gulliksson, Göteborg 140913

List of symbols

A	total search area
n_A	number of sub areas of the total area
i	sub area index
A_i	sub area i
p_i	probability for sub area i
n_H	number of hexacopters
h	hexacopter index
$\mathbb{S}$	the set of possible paths through A
s_k	the path k through A
$s_k(h,j)$	the j:th visited sub area by hexacopter h following s_k
j_h	number of areas given to hexacopter h
t_{A_i}	time when A_i is visited
$E[T]$	expected search time
$E[T]_{s_k}$	expected search time when following s_k
$E^*[T]$	shortest possible search time
s^*	optimum search path
A_R	a set of non-shared out sub areas
$\hat{E}_{A_i}[T]$	expected search time if only sub area A_i is given
p_{A_R}	sum of probabilities for sub areas in A_R
τ_{A_R}	time to search sub areas in A_R
H_R	set of hexacopters not given any sub areas
τ_c	time to search one sub area
$\widetilde{E}^*[T]$	lower approximation of $E^*[T]$

Contents

1 Introduction **1**
- 1.1 Background . . . 1
- 1.2 Aim . . . 2
- 1.3 Scope & limitations . . . 2
- 1.4 Problem Formulation . . . 2
- 1.5 Method . . . 3
- 1.6 Investigation with the police . . . 4

2 Program outline **6**

3 Modeling **7**
- 3.1 Probability map . . . 7
- 3.2 Divide the search area . . . 9
- 3.3 Optimization problem formulation . . . 10

4 Search Algorithm **13**
- 4.1 Traveling salesman problem . . . 13
- 4.2 Start solution - nearest neighbor algorithm . . . 14
  - 4.2.1 NN1 . . . 14
  - 4.2.2 NN2 . . . 15
- 4.3 Optimization - genetic algorithm . . . 17
- 4.4 Paths first . . . 20
- 4.5 Generate hexacopter tracks . . . 21

5 Simulation environment **22**
- 5.1 Simulating the hexacopters . . . 22
- 5.2 Graphical user interface . . . 22
- 5.3 Change conditions during mission . . . 23

6 Results **25**
- 6.1 Cooperation . . . 26

7 Discussion **31**
- 7.1 Nearest neighbor algorithm . . . 31
- 7.2 Genetic algorithm . . . 32
- 7.3 Cooperation . . . 33
- 7.4 Not considered parameters . . . 33

8 Conclusions **34**

9 Future work **34**

1 Introduction

Every year there are around 7000 missing people reported in Sweden and 360 of them are in the police district of Gothenburg. Behind this last number is on average:

- 5 crimes
- 5 cases when someone went lost
- 25 senile which absconds
- 25 cases with ethnic reasons
- 25 suicides
- 25 suicide attempts
- 250 cases of mental problems where the person absconds their relatives
- other reasons

The number of reports is significant higher in the summer period, which can be explained by the main vacations in the summer, people are more outdoors and more people are in movement.[1]

Search missions for missing people is a demanding job and requires a lot of people and good administration. These missions can sometimes pose a risk for the rescue team which puts the group leader in a dilemma, whether it is worth to execute the rescue mission or not.

1.1 Background

During earlier master's thesis projects at Combine an autonomous hexacopter has been developed.[3][4] This air-vehicle is able to work outdoors using GPS navigation. One possible use of these vehicles could be to scan a large area with the purpose of finding a missing person, where time is usually critical. The hexacopters would be able to scan an area from the air much faster than a rescue team by foot. Even though they can fly fast, it does not necessarily mean that they will find the lost person faster since lost people usually behave in a certain way in these situations.

Some advantages of using hexacopters are:

- They can move quickly over large areas
- Decrease the demand of rescue workers

- Can initiate the mission earlier
- Safety, to prevent having rescuers in dangerous conditions
- Do not affect the search area, i.e. leave false traces or destroy evidences.

1.2 Aim

The aim of this project is to develop an algorithm for several hexacopters to scan an area as efficiently as possible. The algorithm should be able to deal with any number of hexacopters and one task to solve is how to optimize their cooperation. The algorithm should also take into account the circumstances of the terrain and properties of the missing person. It has to be investigated which of these circumstances that can be useful during a mission and how they affect the algorithm. A system like this can be of great value for society, and emergency services have asked for this kind of technology.

A second aim is to develop a simulation environment to test the algorithm and virtually visualize the system. It will here be possible to create a mission with various initial conditions and then investigate how the system handles the problem in real time.

1.3 Scope & limitations

The hexacopters will be able to communicate with each other and deliver information such as current position and velocity. A first limitation is that the missing person is assumed to not be moving and wants to be found. There will not be any investigation how a human can be detected, nor any other physical properties of the hexacopter or its communication. The output of this project will give a strategy of how to search an area and thus not necessarily needs to be used by hexacopters. It can be used by any type of UAV's (unmanned aerial vehicles). Since a hexacopter has been developed earlier at Combine, the vehicle will be denote so throughout the report.

This project contains only virtual testing of the algorithm in the simulation platform and no testing on real hexacopters.

1.4 Problem Formulation

The problem is how to scan an area and to find a missing person as fast as possible, with several hexacopters, depending on the circumstances of the terrain and properties of the missing person.

1.5 Method

Initially an information study was done to investigate which parameters of the terrain and the person, that are of importance, this by talking to experienced people in the field. It was also discussed how an automated system is preferred to work for good cooperation with already existing strategies in these kind of missions. It is important that the system does not disturb the mission and is straight forward to use, so it will not need more assistance than it contributes.

The search algorithms was first developed in Matlab, but Matlab has limitations regarding graphical interfaces and multi threading. Therefore, the algorithms were translated into Python, which has a package called QT that can be used to build up a graphical interface, and has build in functions for multi threading.

The simulation environment was also implemented in Python and designed in such a way that the search algorithm works exactly the same regardless of if it is used by real hexacopters or executed in the simulation environment.

1.6 Investigation with the police

A meeting was arranged with a police officer at the Swedish National Police Board (Riskpolisstyrelsen) in Stockholm to discuss how an automated search should be performed, both in the aspect to find the person fast but also for good cooperation with the rescue team.

At search missions the police follows an international method called MSO, Managing Search Operations. This method is based on a large amount of collected data from previous cases, which makes it possible to find behavior patterns for different categories of people. It is primarily children or elderly that go astray, while adults can deliberately stay away due to for example depression. Time is usually a very critical parameter due to the cold climate in Sweden and people that go astray are rarely dressed appropriately to stay out for a long time.

The first search should be of all the known paths in the area. This so since people tends to stay on paths. If they get lost in the woods and find a path they usually start to walk along it. Paths can be searched relatively fast since they are one-dimensional and often open which gives a good sight from the air. In a first step, a hexacopter should fly along the paths and in the next step it can also search through an area around the paths.

The next place to cover should be water edges. Although the probability of finding a person there is low, one wants to make sure that no one has fallen into the water. If so, it is necessary to discover this early since the person can be in an emergency situation.

When those area scenarios are covered, the hexacopters should continue with other open land areas. These areas can be covered fast from air since there is good sight and the hexacopters can fly fast without obstacles. Open water will be low prioritized since it is assumed that no one swims out in a lake if they are lost.

Usually, a lot of information about the terrain and the person is gathered before initiating the search mission. This can be very varying information, which needs to be interpreted together, and does not follow any simple pattern. It is therefore almost impossible to let the algorithm interpret this information. It is best to let the mission leader decide how to prioritize the search area from this information and it should then be possible to add these priorities to the map. Therefore, areas should manually be able to be prioritized higher or lower.

Some possible reasons why areas can be lower prioritized:

- A very unaccessible area where the person would normally not be expected to enter
- A fenced area
- The area on the other side of a river

- If a brief search has already been done in the area

Reasons for higher priority:

- Last known position of the person
- If any trace has been found
- Areas of special interest for the person, for example, mushroom areas if the person were in the woods for this reason
- Higher hills where the person might have gone up to get a better sight
- Power line paths

The most important item here is the last known position of the person. The search should begin from there and spread out. The police uses something called a search circle which is a circle within which one can assume to find the person. The center of the circle is the persons last position and the radius is the distance the person could have walked. The time when the person was at the last known position determines the radius.

Various parameters determine how fast the person moves, such as her age, her condition, if the terrain is easily accessible, if it is dark or daytime, and the weather conditions. If it is hard weather and dark, one assumes that the person has taken cover and have not walked as far as in clear weather.

The search circle could in theory have been generated as a higher prioritized area automatically by giving all these parameters. But since it can depend on so many things it is better that the search leader manually defines this area just as any other prioritized area.

It often happens that the prioritized areas change during the search, some new tip may come in or a new trace is found. One should therefore be able to reorganize the hexacopters to prioritize new areas.[2]

2 Program outline

The program structure is shown in Figure 1. There are three in-parameters to the

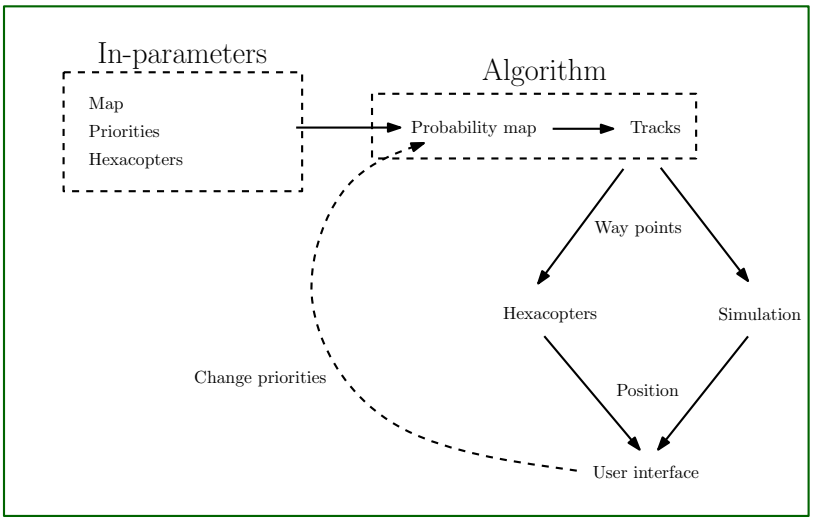

Figure 1: Structure of the program,

algorithm:

- **Map:** Contains size of the search area and its terrain
- **Priorities:** Manually added by the user
- **Hexacopters:** Number of hexacopters, start positions, line of sight and velocity.

The map together with the added priorities generate the probability map, the first step in the algorithm, described in Section 3.1. This is used in the next step to generate the tracks for the hexacopters to follow. The track is a sequence of way points for every hexacopter to travel through.

During a real mission, these tracks will be sent to the hexacopters, but in this case they are sent to a simulation environment instead. The hexacopters, or the simulated ones, return their positions with a certain time interval and also information whether anything interesting has been found.

This is in the end visualized in a user interface where one can see how the hexacopters search through the map. The in-parameters are set in the beginning in the same user interface. Here the user is also able to change the priorities during the mission, these changes are then sent back to the algorithm to generate new tracks for the hexacopters.

3 Modeling

In this section it is discussed how the terrain of the area is modeled and further how the hexacopter tracks are generated.

3.1 Probability map

The first part of the algorithm is to create a probability map, i.e. a scalar field over the search area where the value in every point denotes the probability to find the person in that point. The two parts building up this scalar field is the map, with information about the terrain, and the manually added priorities. How the map is created and how the priorities are added will be described in Section 5.2, which describes the simulation environment.

In the map there are four types of terrain that are considered:

- Path
- Open land
- Forest
- Water

The priority can manually be set to two higher levels, +1 and +2, and two lower levels, -1 and -2, with the default set to 0. Table 1 and Figure 2 gives the value in every point in the scalar field depending on both the terrain and priority. The actual values are not important, but the ratio between them is. This so, since the total probability over the search area will be scaled to sum up to 1.

Table 1: Probability depending on terrain type and manually added priority.

	-2	-1	0	+1	+2
Path	0.01	50	2500	125000	150000
Open land	0.01	1	50	2500	150000
Forest	0.01	1	50	2500	150000
Water	0.01	0.1	5	250	150000

The ratio between different terrains when no priority is added, column ”0” in Table 1, was determined by testing various values. The trade-off here was how far away from for example a path the hexacopter can be and still travel to the path to search there first. The columns to the left and right, -1 and +1, divide and multiply all values

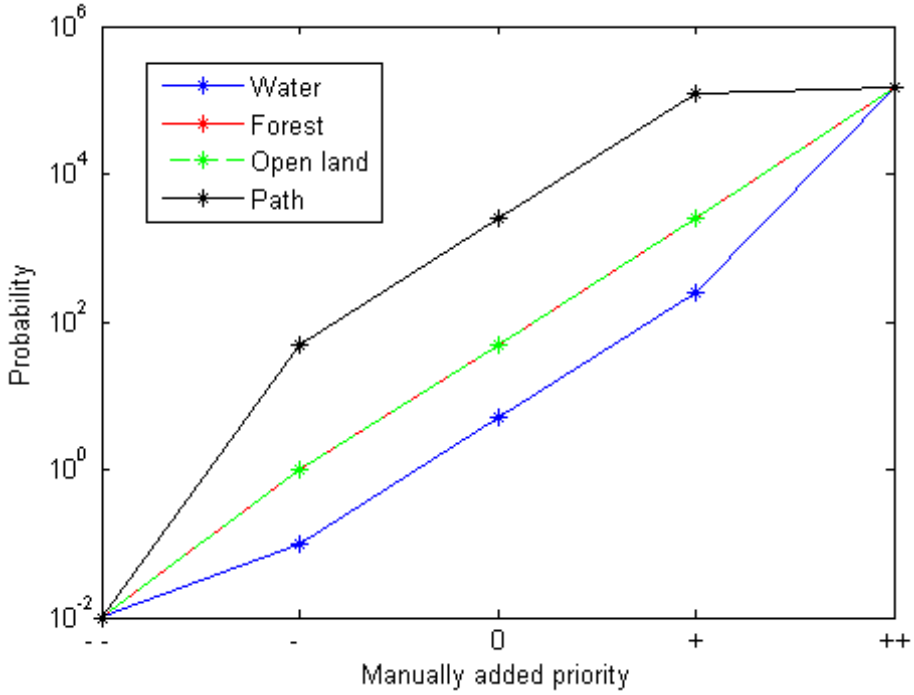

Figure 2: Probability values in every point on the map depending on terrain type and manually added priority.

with 50. The reason why the values are multiplied with a constant and not added to a constant is that the ratio between terrain types is now preserved even if the priority is changed:

$$\frac{\text{Path}_{-1}}{\text{Forest}_{-1}} = \frac{\text{Path}_{0}}{\text{Forest}_{0}} = \frac{\text{Path}_{+1}}{\text{Forest}_{+1}}$$

So if a large area is set to priority +1 or -1 the search within that area will be the same as if no priority was added at all. The value of division and multiplication, 50, is chosen so a lower prioritized path is equal to open land with no priority, and higher prioritized open land is equal to a path with no priority,

$$\text{Path}_{-1} = \text{Open land}_{0} \qquad \text{Path}_{0} = \text{Open land}_{+1}.$$

For priority -2 all values are set to 0.01, which is 10 times lower than the lowest value in priority -1. When this priority is set, those parts will always be taken last regardless of terrain type. If the same ratio between the terrain types was used here some values would be too small and numerical errors could start to occur.

The same reasoning is used for priority +2 where all those values are larger than the largest value in priority +1. So regardless of terrain type, this area will always

be searched first. If the same ratio was kept here some numbers would be very large, which is not directly a problem, but since the whole map is scaled to sum up the probability of 1, large numbers scale down the other values by a large magnitude and one may then get numerical errors.

3.2 Divide the search area

The total search area, A, is divided into n_A sub-areas $A_i \in A$, $i = 1, 2, \ldots, n_A$. These sub-areas are square, with the same side length and placed as in Figure 3 with A_1 starting in the lower left corner and counting up to the right. The reason to divide

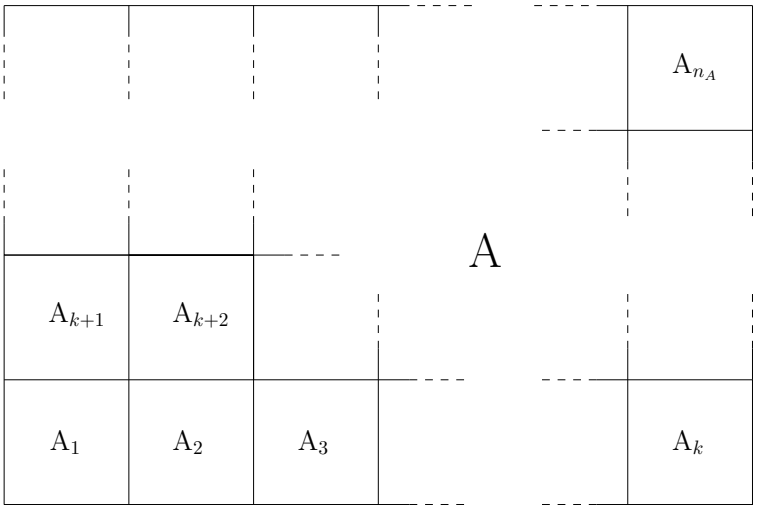

Figure 3: How the search area is divided into n_A sub-areas.

the map into sub-areas is to get a more manageable computational problem, it would not be doable to look at every individual square meter. Every sub-area is searched through in a type of meander pattern, also referred to as the lawnmower pattern, shown in Figure 4.

The size of these sub-areas has impact on the efficiency of the search. For larger squares every lane between the 180^o turns is longer and the total number of turns in the search area A will be fewer. One wants to minimize the number of turns since the hexacopter has to slow down for every turn.[5][6] Larger squares also result in a smaller n_A which reduces the optimization problem. But at the same time, with larger squares the resolution of the map gets worse since this area division gives a coarser grid. The optimum square size should depend on the hexacopter's field of view, which determines the lane width, the number of hexacopters and size of the total search area. No optimization regarding this was done during the project, the

side length of a square is set to 100 length units and the field of view for one hexacopter is set to 10 length units. This results in 10 lanes per sub-area.

As shown in Figure 4 the pattern can be oriented either horizontally or vertically. The hexacopter always enters the square in the corner closest to its current position and tries to exit close to the next square. Depending on how the enter and exit corners are related, that determines whether the hexacopter will search horizontally or vertically.

In Section 3.1, it was explained how the terrain and manually added priorities are combined to give every point on the map a certain probability value. The highest value in area A_i is used as its probability p_i to find the person in A_i. E.g., if a path goes through A_i, the sub-area always gets the path's value as p_i, no matter what the other terrain looks like in that sub-area.

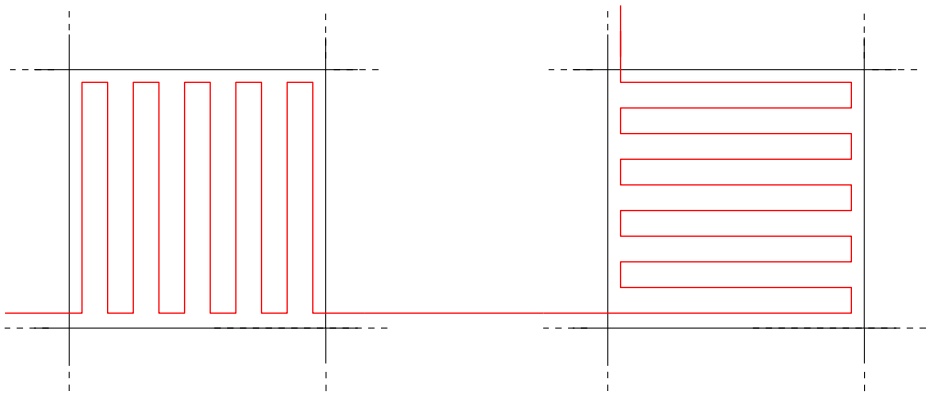

Figure 4: How the lawnmower pattern is performed within every sub-area.

3.3 Optimization problem formulation

The total search area, A, is divided into n_A sub-areas $A_i \in A$, $i = 1, 2, \ldots, n_A$, where the probability of finding the missing person in area A_i is p_i. The probabilities are scaled in such a way that

$$c \cdot \sum_{i=1}^{n_A} p_i = 1, \tag{1}$$

thus it is assumed that the person is within the search area, at least from a mathematical point of view.

The search area will be searched by n_H hexacopters following individual tracks. In the previous section was described how an individual sub-area is searched so it needs to be determined in which order to take the sub-areas.

Let $\mathbb{S}$ represent the set of all possible tracks for n_H hexacopters through the area. Each track, $s_k \in \mathbb{S}$, is defined as n_H sequences where each sequence determine in what order the hexacopter will visit the sub-areas. Every sub-area will be visited exactly once by one of the hexacopters,

$$s_{k_i} \cap s_{k_j} = 0 \text{ for } i \neq j. \tag{2}$$

This originates from the limitation that the person is assumed not to be moving. Traveling directly from any sub-area A_i to any other sub-area A_j is permitted. If there is only one hexacopter, $n_H = 1$, every track s_k is a permutation of the set $\{A_1,A_2,...,A_{n_A}\}$. Otherwise s_k consists of n_H sets, each of a length j_h where h is the hexacopter index, $h = 1, 2, \ldots, n_H$. The only constraint on j_h is that the hexacopters together have to visit all the n_A sub-areas,

$$\sum_{h=1}^{n_H} j_h = n_A \ . \tag{3}$$

A graphical example of a track s_k with $n_A = 100$ and $n_H = 4$ is shown in Figure 5 and has the mathematical representation,

$$s_k = \begin{bmatrix} [\ A_1 & A_{12} & A_{11} & A_{21} & \ldots & A_{74}\ ] \\ [\ A_{13} & A_{24} & A_{35} & A_{45} & \ldots & A_{46}\ ] \\ [\ A_{14} & A_{15} & A_{16} & A_{26} & \ldots & A_{80}\ ] \\ [\ A_2 & A_3 & A_4 & A_5 & \ldots & A_{70}\ ] \end{bmatrix} \ . \tag{4}$$

The j:th visited sub-area by hexacopter h along the track s_k is defined as $s_k(h,j)$.

The time, elapsed from the start of the mission, when sub-area A_i is visited is denoted t_{A_i}. When following the track s_k, the first hexacopter $h = 1$ will visit its first area at $t_{s_k(1,1)}$ and $0 < t_{s_k(1,1)} < t_{s_k(1,2)} < ... < t_{s_k(1,j_1)}$. Since it takes some time to search through one sub-area, t_{A_i} is defined as the time when half the sub-area is searched.

The expected time $E[T]$ it takes to find the person, when following s_k, is

$$E[T]_{s_k} = \sum_{i=1}^{n_A} p_{A_i} t_{A_i} = \sum_{h=1}^{n_H} \sum_{j=1}^{j_h} p_{s_k(h,j)} t_{s_k(h,j)} \ , \tag{5}$$

where the latter expression sums terms in the order of visited sub-areas per hexacopter, instead of the order they where enumerated when dividing the search area.

The optimal track, $s^* \in \mathbb{S}$, is defined such that it minimizes $E[T]$. This minimum value is denoted

$$E^*[T] = \min_{s_k \in \mathbb{S}} E[T]_{s_k} \ . \tag{6}$$

Due to the complexity of this problem it is practically impossible to find s^*, but the aim is to find an s_k which gives an $E[T]_{s_k}$ close to $E^*[T]$ within a reasonable time.

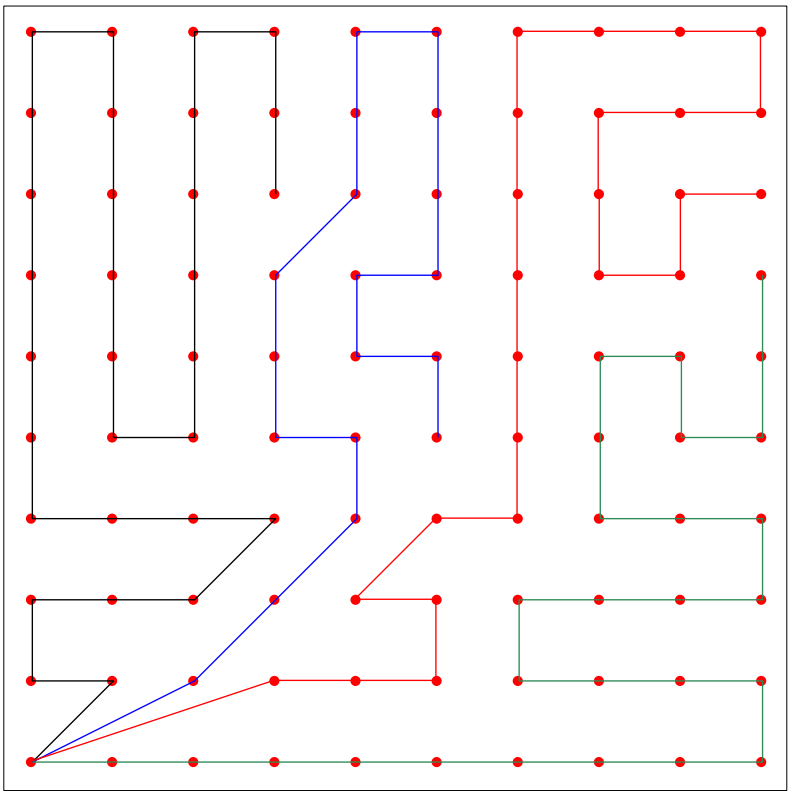

Figure 5: A graphical representation of one arbitrary track s_k given in Equation 4. Each line represent the track of one hexacopter and each dot is a sub-area. The line is not the actual track the hexacopter will follow, it only shows in which order to visit the sub-areas. In this case $j_1 = 31$, $j_2 = 18$, $j_3 = 24$ and $j_4 = 27$ which is the number of points on each line.

4 Search Algorithm

The main part that affects how the system performs is how the tracks are optimized. One wants to find a track that gives an expected search time close to $E^*[T]$ within a reasonable calculation time as described in Section 3.

4.1 Traveling salesman problem

This optimization problem is closely related to a famous problem called the traveling salesman problem, or TSP. Here one asks for an optimal tour through a specified set of cities and all cities shall be visited exactly once. The optimal tour is the tour that minimizes the total distance traveled. In our case, it is instead of interest to minimize the expected searching time, rather than the total distance. The ordinary TSP has only one salesman but this system must be able to handle several hexacopters. Apart from this, the problems are similar and the cities represent the sub-areas in our case.

The TSP is a so called NP-hard, Non-deterministic Polynomial-time hard problem, which briefly means that it is not possible to find the exact solution within a polynomial time.[7] This problem is also non-convex and nonlinear. This means there exists many local minimum where deterministic functions easily get stuck. A relatively simple way to circumvent this is to use genetic algorithms, which are very efficient for this kind of problems. Genetic algorithms were developed in the field of artificial intelligence and mimics the process of natural selection. One usually wants to know how far the optimal solution is from the achieved solution. Even though s^* is unknown it is in some problems possible to find $E^*[T]$ and compare it to the current result to see if it is close enough. A limitation of the genetic algorithm is that it cannot determine $E^*[T]$. What is possible though is to execute the algorithm several times. If it ends up roughly with the same $E[T]$, it should be close to $E^*[T]$ and be a good enough solution.[8]

The genetic algorithm needs a start solution which can be generated with several techniques. With a good start solution the genetic algorithm finds a good solution faster, but it may also take more time to find a better start solution, so there is a trade-off here. One good technique for a start solution is the nearest neighbor method. Here a starting point is chosen and from there one always moves to the closest not visited point, its nearest neighbor, until all points are visited. In the beginning one moves short distances but towards the end when there are few points left they are usually far separated as can be seen in Figure 6. It has been shown that for many randomly distributed cities, the nearest neighbor algorithm on average yields a tour 25% longer that the optimum tour.[9]

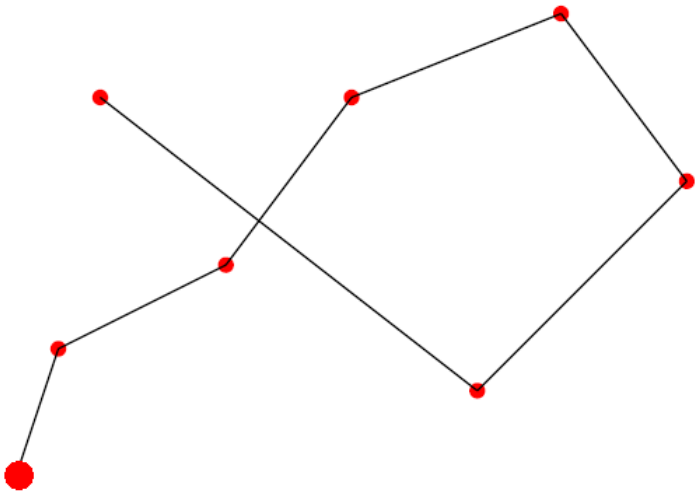

Figure 6: An example of how the NN algorithm calculates the track if the large dot is the starting point. Note how the early moves are shorter and the latter are longer.

4.2 Start solution - nearest neighbor algorithm

To get a start solution, a variant of the nearest neighbor algorithm is used, where the next point minimizes the expected search time instead of distance. Two types of this algorithm is used, which are denoted NN1 and NN2.

4.2.1 NN1

Pseudo code for this method is shown in Algorithm 1.

Algorithm 1 NN1: Create an initial track for every hexacopter

```
while A_R is not empty do
  for all hexacopters do
    for n = 1,2,3 do
      for all A_i ∈ A_R do
        calculate Ê_{A_i}[T]
      end for
      add area, that minimizes Ê_{A_i}[T], to s_k for current hexacopter
      delete A_i from A_R
    end for
  end for
end while
```

Initially A_R is the set of all sub-areas. As the sub-areas are given out to the hexacopters they are also deleted from this set, so A_R contains the remaining sub-areas. The outermost while loop runs until A_R is empty. Then all the sub-areas are distributed among the hexacopters and the track s_k is complete. The next for loop iterates over every hexacopter. For each one of them three areas are given out as n iterates over 1,2 and 3. $\hat{E}_{A_i}[T]$ is then calculated for all remaining areas, $A_i \in A_R$. The sub-area that minimizes $\hat{E}_{A_i}[T]$ is added to the track s_k for the current hexacopter and is at the same time deleted from A_R.

$\hat{E}_{A_i}[T]$ is the expected time to find the person if the search track starts with A_i.

$$\hat{E}_{A_i}[T] = p_{A_i} t_{A_i} + p_r t_r \tag{7}$$

where p_{A_i} is the probability for A_i and t_{A_i} is the time is takes for the hexacopter to travel the distance to A_i and search it through. The distance is from the hexacopter's recent area, or from its initial position if this was its first area. p_r is the probability the person is not in A_i,

$$p_r = 1 - p_{A_i}. \tag{8}$$

t_r is the time it takes to search through the rest of the areas if one assumes no travel time between the sub-areas. The term $p_r t_r$ gives an average value over the total search area excluding A_i. This expected value can be extended to include more than one sub-area by adding more terms into Equation (7). Sub-areas searched before A_i could be added but this not necessary. This would just add the same constant to $\hat{E}_{A_i}[T]$ for every A_i and not affect which sub-area that gives the minimum.

After testing with various map sizes and numbers of hexacopters, the best results where generally achieved when the sub-areas were given out three at a time. This number can be decreased if there are few sub-areas. If this number is too small, for example one, the hexacopters would get every other sub-area. Thus it would be a lot of traveling time between the sub-areas since they would be less connected. If the number is too large, the first hexacopter would receive many sub-areas with the highest priority and then work on its own exploring this large area. It would be better if this area is divided among several hexacopters.

At the end this algorithm usually not adds up even with the sub-areas. The last ones are instead tested on all hexacopters to determine which of them the sub-areas should be assigned to. This extra part is not included in the described algorithm 1.

4.2.2 NN2

NN1 does not take into account which hexacopter that is closest to the best sub-areas, the best ones are given to the first hexacopter. Usually the hexacopters are started up on the same position so it does not matter which of them is given the

first assignment. It will be described later how the mission can be interrupted and restarted with new priorities. Then the hexacopters will continue from their current various positions. In this case a slightly different algorithm needs to be used, which is described in Algorithm 2.

Algorithm 2 NN2: Create an initial track for every hexacopter with different start positions

```
while A_R is not empty do
  if H_R is empty then
    H_R ← 1 : n_H
  end if
  for all hexacopters ∈ H_R do
    for all A_i ∈ A_R do
      calculate Ê_{A_i}[T]
    end for
    h ← min(E)
  end for
  H ← indexOfMin(h)
  for n = 1,2,3 do
    for all A_i ∈ A_R do
      calculate Ê_{A_i}[T]
    end for
    add area, that minimizes Ê[T], to s_k for hexacopter H
    delete A_i from A_R
  end for
  delete H from H_R
end while
```

In this algorithm a vector H_R keeps track of the hexacopters that have not been given orders. The best start sub-area is calculated for every hexacopter in H_R and saved in a vector h. After all hexacopters in H_R have been considered, the smallest value in h determines which hexacopter provides the best $\hat{E}[T]$. This one gets three sub-areas and is at the same time deleted from H_R. When all hexacopters have been given three sub-areas, and H_R is empty, H_R is filled again with all the hexacopters. Thus it works the same as Algorithm 1, except that the best suited hexacopter is always given orders instead of the next hexacopter in numerical order.

4.3 Optimization - genetic algorithm

To further optimize the start solution, which is the output from the nearest neighbor algorithm 1 or 2, a genetic algorithm is used. How the algorithm is performed is described in Algorithm 3.

Algorithm 3 Genetic algorithm for track optimization

```
track[1 : 80] ← startSolution
globalMin ← inf
loop
  for all tracks in track do
    calculate E[T]
  end for
  if min(E[T]) < globalMin then
    globalMin ← min(E[T])
    bestTrack ← track with least E[T]
  end if
  track ← permute(track)
  for p = 8 : 8 : 80 do
    bestOf8Track ← track[p − 7 : p] with least E[T]
    tmpTrack[1 : 8] ← bestOf8Track
    for k = 1 : 8 do
      perform genetic mutation k on tmpTrack[k]
    end for
    newTrack[p − 7 : p] ← tmpTrack
  end for
  track ← newTrack
end loop
```

In this optimization it is not enough to just calculate $\hat{E}_{A_i}[T]$ since the genetic mutations may effect the whole track so $E[T]$ is used and calculated according to Equation (5). Every iteration in the loop represents one generation with a population of 80 individuals, where each individual represents a track.

First in the loop, $E[T]$ is calculated for all tracks. If one of them has a smaller $E[T]$ than $globalMin$, the saved best value, this track will be saved as the new best track. Then all individuals are mixed, permutation of $track$ and the first eight are chosen, this is now eight randomly picked from the whole population. The best one of these eight, the one with least $E[T]$, is denoted as $bestOf8Track$ and is kept. The other seven tracks are thrown away. $bestOf8Track$ mutates into eight new individuals which are the first individuals in the next generation. Then the next eight

individuals are picked from *track*, only the best one is kept and mutated into eight new individuals. Eight individuals are picked from *tracks* until all have been used. Now a new complete generation is created slightly different from the previous one. Why eight individuals are picked every time and why the population size is 80 is discussed in Section 7.2.

The more generations that is generated the higher is the chance to receive a good solution and thus lower $E[T]$. A typical example of how a solution can evolve is illustrated in Figure 7 where the solution improves in the beginning and eventually levels out. When the solution has not improved any more within a certain amount of generations the algorithm can be stopped and one can assume that the solution will not improve more even if it continues to execute.

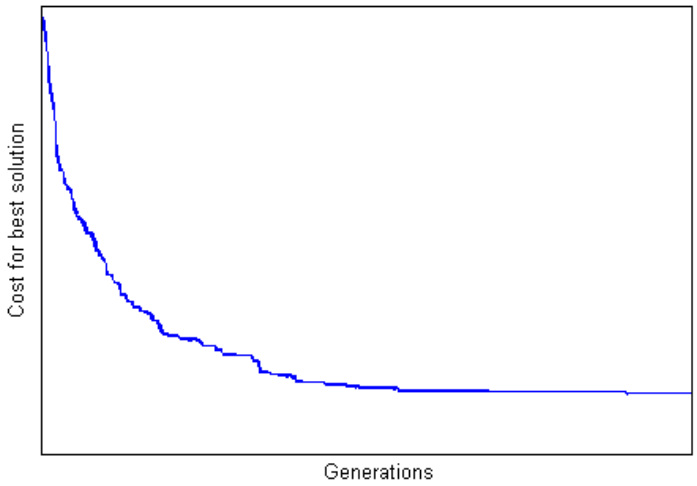

Figure 7: Illustration of how the solution evolves with a genetic algorithm. The cost, which is desired to be minimized, decreases faster in the beginning and eventually levels out.

The following eight genetic mutations are performed:

1. Do nothing, keeps $bestOf8Track$
2. Swap two sub-areas
3. Swap two pairs of sub-areas
4. Invert a vector of sub-areas
5. Slide sub-area up
6. Slide sub-area down

7. Slide the last sub-area down

8. Swap two vectors of sub-areas between two hexacopters

where mutation 4-7 only occur within the sub-area sequence for one hexacopter, 8 is always between two hexacopters and 2 and 3 can be between any sub-areas. Since mutation 1 does not change the individual, it always keeps the best track. Mutation 7, *slide the last sub-area down*, may be considered as already covered by mutation 4, *slide sub-area down*. The reason why to use an extra mutation for the last sub-area is that the last sub-areas are usually worse optimized from the nearest neighbor algorithm.

Some of these mutations are graphically explained in Figure 8. At the top is mutation 2, where two sub-areas are swapped, and mutation 3, where two pairs are swapped. In the illustration the swap is between two hexacopters, but it could also be within the sub-area sequence of the same hexacopter. The two mutations at the bottom is number 4, invert a vector, and number 6, slide area down. These always occur within the sub-area sequence of one hexacopter.

Mutation 4 in the list, invert a vector, unties knots in the track which is illustrated also in Figure 9.[10]

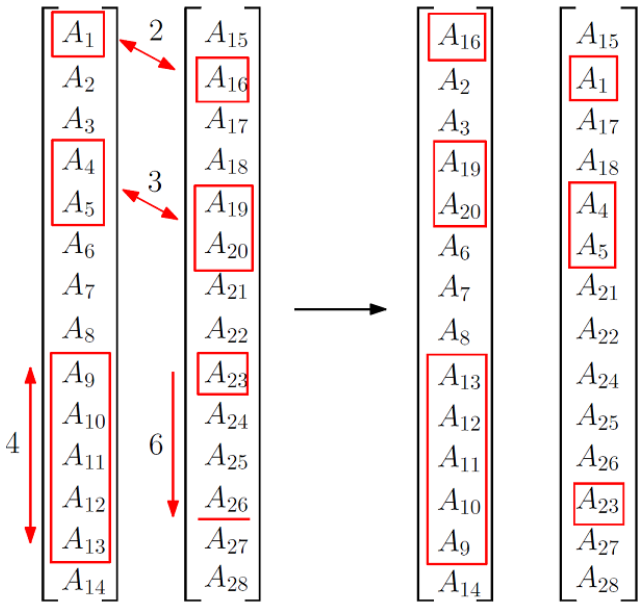

Figure 8: This figure illustrates four of the genetic mutations applied on the sub-area sequences for two hexacopters.

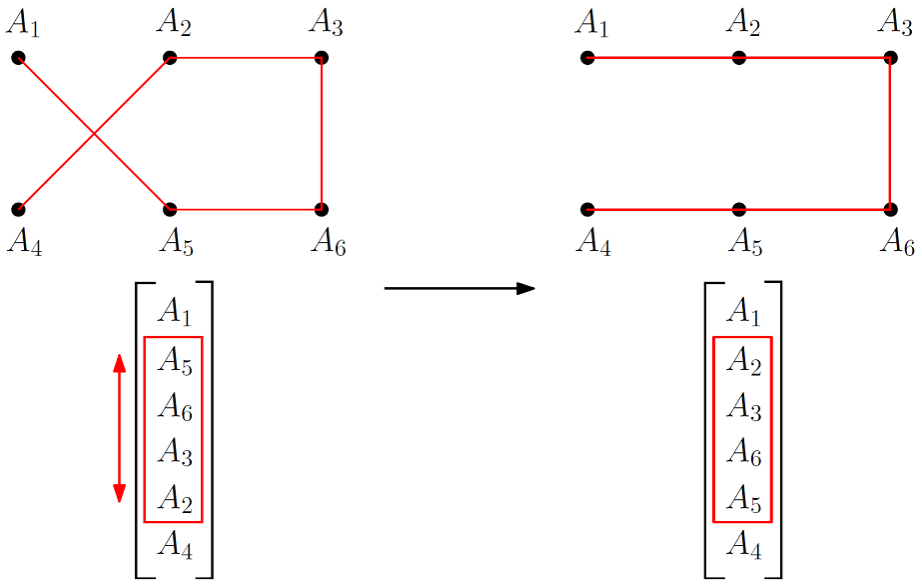

Figure 9: Illustration of how mutation 4, invert a vector, unties knots. In this case the four sub-areas in the middle are inverted. This results in a shorter track since it does not cross itself anymore.

4.4 Paths first

Since people tend to stay on the paths, and not walk beside them, it would be a waste of time to initially search paths as a meander pattern. Therefore some hexacopters will search along the paths first by following them. For every two paths on the map, one hexacopter is involved in this search. These tracks for the hexacopters are generated first since the end coordinates of this search are the start coordinates in the nearest neighbor algorithm. Paths are given out to the hexacopters with the same principle as the nearest neighbor algorithm, closest first.

This path search is not considered when calculating $E[T]$ since first, the model for $E[T]$ is only defined to include sub-areas and second, $E[T]$ is only used to optimize the sub-area order. However if some hexacopters are doing path search, the first sub-areas will be given to other hexacopters. Hexacopters doing path search are therefore given less sub-areas and an example of this is shown in Figure 10.

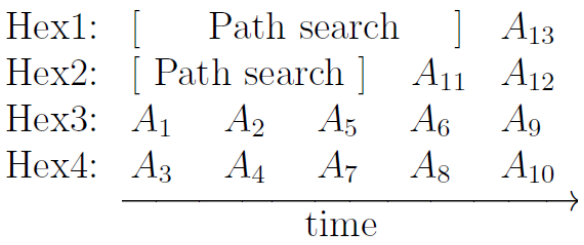

Figure 10: Hexacopter 1 and 2 are first searching paths and is therefore given less sub-areas. The best sub-areas are distributed to hexacopter 3 and 4 in order for those sub-areas to be searched as early as possible.

4.5 Generate hexacopter tracks

When the whole sequence of sub-areas are set, the enter- and exit corners for each sub-area can be determined. A hexacopter should enter a sub-area in the corner closest to its current position and try to exit from corner closest to the next sub-area. The relation between enter and exit corner for a sub-area determines whether the search orientation is horizontal or vertical within that sub-area.

The output of the algorithm is tracks consisting of way points, GPS-coordinates, for the hexacopters to follow. Each hexacopter is given an individual set of way points which are followed in a certain order. A sub-area consists of 20 way points, i.e. the corners in the meander pattern in Figure 4. It is 20 since the sub-area square length is set to 100 length units and the width of the field of view for one hexacopter is set to 10 length units which results in 10 lanes per sub-area. Way points for a path search is points along the path.

5 Simulation environment

To be able to test the algorithm a simulation environment had to be developed; it would not be practically possible to do live testing. However for this project it is not necessary to test on real hexacopters, this so since the algorithm does not take any input from the hexacopters and works the same regardless if tested in simulation or in a real environment.

The simulation environment consists of mainly two parts; one that simulates the hexacopters and how they fly depending on the output of the algorithm, and one part that is a graphical interface where the mission parameters are set and where the map is created with terrain and prioritized areas. After the mission is started, one is able to follow the search on the map in real time.

5.1 Simulating the hexacopters

The algorithm gives an output of tracks with way points for the hexacopters to follow. Usually these way points are sent to the real hexacopters and they continuously send back their positions. This is now instead simulated and runs in its own thread apart from the rest of the program to achieve a realistic simulation.

This thread receives all the way points after the algorithm is finished. The average speed of an hexacopter is known so the positions along the tracks can continuously be estimated during the whole search and sent back to the program.

5.2 Graphical user interface

Initially a picture of the search area is loaded into the graphical user interface. One has to specify the number of hexacopters that will be used and indicate their start positions. These positions are used when optimizing the track and they are the starting points of the simulated hexacopters. The real hexacopters will of course not consider where these points are placed, they start where they are, but their first order depends on these points.

The program cannot recognize the terrain itself from the map, thus the user has to point out all terrain objects such as paths, water, open areas and woods. The prioritized areas are also defined here, as described in Section 3.1. All these objects, except from the paths, are represented by closed polygons where the corners are coordinates on the map. The paths are line objects and instead represented by open polygons. Every point on the map is evaluated whether it is inside any of these polygons or along a path line. A point can be both inside a polygon describing the terrain and a polygon for the priority. As described in Section 3.1, these two parts

of information are used to create the probability map. I screen shot of the graphical user interface can be seen in Figure 11.

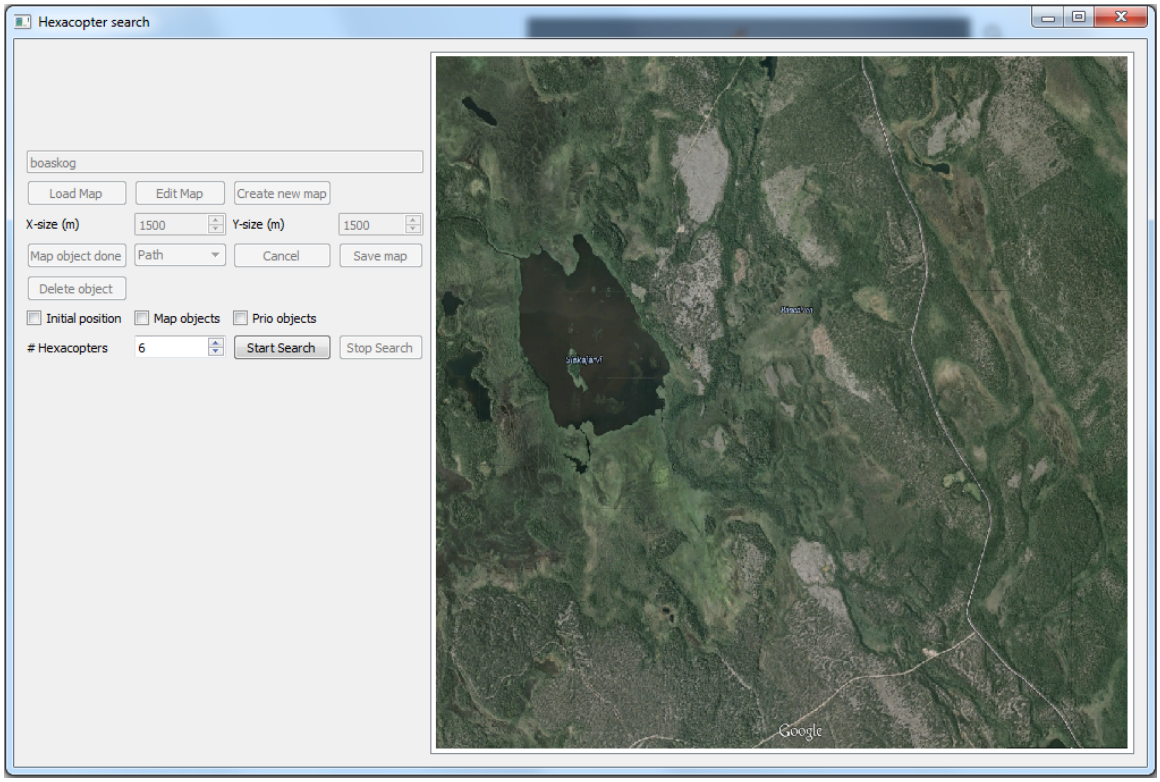

Figure 11: A screen shot of the graphical user interface. With the buttons to the left one is able to define the terrain and prioritized areas on the map to the right.

5.3 Change conditions during mission

It is possible to stop the search during a mission to change the terrain or priorities. The reason why one would need to do this was discussed in Section 1.6, e.g. a trace, like a glove, has been found and one want to concentrate the search to that region. When the mission is stopped the hexacopters will complete their current order and then wait until the user have restarted the mission.

The hexacopters receive way points in sets of 20 which corresponds to one sub-area. When one such set of way points has been searched, the hexacopter reports back that the sub-area is clear. The indices of all finished sub-areas are saved and when new tracks are calculated those sub-areas are not included in the algorithm. One can view this as they are deleted from A_R in Algorithms 1 and 2 already when

the calculation starts. The output is shorter tracks without the already searched sub-areas.

In this case is it important to use the second algorithm for the nearest neighbor method, Algorithm 2, since the hexacopters are located on different positions over the map when the search is restarted.

6 Results

In some optimization problems, where the optimum solution is unknown, the optimal value can be estimated and compared to the achieved value in order determine how good the solution is. But in these kind of problems it is not possible to estimate the optimum value $E^*[T]$, though some other values can be used to evaluate the solutions. One interesting case is if all areas were searched in an order so no traveling distances are needed and the priorities are not considered. This could be considered as taking an average over the whole search area, as done when calculating $\hat{E}_{A_i}[T]$ in Equation (7), but with A_i empty. This value is denoted $\hat{E}_0[T]$ and is calculated

$$\hat{E}_0[T] = \frac{Nt_c}{2} \cdot \frac{1}{n_H} \ , \tag{9}$$

where N is the total number of sub-areas, t_c is the time needed to search through one sub-area and n_H is the number of hexacopters used in the search mission. Nt_c is the time required to search all the sub-areas and the expected search time is given by dividing by 2, this so since the person is on average found when half the search area is scanned. Division by the number of hexacopters assumes all hexacopters search over equally large areas. This is equivalent to only consider the last term in Equation (7) with all sub-areas included in this term.

During the distribution of sub-areas with the nearest neighbor algorithm, $\hat{E}[T]$ is calculated and should decrease as more sub-areas are given out. If A_1, A_2 and A_3 are given out, the search time is

$$\hat{E}_{A_1,A_2,A_3}[T] = p_{A_1}t_{A_1} + p_{A_2}t_{A_2} + p_{A_3}t_{A_3} + p_r t_r \ . \tag{10}$$

The optimum value $E^*[T]$ is a lower bound but unknown and cannot be calculated. It can be approximated with a lower value denoted $\widetilde{E}^*[T]$. In this approximation it is assumed the sub-areas are searched in the order of decreasing p_i and with no traveling time. With n_H hexacopters, the n_H sub-areas with highest priority are searched at $t_c/2$, the next n_H sub-areas at $3t_c/2$ and so on. Equation (5) is then used to calculate $\widetilde{E}^*[T]$. This lower bound is not possible to reach since the sub-areas are usually not placed in such a way that they can be searched in the right order without any traveling.

For the results presented in this section the algorithm is executed on two test maps, A and B, shown in Figure 12. In Figure 13 and 14 are the results from test map A, with $n_H = 3$ and $n_H = 6$, presented respectively and in Figure 15 and 16 are the results from test map B. The figures shows how $\hat{E}[T]$, calculated from Equation (10), decreases with the nearest neighbor algorithm as more sub-areas are divided among the hexacopters. The final value for the nearest neighbor algorithm is the initial value

for the genetic algorithm which optimizes $E[T]$ further for 1000 generations and is also presented in the same figures.

The values of $\hat{E}_0[T]$ and $\widetilde{E}^*[T]$ for each of these four cases are presented in Table 2 and included in Figures 13, 14, 15 and 16. In Table 3 is $\widetilde{E}^*[T]$ compared to the achieved values for $E[T]$.

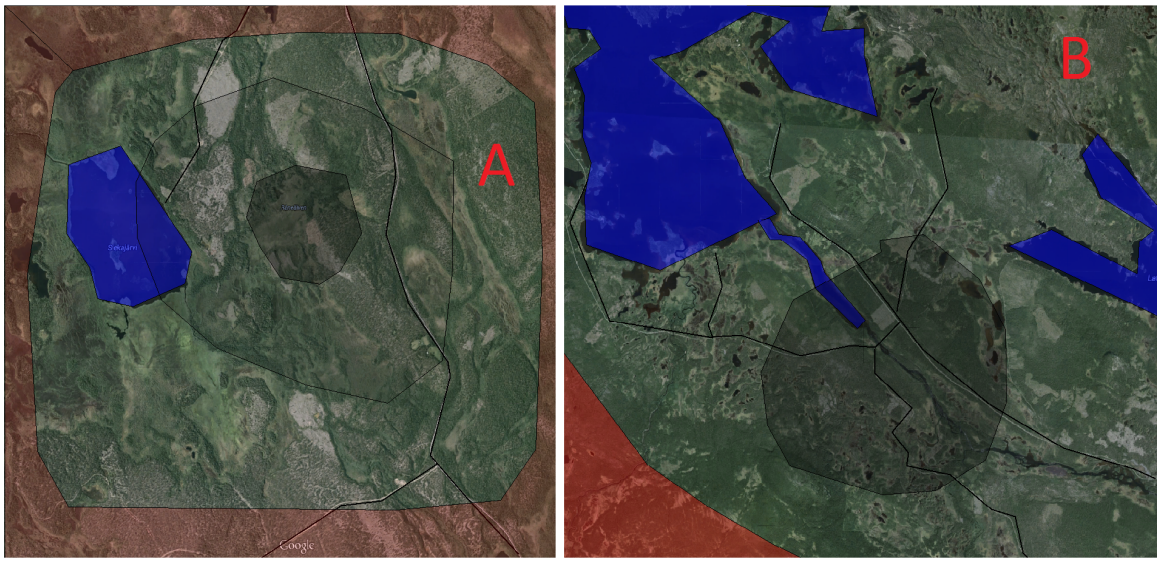

Figure 12: Two created test maps A and B used to evaluate the algorithm. The dark parts are high prioritized areas and the red parts are low prioritized areas. Paths are represented by black lines and the blue parts are water. Both maps are 1500×1500 length units.

6.1 Cooperation

It is also important to investigate how the search time scales with the number of hexacopters. The algorithm is executed on both test maps with 1000 generations for $n_H = 1, 2, 3, \ldots, 12$. The test was performed 10 times for every n_H and the average results are presented in Table 4 and in Figure 17 for test map A and in Figure 18 for B together with theoretical values. The theoretical values are estimated by starting with the optimized value for $n_H = 1$, and then dividing by the number of hexacopters, $E_{n_H}[T] = E_{n_H=1}[T]/n_H$.

Table 2: The values for $\hat{E}_0[T]$ and $\widetilde{E}^*[T]$ on test map A and B with $n_H = 3$ and $n_H = 6$. These values are included, as red dashed lines, in Figures 13, 14, 15 and 16.

	Tesatmap A		Tesatmap B	
	$\hat{E}_0[T]$	$\widetilde{E}^*[T]$	$\hat{E}_0[T]$	$\widetilde{E}^*[T]$
$n_H = 3$	3750	574	3750	885
$n_H = 6$	1875	278	1875	438

Table 3: Values for $\widetilde{E}^*[T]$ from Table 2 compared to achieved values presented in Figures 13, 14, 15 and 16.

Map/n_H	$\widetilde{E}^*[T]$	$E[T]$	Deviation
A / 3	574	626	9%
A / 6	278	303	9%
B / 3	885	955	8%
B / 6	438	486	11%

Table 4: Search time $E[T]$ depending on the number of hexacopters n_H for test map A in the table to the left and for test map B in the table to the right. These values are plotted and compared with estimated theoretical values in Figures 17 and 18.

n_H	$E[T]$ (s)
1	1820
2	954
3	647
4	505
5	431
6	379
7	354
8	335
9	321
10	311
11	308
12	305

n_H	$E[T]$ (s)
1	2865
2	1394
3	995
4	775
5	657
6	579
7	535
8	497
9	479
10	466
11	453
12	444

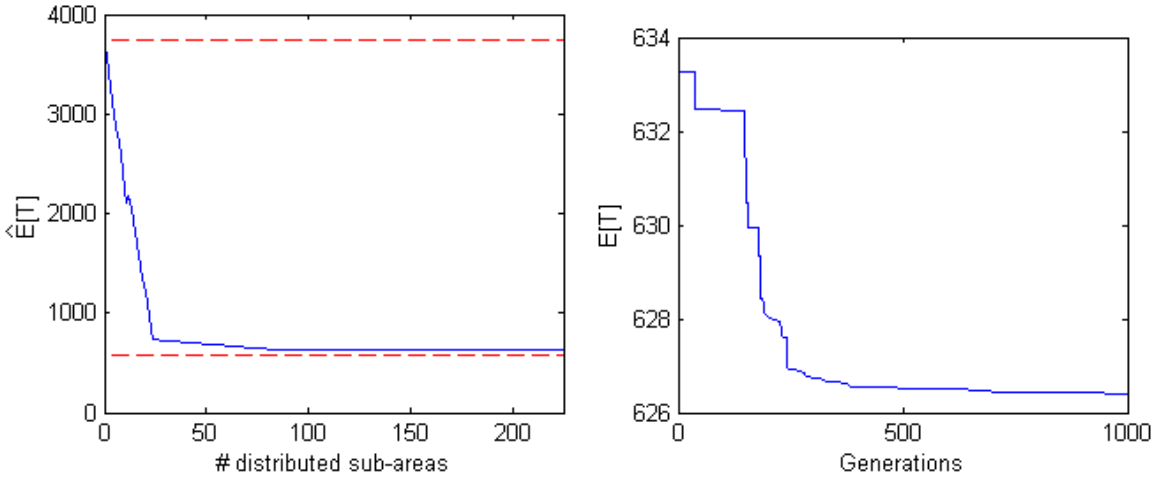

Figure 13: Results for 3 hexacopters on test map A. The left figure shows how $\hat{E}[T]$ decreases with the nearest neighbor algorithm as the sub-areas are divided among the hexacopters. The top red dashed line is $\hat{E}_0[T]$ and the bottom red dashed line is $\widetilde{E}^*[T]$, also presented in Table 2. The final value in the left figure is the initial value in the right figure which shows how the genetic algorithm optimizes further for 1000 generations of track mutations.

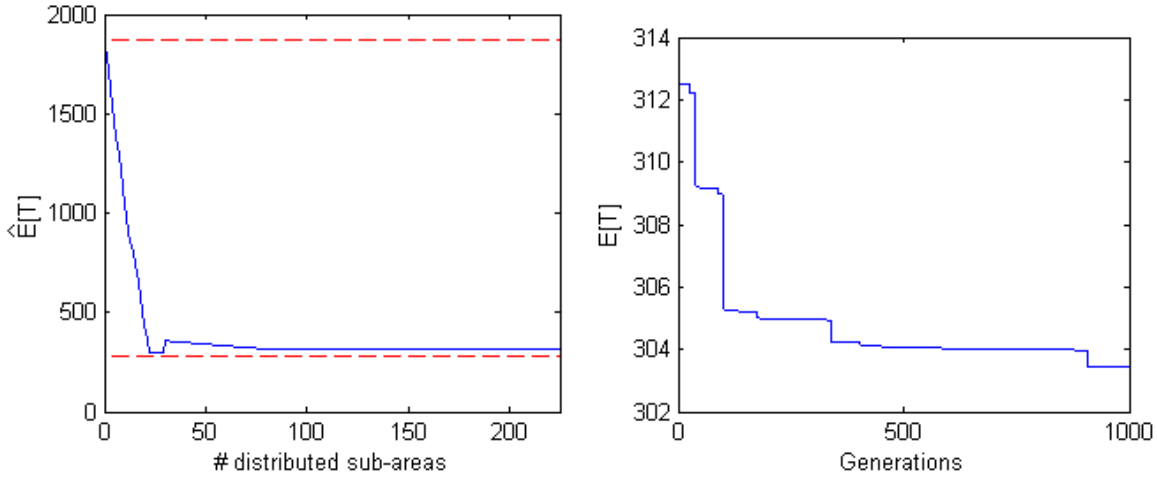

Figure 14: Results for 6 hexacopters on test map A with the nearest neighbor algorithm to the left and the genetic algorithm to the right.

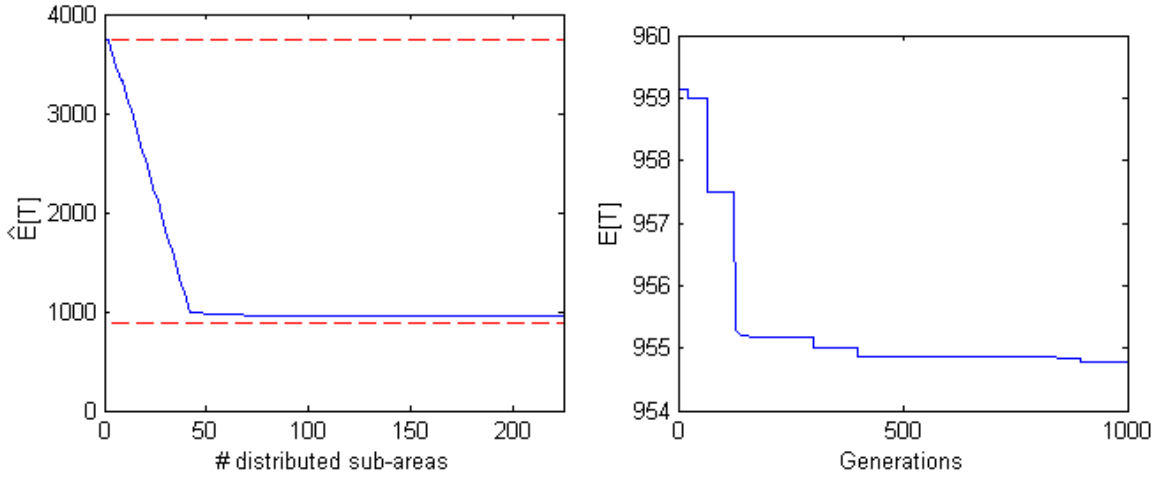

Figure 15: Results for 3 hexacopters on test map B with the nearest neighbor algorithm to the left and the genetic algorithm to the right.

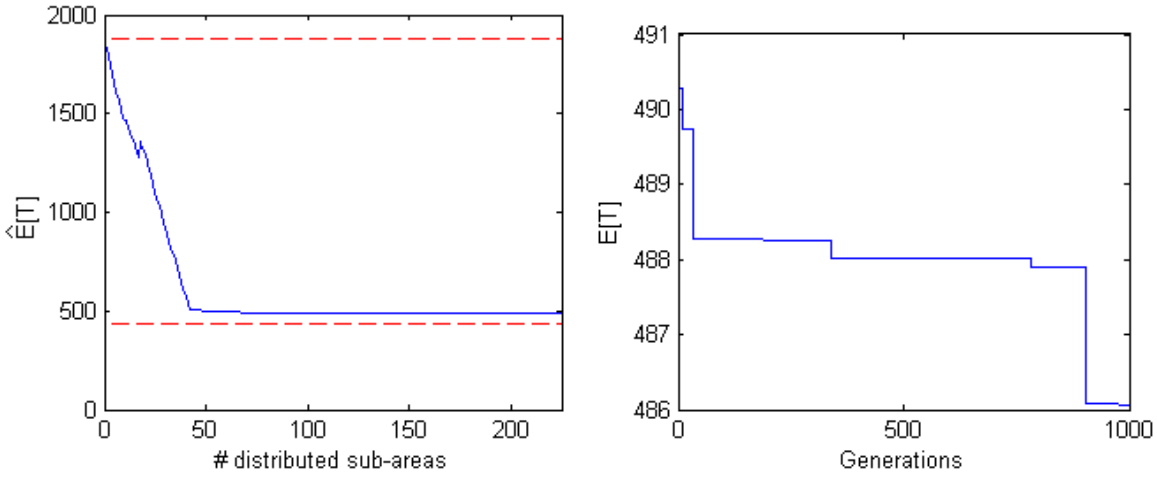

Figure 16: Results for 6 hexacopters on test map B with the nearest neighbor algorithm to the left and the genetic algorithm to the right.

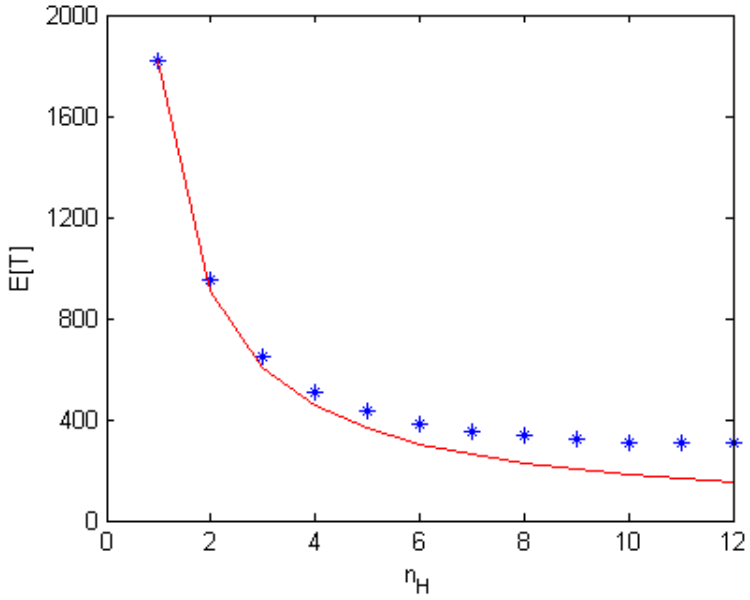

Figure 17: Search time $E[T]$ on test map A depending on the number of hexacopters n_H as blue dots compared to theoretical values, the red line.

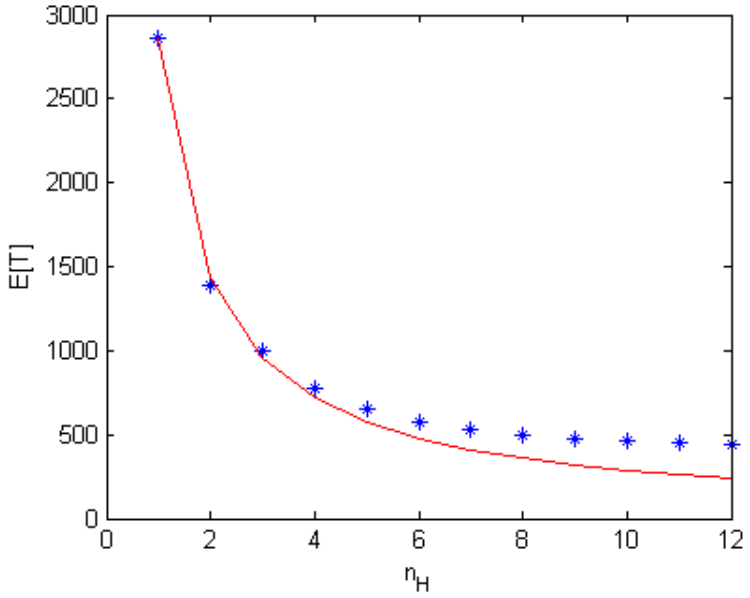

Figure 18: Search time $E[T]$ on test map B depending on the number of hexacopters n_H as blue dots compared to theoretical values, the red line.

7 Discussion

What is desired is a low expectation value of the search time, but one question is if $E[T]$ indeed reflects the expected search time. Mathematically it is correct from the model that was set up with probabilities for every sub-area. But the probabilities are just rough estimations and may be far away from the truth. In some cases the probabilities represent how prioritized that sub-area is rather than the probability to actually find someone there, as in the case for water edges - the probability is very low that a person is found here but one wants to secure these areas. Since the probabilities also represent where it is desired to search, more of a weighted expectation value of the search time is achieved.

Since the estimations are so rough, the best solution to the model might not be the best way to search through the map. Therefore it is unnecessary to find the optimum solution since the inputs to the problem have so large uncertainty. A slightly worse solution can be just as good in practice, and it is therefore not worth to put in a lot of effort to find a perfect solution.

7.1 Nearest neighbor algorithm

As discussed earlier, it has been shown that the nearest neighbor algorithm on average yields a tour 25% longer that the optimum tour and that the loss is usually towards the end of the track. This loss might not be so critical for two reasons; To begin with, the most probable sub-areas are searched first so it is more likely that the person is found at an early stage during the mission. Even though the later part of the track is worse optimized, there is a big chance that the person is already fonud and then this part of the solution is not considered. Moreover, the hexacopters just need to know their next sub-area, not the full track. Therefore the genetic algorithm can optimze during the search and has more time to optimize the end of the track to compensate for the nearest neighbor algorithm.

An interesting phenomena occurs in the left part of Figure 14 where $\hat{E}[T]$ first drops and then then increases again after around 25 shared out sub-areas. The reason why this can happened is because Equation (10), used to calculate $\hat{E}[T]$, assumes no traveling time between the sub-areas that have not yet been assigned. Since the hexacopters need to visit all sub-areas one cannot stop the algorithm here, all sub-areas have to be divided among the hexacopters and when traveling time is added for those sub-areas $\hat{E}[T]$ may increase.

It is necessary to use both NN1 and NN2. NN1 was tested to be about 50% faster but NN2 gives much better results when the hexacopters are not started on the same position, which is the case when restarting the search. Even though the genetic algorithm could execute for a longer time if only NN1 was used, it would not

compensate the results from NN2.

The nearest neighbor algorithms was speeded up even more by considering only the sub-areas with the highest priorities, i.e, to not include all sub-areas into A_R in Algorithm 1 but the x sub-areas with highest probability. If x is half of the total number of sub-areas the speed of the nearest neighbor algorithm is improved by 25%. By only considering probabilities and not the position when choosing these x sub-areas, it might lead to an hexacopter only sorrounded by low priority sub-areas and none of them included in A_R. The hexacopter is then given sub-areas further away even though closer sub-areas whould be better to search. This special case was found very rare so the possible improvment of calculation speed is definitley more important.

7.2 Genetic algorithm

Eight different mutations of the tracks are used since these were the useful number of mutations for this problem. More variants could be used but it would waste calculation capacity. The size of the population is optimized to achieve as good results as possible within the shortest time. If the population is too large, every generation needs too much calculation capacity and the solution cannot be optimized for many generations. Since the probability of a positive mutation is much smaller than a negative mutation, many individuals is needed to improve the chance of having some positive mutations in every generation.

In Figures 13, 14, 15 and 16, one can see how the genetic algorithm does not improve the solution that much after the nearest neighbor algorithm. For example in Figure 13, the nearest neighbor algorithm optimizes from 3700 to 633 and the genetic algorithm continues from there, 633, and optimizes down to only 627. This is due to that the solution is already very good. When there is large a difference in priority values, the sub-areas are more likely to be chosen in an order depending on the priority rather than position to achieve shorter search time. It is easier to order sub-areas in decreasing priority than to achieve the shortest possible track. This is since the priorities are constant during the search while the distances to every sub-area are changing when the hexacopters are moving. Since the priorities have more impact on the search time and this is easier to optimize, the nearest neighbor algorithm delivers a very good solution.

The genetic algorithm optimization is then used to better order the sub-areas within the same priority range to achieve a shorter traveling distance. But the traveling distances between the sub-areas are short compared to the distance to search inside a sub-area. These sub-areas are searched by 10 lanes which means that a hexacopter can travel past more than 10 sub-areas in the same time as searching through one. The time spent on traveling compared to active searching is very small, so optimizing the distance have small impact on the search time. Therefore it will look as

the genetic algorithm does not optimize as much as the nearest neighbor algorithm. If all priority values instead were within a smaller range, then a short traveling distance would have more impact on the search time and the genetic algorithm could optimize more for this than the nearest neighbor algorithm would.

One can see in Table 3 how the search time for the found solution is within 11% from the approximation of the optimium solution for all test cases. This has to be considered as very good, especially when the approximation $\widetilde{E}^*[T]$ is lower than the actuall optimum value $E^*[T]$.

7.3 Cooperation

For perfect cooperation, $E[T]$ should be inversely proportional to n_H, i.e. if the number of hexacopters are doubled the area should ideally be searched twice as fast. In Figures 17 and 18, one can see that the results deviates more from the ideal value when more hexacopters are used. This is expected since it is more complex to organize several hexacopters and they have to travel longer to move past sub-areas assigned to other hexacopters. The theoretical curve in the figures is estimated from the search time for one hexacopter, which then has a high impact on the curve.

7.4 Not considered parameters

One parameter not known during this project is the vision of sight for the hexacopters. That parameter determines the width of the lanes in the sub-area search which then determines the number of lanes in every sub-area. As discussed earlier, the size of a sub-area is important for the achieved results but no optimization of this has been done during the project. For doing this the lane width, number of hexacopters and total size of the search area have to be considered. In this project the side length of a sub-area was set to 100 length units and the width of the field of view for one hexacopter was set to 10 length units which results in 10 lanes per sub-area.

One very important aspect that is not taken into account in this project is the battery charging. Small electrical hexacopters might have a flight time below 15 minutes, which would not cover many sub-areas. When considering charging, more traveling is needed since the hexacopters have to go back and forth to the charging station. With this addition, the optimization problem complexity would be increased and the start solution would be further away from the optimum solution. But as discussed above, the genetic algorithm is good to enhance the track in the sense of traveling distance. When more traveling is included, the genetic algorithm could probably be able to optimize more than it can do for the moment.

8 Conclusions

The nearest neighbor algorithm appears to produce very good solutions, causing the genetic algorithm to be a bit redundant. The algorithm also showed good performance for cooperation with several hexacopters. What is missing here though is the battery charging which probably would change the results and the genetic algorithm could be of much more importance.

9 Future work

For future work on this project, a maximum flight time can be introduced in order to force the hexacopters to fly back and charge. This will probably also include some rework on the algorithm to still achieve good results.

A simplification is that every place is only searched once, if nothing is found it is assumed that nothing will be there later either. An idea could be that the probability of a sub-area is zero right after it has been searched. Then, as time goes on, the probability increases, proportionally fast to its initial value. Eventually this sub-area will have high enough probability to be searched again.

A maximum distance between the hexacopters could be introduced in order to sustain communication between them. This condition will have large impact on the model and it may be hard to even find a solution at all.

As mentioned in the beginning, there are people that are absent by free will and do not want to be found. To find these people the search strategy has to be adjusted by changing the priorities on the map.

References

[1] Rikspolisstyrelsen. *Efterforskning av försvunna personer*. Borlänge: Dala Print Media AB, 2000

[2] Dick Johansson, Rikspolisstyrelsen, March 10, 2014.

[3] Ohlsson, N., Ståhl, M., 2013 *Model-Based Approach to Computer Vision and Automatic Controll using Matlab Simulink for an Autonomous Indoor Multirotor System.* M.Sc. Chalmers.

[4] Fogelberg, J., 2013 *Navigation and Autonomous Control of a Hexacopter in Indoor Environments.* M.Sc. Lund University.

[5] Araujo, J.F., Et al., 2013. Multiple UAV Area Decomposition and Coverage. *IEEE Symposium on Computational Intelligence for Security and Defense Applications*, pp 30-37.

[6] Huang, W.H., 2001. Optimal Line-sweep-based Decompositions for Coverage Algorithms. *Procedings of the 2001 IEEE International Conference on Robotics & Automation,* Seoul, Korea. pp.27-32.

[7] Garey, M.R., 1979. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. First Edition Edition. W. H. Freeman.

[8] Mitchell, M., 1996. *An introduction to Genetic Algorithms.* Cambridge, MA: MIT Press.

[9] Johnson, D.S., McGeoch, L.A.,1997. *The traveling salesman problem: A case study in local optimization*, Local search in combinatorial optimization, 1997, 215-310.

[10] Schaefer, R., 2007. *Foundations of Global Genetic Optimization.* Berlin: Springer.