CHALMERS
UNIVERSITY OF TECHNOLOGY

Deployment Planning for Artillery Hunting Radar Systems Using High-Resolution Geospatial Data

Master's thesis in Complex Adaptive Systems

KEVIN LARSSON

Department of Space, Earth and Environment
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2018

Master's thesis

Deployment Planning for Artillery Hunting Radar Systems Using High-Resolution Geospatial Data

KEVIN LARSSON

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Space, Earth and Environment
Division of Physical Resource Theory
Complex Systems
Chalmers University of Technology
Gothenburg, Sweden 2018

Deployment Planning for Artillery Hunting Radar Systems Using High-Resolution Geospatial Data
KEVIN LARSSON

Supervisor: Håkan Warston, Saab Surveillance
Examiner and supervisor: Claes Andersson, Department of Space, Earth and Environment

Master's Thesis
Department of Space, Earth and Environment
Division of Physical Resource Theory
Complex Systems
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: An ARTHUR unit.[1]

Typeset in LaTeX
Gothenburg, Sweden 2018

Deployment Planning for Artillery Hunting Radar Systems Using High-Resolution Geospatial Data
KEVIN LARSSON
Department of Space, Earth and Environment
Chalmers University of Technology

Abstract

In military mission planning for radar units, good quality geospatial data is of importance as a basis for successful missions. Terrain data with a resolution of 0.5 m, in contrast to often used data with a resolution of 30 m, has recently become available. This comes with opportunities in the form of enabling better quality of planning, and limitations due to increasing execution times. In this thesis, an analysis of methods for utilising high-resolution data in deployment planning for the ARTHUR radar system has been performed. Deployment schedules that are generated should be such that they enable radar units to successfully monitor a defined area of interest, and so that it is difficult for the enemy to localise and attack the units. In addition to an initial analysis, an implementation of the most promising method has been performed for demonstration and further testing.

Three main aspects have been covered: how to evaluate the quality of obtained deployment schedules, how to transport units between successive deployment sites (that are given as input to the system), and how to optimise a series of successive deployment sites that constitute the deployment schedules for single units. In order to evaluate schedules, a modelled probability of a successful mission is taken as objective function. A method that determines which points are traversable in the terrain, and that vectorises the resulting raster by a presented skeletonisation method, is presented for finding routes for transportation between deployment sites. For optimising schedules, a genetic algorithm is used.

Results show that the proposed method is working reasonably well under the given circumstances and that it is flexible and can handle adjustments of the objective function. Part of the input data has however been synthetic, and better data should be used for further analysis, which is required if the method should be utilised in a real-life setting. A number of simplifications have been made during the project, and possible alterations to account for these are proposed and discussed, as well as other improvements and alternative approaches.

Keywords: military mission planning, genetic algorithm, route planning, skeletonisation

Acknowledgements

I would like to thank everyone that have made my studies in general, and this project, possible and enjoyable. This involves everyone at Saab Surveillance in Gothenburg that has contributed to making this project happen, and my supervisor Håkan Warston in particular. Furthermore, I want to thank Vricon in Linköping and Saab Bofors Dynamics in Karlskoga that are part of the collaboration which includes this project. Particularly, I would like to thank Joachim Wickman at Saab Bofors Dynamics for taking the time and providing useful insights. Also, my supervisor at Chalmers, Claes Andersson, should be mentioned for his contributions and the valuable discussions we have had.

All the friends I have made during the last five years deserve a lot of credit for all the fun times we have had. I will never forget the warm reception and togetherness I have felt at the university since that moment I first walked up to Götaplatsen in 2013. A special thank you goes out to everyone involved in the associations I have been a part of, and to everyone partaking in our late-night studying sessions in Nästet. Last, but not least, I would like to thank my family for their unconditional support.

Kevin Larsson, Gothenburg, November 2018

Contents

List of Abbreviations

ACO ant colony optimisation

DA deployment area

DSM digital surface model

ELINT electronic intelligence

ESM electronic support measures

GA genetic algorithm

GPS Global Positioning System

GUI graphical user interface

MGRS Military Grid Reference System

OpenCV Open Source Computer Vision Library

OSM OpenStreetMap

TAI target area of interest

UAV unmanned aerial vehicle

UPS Universal Polar Stereographic

UTM Universal Transverse Mercator

WGS 84 World Geodetic System 1984

List of Algorithms

List of Figures

List of Tables

1

Introduction

At Saab Surveillance in Gothenburg, a variety of radar systems are developed. The purposes of systems vary, but include weapon location and fire control. At the business unit Surface Radar Systems, the product range consists of radar systems for land and naval usage. One of the products is the land-based system ARTHUR, an abbreviation for **Art**illery **Hu**nting **R**adar, whose primary capability is to detect firings from hostile artillery and track the identified projectiles' trajectories.

When planning a military operation that includes land-based radar units such as ARTHUR, it is common that a number of tactical restrictions are given beforehand. The mission is usually to monitor a certain area where the enemy is likely to be operating, called target area of interest (TAI). Furthermore, a deployment area (DA) where the radar units are constrained to operate themselves is generally defined in advance. In order to successfully complete a mission, radar units should be positioned in the DA so that it has a good view over the TAI. Essentially, a unit should have a free line of sight to as large a proportion of the airspace just above the TAI as possible, so that there is no terrain, vegetation, building or other object obstructing the view.

Apart from choosing deployment sites where TAI coverage is good, the enemy's ability to track the position of the unit must be considered. An enemy can typically gain a substantial advantage in combat if it can reduce one's surveillance capability by tracking and taking out radar units. As any operating radar transmits signals into the open air, the probability that enemy electronic intelligence (ELINT) or electronic support measures (ESM) will intercept and locate the signal source is high. In order to ensure the survivability of a unit, it can therefore typically only operate at a single site for a limited amount of time; with that time being determined by the enemy's assumed ability to locate and attack the position.

During a mission, a unit thus typically operates at a site during a given amount of time, and then moves to a new position from where the process is repeated. In order to have continuous monitoring of a TAI during a mission, one must then use several units such that a new unit starts operating from a new location as soon as the current unit ceases operation. Challenges when planning a mission thus includes coordinating times of operation for each unit, determining what deployment sites to use and when to use them, as well as deciding how to transport units between subsequent deployment sites.

To successfully plan a mission, one must then have access to all needed information concerning the areas of interest and the mission. An important part of this

information is high-quality geospatial data such as elevation and types of terrain at different locations. Until recently, the data available for planning applications has had a resolution of at best 30 m between data points, as in the case with the often used DTED level 2 data.[2] However, Vricon (that is co-owned by Saab and DigitalGlobe)[3] can now provide data on terrain elevation and classification where there are only 0.5 m between data points. This means that the existence of obstacles such as trees, smaller buildings and trenches can now be known during the planning stage of a mission.

Utilisation of such high-resolution data can possibly improve the quality of planning application output, since fewer surprises should appear in-field when the available information is more detailed from the beginning. Usage of data with higher resolution comes with a few problems however, since execution times in applications typically increase with the amount of data that is processed. It is therefore important to analyse how high-resolution geospatial data can be implemented in future planning applications while maintaining reasonable execution times, which is the topic of this thesis.

1.1 Aim

The primary objective of this project is to investigate how high-resolution geospatial data can be utilised to plan a surveillance mission for the ARTHUR system.

Apart from analysing different methods that are based on such data, creating a proof-of-concept Java application that implements the most promising methods from the analysis has been considered the primary product of the project. Furthermore, that application is desired from Saab's perspective as it will serve as a possible basis for a future implementation in their systems.

1.2 Limitations

During the project, deployment planning has only been performed for one unit at a time, which is motivated in the third and fourth paragraphs of Chapter 4. Furthermore, the units' possible speeds are set to constant, with one value for traversal of the terrain and one (larger) value for traversal of roads. Discussion on how to remove these limitations is provided in Sections 6.2.2 and 6.3.2.

The need for communication with own troops and allies has also been neglected, as well as the need for refuelling. These simplifications are discussed in Section 6.6.

1.3 Report outline

In Chapter 2, some background to the project is given. Sections of Chapter 2 that covers concepts which the reader has prior knowledge about can be skipped, and referenced when needed.

Next comes Chapter 3 about modelling a scalar-valued objective function to use for evaluating obtained deployment schedules. Chapter 4 describes the proposed method that is the outcome of the performed analysis on methods to use.

Further on, Chapter 5 describes the Java implementation that has been done as a proof-of-concept for the proposed method of Chapter 4. Chapter 6 then elaborates on possible further development of the project, other approaches that can be taken to solve the problem, and the possibility of utilising the developed concepts in future versions of planning applications. Lastly, Chapter 7 wraps up the discussion and concludes the most important aspects of the project.

2

Background and Theory

In this chapter, some central concepts will be presented in order to give the reader a solid background before continuing to subsequent chapters that describes what has been accomplished during the project. Some background about Saab and relevant components that stems from Saab is presented in the first section. Then there are two sections about map data before continuing with theoretical sections about techniques that have been used in the project.

2.1 Saab

Saab is a Swedish company that develops and manufactures products and services for the defence and security markets. One of Saab's six business areas is Saab Surveillance that develops a number of land-based, naval and airborne radar systems, as well as military command and control systems. Out of these products, the land-based radar system ARTHUR is of interest for this thesis and will be described in the next section. Additionally, the resources that are provided through Saab and the background to the thesis project from Saab's point of view will be described.

2.1.1 ARTHUR

The primary capability of the radar system ARTHUR is to detect artillery fire in the airspace in two operational modes: weapon location and fire control. As artillery fire is detected, it can follow the trajectory of the projectile and calculate where it was fired from (point of origin) and where it will hit (point of impact). In that way, the location of the enemy's weapon can be found (weapon location mode). Additionally, ARTHUR can follow own troops' artillery projectiles and follow up on whether they hit the desired point of impact (fire control mode).

In order to detect artillery fire as early after firing as possible, ARTHUR scans the horizon and monitors the area just above it. When a projectile is detected, the system can quickly track its ballistic trajectory and estimate the point of origin and point of impact early, in order to take action within seconds.

When arriving at a site, ARTHUR takes some time to deploy before it can start monitoring the airspace. In the same manner, some time is required before the unit can start moving again after operation is ceased.

The ARTHUR unit is contained in a standard type of shelter, meaning it can be transported by a variety of vehicles with a large enough payload. Therefore, the ability with which the system can be transported is largely dependent on the transporting vehicle. Possible types are regular trucks that can move efficiently on roads as well as tracked vehicles that can move efficiently in more complicated terrain. A typical configuration with a track-based vehicle can be seen in Fig. 2.1.

Figure 2.1: An ARTHUR unit mounted on a track-based vehicle where the radar system's shelter can be observed on the rear part, hitched to the pulling bandwagon.[1]

2.1.2 Vricon data

Vricon has developed a digital surface model (DSM), containing high-resolution height data for the terrain in the area it covers.[4] Contrary to many other height data models, it includes permanent objects above ground, such as trees and buildings, in addition to the height of the ground itself. It consists of data points in a grid with a resolution of 0.5 m between data points, allowing for inclusion of relatively small objects. The detail of this model is to be put in contrast to the DTED level 2 data[2] that is often utilised, where the resolution is at best 30 m between data points and where vegetation and buildings are not accounted for.

Additionally, there is classification data with the same resolution as in the DSM available, indicating types of terrain. The data classifies the terrain into seven different classes, as presented in Table 2.1. However, it is expected that this classification will be extended and improved in the near future. Nevertheless, the currently available classification data in combination with the DSM do constitute higher quality geospatial data than has previously been available.

2.1.3 Earlier work

This Master's thesis is one of four thesis projects that are part of a larger project, concerning utilisation of high-resolution geospatial data. The full project is a collaboration between Saab Surveillance in Gothenburg and Halden, Saab Bofors Dynamics in Karlskoga and Vricon in Linköping.

Table 2.1: Different classes in Vricon's current data, corresponding to identified objects and terrain types at the surface.

Classification	Description
Uncertain	
Ground	Grass, dirt etc.
Low vegetation	0.5-2 m
Medium vegetation	2-5 m
High vegetation	> 5 m
Man-made structures	Buildings etc.
Water	
Paved surface	Roads, parking lots etc.

The first thesis project took place during the spring term of 2018; Fredrik Aas Isaksen and Kim Nilsen Brusevold wrote the thesis for their Bachelor's degree from Østfold University College at Saab Surveillance in Halden.[5] They examined different algorithms for computing viewsheds from points in a given DA. Furthermore, they wrote partially completed Java code for finding good deployment sites in a given DA, with respect to a specific TAI. Parts of the code they produced have been taken as a starting point in this project.

2.2 OpenStreetMap

OpenStreetMap (OSM) is a map database that relies on user-contributed data. It is supported by the non-profit OpenStreetMap Foundation that maintains servers with data and raises funds for the project. The data content of OSM can be accessed through several APIs for usage in different applications, as well as directly through the OSM website[1].

In the database, there are three types of elements: nodes, ways and relations. Nodes represent single spatially defined points, whereas ways connects several nodes to represent one-dimensional structures such as roads or boundaries of areas. Relations establish a relationship between data elements that can be nodes, ways or other relations.

All three element types can have tags that convey information of what the elements represent. A tag consists of a key and a value, that in combination represent some information related to the object the tag belongs to. A typical tag on a way that represents a road in an urban area is 'highway=residential', where 'highway' is the key and 'residential' is the value.

[1] `https://www.openstreetmap.org/`

2.3 Geographic Coordinate Systems

To represent locations on earth, there are several systems that use different types of coordinates to indicate positions. The typical way to uniquely specify a location on the surface of the earth is to give its latitude and longitude in degrees.[6] The latitude specifies the north-south position so that it takes the value 90° at the poles and 0° at the equator, with the suffix N or S denoting if the location belongs to the northern or the southern hemisphere. Similarly, the longitude specifies the west-east position from 0° in Greenwich, England to 180° at the opposite side of the earth, in the Pacific ocean. Whether the point lies in the western or eastern hemisphere is specified by a suffix W or E. For computational purposes, the latitude is often defined as negative in the southern hemisphere and the longitude is defined as negative in the western hemisphere, allowing omittance of the suffixes.

2.3.1 WGS 84

Since the shape of the earth is irregular and not a perfect sphere, the altitude at a position given by its longitude and latitude is not trivial to calculate. Throughout history, several standards that define altitude have been used. A commonly used system that was created in an attempt to standardise the modelling of the earth is World Geodetic System 1984 (WGS 84).[7] It approximates the surface of the earth as an ellipsoid and measures altitude as the height above the ellipsoid. Given longitude, latitude and altitude as defined by WGS 84, locations on the earth can be uniquely defined.

WGS 84 has become a popular standard and is used in the widespread Global Positioning System (GPS) among many other applications. More importantly, the data from Vricon and OSM are defined using WGS 84.

2.3.2 UTM and MGRS

A problem with specifying coordinates in the latitude-longitude representation is that length is not a simple linear function of latitude and longitude. In other words, the length of one longitudinal degree is dependent on the latitude so that it is maximal at the equator and zero at the poles. The same goes for the latitude itself, since the shape of the earth is badly represented by a sphere. If the observed area is far from the poles and rather small compared to the size of the earth, lengths in north-south and west-east direction can be approximated as linear functions of latitude and longitude. However, another coordinate representation system is useful if one wants to represent several areas from different parts of the world equivalently, with equal coordinate scaling independent of where the area is positioned.

Several alternative coordinate representations that are suitable for different scales of operation exist. A popular such system is the Universal Transverse Mercator (UTM) coordinate system that is based on the WGS 84 ellipsoid and the Mercator projection.[8] It divides the world along every sixth meridian (longitudinal degree)

into sixty zones, labelled as zone 1 through 60. In each zone, coordinates are given as an easting (x-coordinate) and a northing (y-coordinate). The easting specifies the distance in meters along the west-east direction to the central meridian of the zone, and the northing specifies the north-south distance to the equator in meters. In order to avoid negative values, the easting is defined as 500,000 m at a zone's central meridian. The northing at the equator is defined as 0 m when observing points in the northern hemisphere and 10,000,000 m when observing points in the southern hemisphere. This contributes with the need to specify which hemisphere is observed to uniquely specify a point.

Since the Mercator projection gets more distorted the closer to the pole you are, UTM is only defined between 80 °S and 84 °N. In the polar regions, the Universal Polar Stereographic (UPS) coordinate system is used in order to completely cover the earth in conjunction with UTM.[8] An extension of UTM and UPS is the Military Grid Reference System (MGRS)[9] that further divides the UTM zones along every eighth parallel (latitudinal degree), into 20 zones from 80 °S to 84 °N where the last zone extends a further four degrees north. These zones are enumerated alphabetically from 'C' to 'X', omitting 'I' and 'O' due to the letters being similar to numbers.

Along with the UTM zone division, the earth is thus divided into $60 \times 20 = 1200$ zones when excluding the polar regions. Each zone is then defined as the zone number followed by a letter. For example, the zone that covers the largest part of southern Sweden is '33V'. UTM and MGRS are often used interchangeably, so that a coordinate in UTM is specified by its corresponding MGRS zone instead of its UTM zone and hemisphere.

2.4 Image Processing

In order to process map data in raster format, knowledge of some image processing techniques is required. The methods that are presented in this section operate exclusively on binary images. Except for the last part of this section that concerns connected components of binary images, the presented methods are morphological operations.[10]

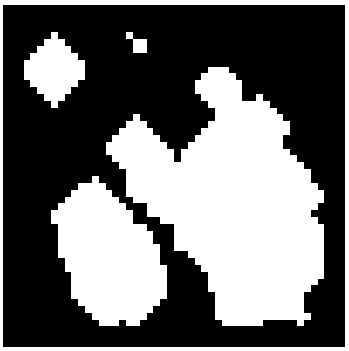
A binary image I is a function $I : (i, j) \to \{0, 1\}$ where (i, j) are the indices of the pixels in the image, with i representing the i:th column of the image and j representing the j:th row of the image. Thus i can take values in $\{1, 2, \ldots, m\}$ and j can take values in $\{1, 2, \ldots, n\}$ if the image is m pixels wide and n pixels high. A pixel taking the value 1 represents the Boolean true value, while a pixel taking the value 0 represents the Boolean false value. Areas taking the value 1 are commonly called foreground, with areas taking value 0 being called background.

2.4.1 Dilation and erosion

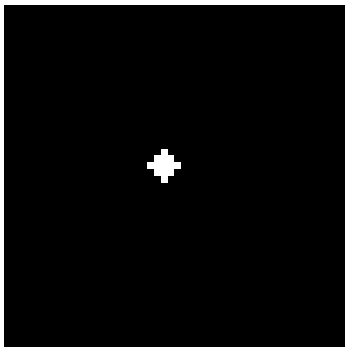
Two basic morphological operations are those of dilation and erosion. Dilation and erosion of the foreground area A of a binary image I are performed with a structural

element $\mathcal{S}$ of the same dimensionality as A.

By using the structural element $\mathcal{S}$ on an area A, dilation (denoted $A \oplus \mathcal{S}$) enlarges A, while erosion (denoted $A \ominus \mathcal{S}$) shrinks A. It does so by centring $\mathcal{S}$ at every single point of A, marking the corresponding pixel in the dilated or eroded image according to the image values at all points that are covered by the structural element. In the case of dilation, the centred pixel is marked as true in the dilated image as long as any pixel inside the structural element is true. Similarly, all pixels that are covered by the structural element must be true for the centred pixel to be true in the eroded image. Examples of dilation and erosion with a structural element $\mathcal{S}$ can be seen in Fig. 2.2.



(a) Input binary map.



(b) Shape of the structural element.

(c) Eroded binary map.

(d) Dilated binary map.

Figure 2.2: Examples of applying erosion and dilation on a binary map where white pixels correspond to foreground and black pixels correspond to background. It can be observed that the foreground is larger after dilation and smaller after erosion, and that the shapes of edges in the resulting maps resemble the shape of the structural element.

2.4.2 Opening and closing

Utilising the definitions of erosion and dilation, morphological opening and closing can be defined. Opening an area A with a structural element $\mathcal{S}$ is equivalent to first eroding A with $\mathcal{S}$ and then dilating the resulting area with the same structural

element $\mathcal{S}$. By taking the erosion and dilation operators in the opposite order, the closing of the area is instead obtained. Specifically, opening ($\circ$) and closing ($\bullet$) are defined by the erosion and dilation operators as

$$\begin{aligned} A \circ \mathcal{S} &= (A \ominus \mathcal{S}) \oplus \mathcal{S}, \\ A \bullet \mathcal{S} &= (A \oplus \mathcal{S}) \ominus \mathcal{S}. \end{aligned} \tag{2.1}$$

Using the opening operator on the foreground of a binary image typically results in small foreground areas becoming background. The opposite generally holds for closing, so that small background areas are incorporated into the foreground, and so that nearby foreground areas become connected. Also, the shape of the resulting areas most often resembles the shape of the structural element as well as the original area. Examples of opening and closing of a binary image is given in Fig. 2.3.

(a) Opened binary map. **(b)** Closed binary map.

Figure 2.3: Examples of applying the opening and closing operators to the same binary input map, and with the same structural element, as in Fig. 2.2. White pixels represent foreground while black pixels represent background. It can be observed that edges are a bit smoother and that the smallest foreground area has disappeared in the opened image. In the closed image, it can be seen that two nearby foreground areas have been merged into a larger area.

2.4.3 Morphological skeleton

A morphological skeleton of a binary image is a thinned-out version of the binary image, such that it consists of one-pixel wide strokes. However, the topology of the original binary image should be preserved to an as large extent as possible, meaning the skeleton should 'span' the entire original image. There is not a uniquely defined way to obtain a morphological skeleton, but rather a plethora of algorithms that can be used to generate skeletons.[11]–[14] In Fig. 2.4, the skeleton of a binary map can be observed, where Matlab's built-in skeletonisation algorithm has been used.

(a) Input binary map. **(b)** Resulting skeleton.

Figure 2.4: An example of a morphological skeleton of a binary map, where white pixels correspond to foreground and black pixels correspond to background. Matlab's built-in skeletonisation algorithm has been used to generate the skeleton.

2.5 Shortest-Path Algorithms

When searching for shortest paths between points on the earth's surface, we observe the terrain as an undirected graph. Mathematically, an undirected graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ is defined by a set of nodes $\mathcal{V}$ and a set of edges $\mathcal{E}$, where the edges connect nodes pairwise. Enumerating the sets as $\mathcal{V} = \{v_1, \ldots, v_m\}$ and $\mathcal{E} = \{e_1, \ldots, e_n\}$, all edges can be defined by a tuple (v_i, v_j) and a length l_k for $i, j \in \{1, \ldots, n\}, i \neq j$ and $k \in \{1, \ldots, m\}$.

A straight-forward way of defining a graph is from a grid. In a grid where some pixels are traversable and some are not, traversable pixels are defined as nodes, and edges are added between any pair of neighbouring traversable pixels.

Another common basis for defining a transportation graph is a road network. In that case, all junctions or road endings are defined as nodes and connected with edges according to where there are actual road segments.

Seeking the shortest path between two nodes v_a, v_b in either type of graph is then equal to the problem of minimising the edge length sum of a set of edges that connects v_a and v_b. Worth noting is that the lengths l_k in a graph might correspond to other metrics than distance. In this thesis, we will typically seek to minimise the travel time between nodes, and edge lengths are in that case measured in units of time.

2.5.1 Dijkstra's algorithm

Probably the most famous algorithm for finding shortest paths in graphs is Dijkstra's algorithm[15] that utilises a breadth-first search algorithm. Given a graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, Dijkstra's algorithm calculates the shortest paths from a given source node $v_s \in \mathcal{V}$ to all other nodes in $\mathcal{V}$.

Starting at v_s, it visits one node at a time and adds it to a set of visited nodes. In every step, it adds the yet unvisited node that has the shortest path from the source

node. It does so by keeping a list of all nodes that are joined by an edge to a node in the visited set, but are not in the visited set itself. By storing the path lengths to all nodes in the visited set, and adding the length of the edge leading to a node outside the set, the node with shortest such metric can be added in every iteration. The shortest path to all nodes in the graph will thus be found and stored when the algorithm terminates. In the case that only a single path between a source node v_s and a target node v_t is desired, the algorithm can be terminated as soon as v_t is encountered.

2.5.2 A* algorithm

In searching for a single path between a source node and a target node, Dijkstra's algorithm can often be refined into the A* algorithm[16] to improve execution speed. The A* algorithm is dependent on the availability of a heuristic, a lower bound on the path length from any node to the target node. Equivalently to Dijkstra's algorithm, one can then add nodes to the visited set in the order of growing heuristics, rather than growing path length from the source node. This will typically favour nodes in the direction of the target node and avoid exploration of nodes in the wrong direction.

If the nodes in a graph are spatially defined and the distance between nodes is of interest, a lower bound on the distance between two points is the straight-line distance. For minimising travel time, the lower bound can instead be taken as the straight-line distance divided by the maximum possible speed of traversal.

2.6 Genetic Algorithms

The underlying idea of genetic algorithms (GAs) relies on the concept of natural selection, as described by Charles Darwin.[17]–[19] In a GA implementation, the set of variables that should be set to maximise fitness (an objective function) is encoded as a 'chromosome'. The chromosome typically takes the form of a data array with entries of a data type that is suitable for the problem. Two usual entry types are Booleans (true/false) and integers confined to some problem-specific interval.

Given a set of chromosomes, called a population, one then applies some genetic operators on the population in order to create a new generation of the population. Common operators are crossover and mutation, whose equivalences in nature are mating and mutations. New generations are then created iteratively by the means of the genetic operators, in hope that the overall fitness will increase.

In creating a new generation, chromosomes are randomly chosen in pairs, such that the probability of selection increases with fitness. Crossover is then performed on the two chosen chromosomes by combining them into two new chromosomes. The resulting chromosomes are then mutated independently by randomly altering some elements of the chromosome. After mutation, the new chromosomes are inserted into the new generation and the process of selection, mutation and crossover is repeated

until the new generation consists of as many chromosomes as the old generation, a number usually called the population size.

The details on how crossover and mutation are performed can vary a lot between different GAs, and additional types of genetic operators might be utilised as well. Depending on the type of problem and the encoding of chromosomes, operators have to be adjusted to yield valid chromosomes and interpretable results. Also, even if the chromosomes generated by the operators are valid, they might not yield good results. For most operators, there are also a number of parameters that need to be set by the user. Depending on the problem, there might be a lot of tweaking of those parameters needed for a GA to be successful.

2.7 Software

The software that has been used during the project is primarily Matlab[20] and Java.[21] During the first half of the project, Matlab was utilised to test out different methods and observe available data. During the second part of the project, Java was used to build the test implementation described in Chapter 5 and for further testing.

Additionally, Open Source Computer Vision Library (OpenCV)[22] has been used for the test implementation. It is a software library written in C++ with an extensive set of image processing functions. It relies on its basic data format `Mat` that represents single- or multi-layered images, or two-dimensional arrays. It offers interfaces allowing it to be used from a number of different programming languages, including a Java interface that supports most of the functionality that is contained in the native C++ version.

3

Objective Function Modelling

Given that a surveillance mission plan has numerous desired properties, it is necessary to examine in detail how one should weigh the different properties against each other. When it comes to optimisation, this equals finding a way to formulate a scalar-valued objective function to be minimised or maximised.

The ultimate goal for the surveillance mission is to successfully monitor and detect all threats in a TAI during the time frame of the mission, while no unit is successfully tracked and taken down by the enemy. Therefore, the probability p_{success} of a successful mission can be written as the product

$$p_{\text{success}} = p_{\text{survival}} \times p_{\text{coverage}}, \tag{3.1}$$

where p_{survival} is the probability to survive the full duration of the mission, and p_{coverage} is the probability to successfully detect all enemy threats while completing the mission.

Using this knowledge, a natural choice of objective function for the optimisation is p_{success}. Since neither the true value of p_{survival}, nor the true value of p_{coverage}, is known during the planning stage of a mission, models that estimates these probabilities reasonably have to be defined in order to be able to evaluate obtained schedules. Definitions of the probability models that have been developed in the project are presented in the following sections.

3.1 Modelling Survival Probability

The probability to survive for the whole duration of a mission can be seen as a product of the probabilities to survive during each step of the mission. Each time that a unit is deployed at a site, and each movement of a unit in-between such sites will be considered steps of the mission in this context. The number of deployments of a unit is denoted with N in this context, yielding $N-1$ routes to be taken between deployment sites.

Unless the mission planning has been leaked to the enemy beforehand, future actions taken by a unit cannot affect the probability of survival in earlier steps. Therefore, the probability of survival during a step of the mission will be considered independent of all subsequent steps. However, dependence on preceding steps should be taken into account and should naturally influence the survival probability.

The three identified ways that a unit can be located by the enemy is by the usage of ESM when a unit is operating at a deployment site, by visual detection when standing on a deployment site and by visual detection during transportation of a unit. The respective models for the probabilities $p_{\text{ESM}}, p_{\text{vis}}, p_{\text{tr}}$ of avoiding the three types of localisation during the whole mission is presented in the following parts of this section. When obtained, they can be combined to form the total probability of survival as

$$p_{\text{survival}} = p_{\text{ESM}} \times p_{\text{vis}} \times p_{\text{tr}}. \tag{3.2}$$

3.1.1 Localisation by ESM

The enemy's ability to operate ESM equipment should greatly influence the modelled risk of being localised by such a system. However, knowing the details of an enemy's all capabilities would be considered an exception, even though it can be expected that most enemies operate some kind of ESM system. Nevertheless, an example of a factor that can be estimated is the visibility of one's deployment site from sites where an ESM system can possibly be placed by the enemy. A complicating circumstance for the enemy is that two ESM systems in separate locations are generally required to locate a transmitting radar, since it is only the direction of transmitted signals that can be obtained by a single ESM unit. Therefore, cross-bearing is utilised to locate the transmitting radar when the direction towards the radar from two separate ESM units are found.

Using the available knowledge about the enemy's general capabilities for using ESM, the risk r_i^{ESM} of being spotted at a site i when staying there indefinitely or returning there repeatedly can be estimated. The probability p_i^{ESM} of avoiding localisation by ESM at site i can then be modelled as

$$p_i^{\text{ESM}} = (1 - r_i^{\text{ESM}}) + r_i^{\text{ESM}} \times f_{\text{ESM}}, \tag{3.3}$$

where $f_{\text{ESM}} \to [0, 1]$ is a function of how the unit operates.

In modelling f_{ESM}, information about how the unit has operated throughout the mission thus far should be taken into account, notably concerning earlier visits to the same or nearby deployment sites. It should also be dependent on the time τ_i^{send} that the unit operates at the site. Denoting the number of earlier visits to the same site with n_i, the two required properties given in the definition of r_i^{ESM} can be written as

$$\lim_{\tau_i^{\text{send}} \to \infty} f_{\text{ESM}} = 0 \text{ and } \lim_{n_i \to \infty} f_{\text{ESM}} = 0. \tag{3.4}$$

Apart from being dependent of τ_i^{send} and n_i, f_{ESM} is implemented with two additional properties influencing its value. If the relevant site has been visited earlier in the schedule, f_{ESM} takes smaller values the shorter the time is since the previous visit to the site. Also, f_{ESM} is implemented so that it decreases with long repeated patterns in the schedule. This is since movement patterns should not be predictable for the enemy. For example, if the unit always moves to site 'B' after each visit to site 'A', there is a risk that the enemy will realise this, and site B will be attacked with a larger probability.

As probabilities for all visits to different deployment sites i during a mission are obtained, the probabilities can be multiplied to obtain the total probability of avoiding detection by ESM during the mission as

$$p_{\text{ESM}} = \prod_{i=1}^{N} p_i^{\text{ESM}}. \tag{3.5}$$

3.1.2 Visual detection on-site

The risk of being detected visually by the enemy on a deployment site is influenced primarily by the visibility of the deployment site from different positions. On one hand, visual detection by the naked eye must be considered, possibly aided by binoculars or other equipment. On the other hand, recording equipment can be used, possibly mounted on an unmanned aerial vehicle (UAV) or other types of vehicles. Similarly to the case of detection by ESM as described in the previous section, it is only the enemy's general capabilities for visual detection that can be estimated. Therefore, the probability p_i^{vis} of not being visually detected at site i is modelled as

$$p_i^{\text{vis}} = (1 - r_i^{\text{vis}}) + r_i^{\text{vis}} \times f_{\text{vis}}. \tag{3.6}$$

Here, r_i^{vis} is equivalent to r_i^{ESM} in the previous section, and corresponds to the risk of being localised visually if staying at a site indefinitely. Also, f_{vis} is a function of how the unit operates that takes values in $[0, 1]$. Except for different parameters that models the differences between ESM and visual detection, the main difference to the ESM case is that f_{vis} is dependent on the time τ_i spent on-site rather than the sending time τ_i^{send} as for ESM. Equivalently to the case of ESM and (3.4), this means that the requirements

$$\lim_{\tau_i \to \infty} f_{\text{vis}} = 0 \text{ and } \lim_{n_i \to \infty} f_{\text{vis}} = 0. \tag{3.7}$$

must hold. Also, f_{vis} is implemented to decrease with shorter times between visits to the site, and with longer repeated sequences of deployment site visits.

Just as in the case of localisation by ESM, the total probability of avoiding visual detection on-site during a mission is then obtained as a product

$$p_{\text{vis}} = \prod_{i=1}^{N} p_i^{\text{vis}}. \tag{3.8}$$

Additionally, the model could be enhanced by knowledge of the general visibility of deployment sites. For example, a site on an open plain is generally visible from a large set of points, both from the ground and the air, while a site in a narrow valley that is covered in woods is generally not visible from as many points. Such data is however not available and even though it can possibly be estimated using the DSM data, it has been considered outside the scope of this project.

3.1.3 Visual detection during transport

In addition to the risk of being detected while standing on a deployment site, the risk of being detected during transport in-between sites has to be considered. In this case, the units are not transmitting any radar signals and the risk of detection by ESM does not have to be accounted for.

Modelling this risk is intricate, since every single point on the routes travelled between sites should be taken into account. Different points along the routes might have different properties affecting the risk of detection. As it comes to repeating patterns and not wanting to traverse a point over and over again, the fact that several routes might intersect have been considered. This is so that traversing a road segment or junction more times than necessary is avoided.

The suggested model of the probability p_j^{tr} of avoiding detection during transportation along a route between two deployment sites is

$$p_j^{\text{tr}} = (1 - r_j^{\text{tr}}) + r_j^{\text{tr}} f_{\text{tr}}, \tag{3.9}$$

similar to the models for detection on-site. The model has been chosen so that it is not too complicated to evaluate, while using as much information as possible.

Once again, r_j^{tr} is the probability of not being detected when travelling on the route indefinitely and f_{tr} is a function of the route and earlier traversals that takes values in $[0, 1]$. However, f_{tr} should not only take earlier traversals of the exact same route into account, but also intersections with other routes that have been traversed.

A complete evaluation including all points that are travelled over would require integration over all travelled routes and could very quickly become computationally heavy. Therefore, when observing routes in a defined transportation graph, intersections of routes can be accounted for by looking at common edges and nodes along said routes. This will yield an approximation, but since numerical integration is an approximation as well, it can be justified by a transportation graph that is not too coarse.

Factors that have an impact on the value of f_{tr} are the elapsed time since the previous visit to each node in the route, the number of earlier visits to each node, and how exposed each node's location is to the enemy. An average of these three factors is then used in evaluating f_{ESM}.

As the probabilities of avoiding detection are obtained for all routes, the total probability of avoiding visual detection during transport for the whole mission can be calculated as

$$p_{\text{tr}} = \prod_{j=1}^{N-1} p_j^{\text{tr}}. \tag{3.10}$$

3.2 Modelling Full Coverage Probability

When monitoring a defined TAI from a given deployment site, there is a certain probability that a threat (enemy firing) will occur during the time that the monitoring is ongoing. Since it is difficult to determine beforehand when an enemy will

fire a weapon from the TAI, the time distribution of enemy firings is modelled by an exponential distribution.[23] Under that assumption, it is only necessary to estimate the mean interval between enemy firings to know the probability of the enemy firing inside a given time interval, since the exponential distribution is memoryless.

If it is further assumed that all points in the TAI are equally probable spots for the enemy to fire from, the probability p_i^{coverage} of detecting all enemy firings from the TAI during a given time frame scales with the proportion q_i^{cov} of the TAI that is covered. If it is then assumed that a proportion p_{find} of the enemy firings from locations that is said to be monitored are truly detected, it is straight-forward to define a model for the probability that we detect all enemy firings from the TAI during a time slot of length τ_i. Carrying out the calculations, we obtain

$$p_i^{\text{coverage}} = \exp\left(-\frac{\tau_i}{\tau_0}\right) + \left(1 - \exp\left(-\frac{\tau_i}{\tau_0}\right)\right) \times p_{\text{find}} \times q_i^{\text{cov}}, \tag{3.11}$$

where $\tau_i \ll \tau_0$ is assumed. The first term in this expression equals the probability that the enemy does not fire during the time frame. Consequently, the first factor of the second term corresponds to the probability of the enemy firing at least once.

When the unit is deployed N times in total during the mission, the total probability of detecting all threats during its time of operation becomes

$$p_{\text{coverage}} = \prod_{i=1}^{N} p_i^{\text{coverage}}. \tag{3.12}$$

3.3 Prioritisation Modes

Depending on the situation, surveillance missions might have different desired properties. In some cases, survival of units might be more important than other times. In other cases, surveillance capacity might be so important that larger risks concerning survival can be taken.

Therefore, a modification that will be called prioritisation mode is added, that rebalances the relation between survival and full coverage probability. Instead of taking Eq. (3.1) as objective function, one takes the same expression but with either p_{survival} or p_{coverage} exponentiated with a scalar value $\alpha > 1$. The optimisation will be said to be in survival prioritisation mode when exponentiating p_{survival}, and coverage prioritisation mode when exponentiating p_{coverage}. If the objective function is then denoted with f_{obj}, it will be defined as

$$\begin{aligned} f_{\text{obj}} &= (p_{\text{survival}})^{\alpha} \times p_{\text{coverage}} \quad \text{and} \\ f_{\text{obj}} &= p_{\text{survival}} \times (p_{\text{coverage}})^{\alpha} \end{aligned} \tag{3.13}$$

in survival and coverage prioritisation mode, respectively.

4

Proposed Optimisation Method

In the proposed method contained in this chapter, the problem of optimising radar unit routes is divided into three parts. The first part is preprocessing the data needed as input to the optimisation, which is not a major concern in this thesis project. The tasks that have to be fulfilled during the preprocessing are however described in the first section of this chapter, since some knowledge of the data that is obtained from preprocessing is necessary.

Finding traversable routes between deployment sites constitutes the second part of optimisation, and is a vital part of this project. More will be described in the second section of the chapter. Lastly, the main task of finding optimised sequences of deployment sites to utilise is described in the third section.

A number of assumptions have been made when developing the method. It has been assumed that all units involved in a mission are given separate deployment areas, disjoint from all other involved units' deployment areas. It has further been assumed that all units are operating at their respective deployment sites during τ time units at a time, before moving to the next site.

These assumptions allow deployment planning for units to be done individually, since all units can be allowed a fixed time for moving between deployment sites. If the mission starts at time t, the idea is that the first unit starts sending at its first deployment site at that time, the second unit at time $t + \tau$, the third unit at time $t + 2\tau$ and so on. At time $t + n_{\text{units}}\tau$, the first unit will then start operating at its second site, having been allowed $(n_{\text{units}} - 1)\tau$ time units since ceasing operation at its first site at time $t + \tau$, when the second unit took over. In this way, there is always a unit that is operating from some site, and units' schedules can be planned individually. Therefore, the schedule for a single unit will be considered from here on.

4.1 Input Interface

This section describes the input data that is used for the project. How the input data should be obtained is not a topic of this project and is only covered briefly. It is however of importance to clarify what input is needed to complete subsequent parts of the project, and how the interface to obtain that data looks.

Mainly, the data needed can be divided into three groups: terrain data, road network

data and deployment site data. What is included in these groups will be further described in the respective subsequent parts of the section.

4.1.1 Terrain data

In order to determine the traversability of the terrain in the deployment area, there has to be data available about the terrain in all points of interest. For the scope of this project, this data consists of Vricon's DSM and classification data. The data is provided as a raster with a resolution of 0.5 m which offers good precision for the task at hand.

However, one would ideally want more details in this data than is currently available. For example, the terrain data provides no details with regard to the bearing capacity of the ground and some of the classes in the classification data could be further divided to provide more detail. Also, some measure of the exposure to enemies at all points in the DA could enhance the resulting application.

4.1.2 Road network data

In Vricon's terrain data, there is only the 'paved surface' class that indicates the existence of a road. This omits all roads that are not paved, meaning that the data from Vricon is not sufficient when it comes to representing the road network. Therefore, data from OSM (see Section 2.2) has been included as the primary source for road data. It is however expected (as mentioned in Section 2.1.2) that the quality of classification data will be better in the near future, in which case there might not be a need for using OSM data. How the method can be modified to account for this will be discussed further in Section 6.2.3.

4.1.3 Deployment site data

Suggested deployment sites are assumed to be given from a preceding analysis step. The data that is needed about deployment sites is primarily its spatial position and attributes relating to its ability to monitor the given TAI.

When given the spatial position, the available map data is able to supplement with additional data and enable the application to calculate ways of entry and exit to the site. Ideally, information could be supplemented by data about the site's visibility (exposure to enemies). This property should in that case typically have a higher value in places when the unit can easily be spotted, such as open plains or hillsides, while it should take a lower value in places where the unit is harder to spot.

As it comes to the ability to monitor a TAI from a specific site, there are a number of relevant properties. In Section 3.2, the proportion q_{cov} of the TAI that is covered was cared about. Since the airspace above the TAI should be monitored, free line-of-sight to the ground is not necessary. Free line-of-sight to an as low height above ground as possible is however desirable. Therefore, q_{cov} is defined as the proportion of an

imaginary surface above the TAI, on a fixed vertical distance $h_{\max}$ above the ground, to which there is a free line-of-sight. The height $h_{\max}$ should be set depending on the requirements on the mission.

4.2 Route Optimisation

To find optimal or near-optimal routes between pairs of deployment sites, an algorithm that processes the input data in several stages is utilised. As both raster data provided by Vricon and vector data from OpenStreetMap should be integrated, an algorithm that combines and takes advantage of both data sources is deployed. The foundation for this approach is the method described in an article by Benton, Iyengar, Deng *et al.*[24], where raster data is vectorised with the aid of a morphological skeleton. Using a modified version of this method, which will be called the *skeletonisation method*, with the Vricon data, the output is combined with data from OSM. Further details about these procedures will be described in the following sections.

In the context of this chapter, we will consider the route that takes the least time to travel between two points as the optimal route between said points. There remains a possibility to look at other optimality criteria as the whole project is concerned. For instance, a route that is less visible from an enemy's perspective, but takes longer time than the fastest route, might be a better alternative in some cases. The prospect of searching for such routes will however not be considered here.

In order to avoid confusion, the spatial curve to be taken between two nodes connected by an edge is called a path. Thus, when a transportation graph is defined, paths will be properties of edges in the graph. We will further denote a sequence of one or more edges and their corresponding paths as routes. In the context of this project, the routes of interest are those between pairs of deployment sites.

The proposed method has been chosen since it efficiently transforms a large raster with a lot of information into a graph that can be more easily processed. The relevant information from the raster is concentrated to enable an efficient method that can still take advantage of the high resolution in the original data. For a discussion about alternative approaches to solving the problem of route finding, see Section 6.4.1.

4.2.1 Vricon data processing

The primary aim with processing the DSM and classification data is to obtain data points that indicate how well a given vehicle can traverse the corresponding points in the terrain. With such a traversability map available, the skeletonisation method (see Section 4.2.2) is deployed to create a transportation graph out of the raster data.

The traversability maps have been restricted to be binary, since it simplifies subsequent steps in the overall application. Namely, the resulting map has the same

resolution as the DSM and classification data, and marks every point as either traversable or non-traversable.

To create such a map, the classes in the classification data has been determined as either traversable or non-traversable, with the traversable classes being ground, low vegetation and paved surface (see Table 2.1). Furthermore, the height data has been used to determine the slope in all points of the map so that all points with a slope exceeding a certain threshold are marked as non-traversable. The map is then generated so that all points belonging to a traversable class, with a slope below the threshold, are marked as traversable. All other points are marked as non-traversable, since they fail to fulfil at least one of the two traversability criteria.

As a vehicle cannot traverse too narrow passages, the traversability map is additionally processed to account for the required width of passages. This is done by eroding the image (see Section 2.4.1), with a circular structural element whose diameter is equal to the required width for passage by the relevant vehicle.

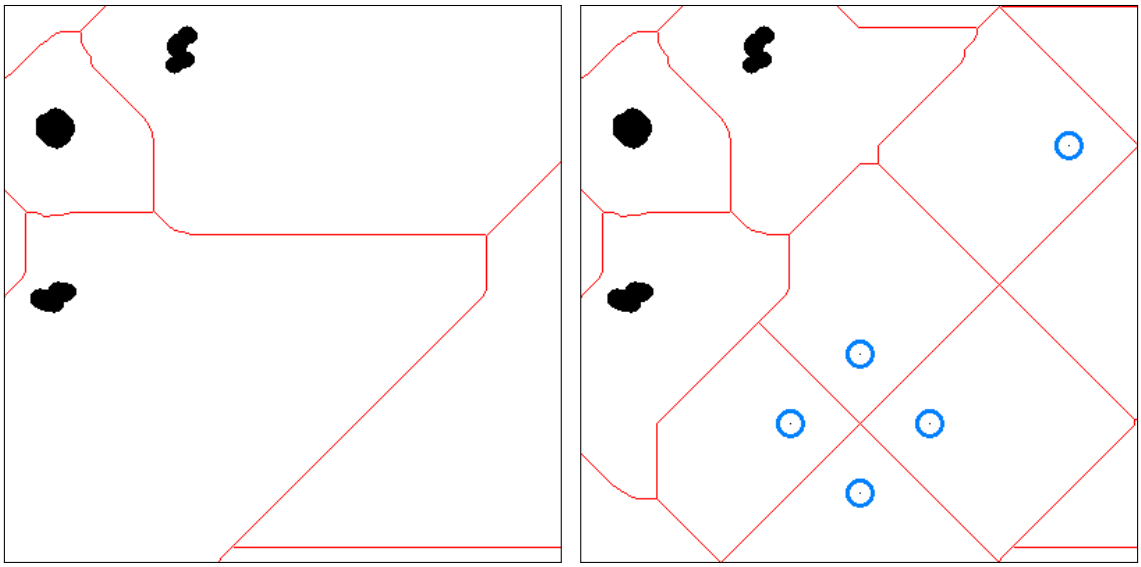
4.2.2 Graph creation from raster data

In order to create a transportation graph from the binary traversability map, the basics of the technique described by Benton, Iyengar, Deng *et al.*[24] is used. In the article, the authors create a morphological skeleton (see Section 2.4.3) out of the traversable areas of a map and then use this skeleton to define a graph. The basic idea to define a graph out of the skeleton is to create nodes at all ends and junctions of the skeleton, and edges between nodes that are connected by the skeleton. The steps of this process are described in the following four sections, starting with a modification of the traversability map that yields a more versatile skeleton.

4.2.2.1 Traversability map modification

To avoid a too extensive skeleton, small connected components of non-traversable pixels are removed from the traversability map before the skeletonisation procedure is applied. There is for example no point in extending the skeleton due to individual stones or other small obstacles. Connected components of non-traversable pixels that do not exceed a size threshold are therefore removed.

In order to generate a skeleton that is more versatile in open areas however, the traversability map is subsequently modified further to avoid having only a single segment of the skeleton go through large areas of open terrain. In large areas where all points are traversable, non-traversable points are added in order to extend the resulting skeleton and allow for larger freedom of movement in the resulting transportation graph. The modification is performed by dividing the traversability map into squares with fixed side lengths and checking if there are any non-traversable points inside that square. If there are no non-traversable points inside the square, four non-traversable points are added at the middle of the square's sides, making a diamond pattern. An example of the result of such a modification can be observed in Fig. 4.1 where it can be seen that the resulting skeleton is more extensive when this modification has been performed.



(a) Traversability map before modification.

(b) Traversability map after modification.

Figure 4.1: Examples of skeletonisation that show the effect of traversability map modification. To the left is the resulting skeleton for a part of a binary map that is skeletonised without prior modification for extending the skeleton to large traversable areas. To the right is the resulting skeleton for the same binary map, but with prior modifications. The red pixels are part of the skeleton that results from the white-coloured foreground area and the black-coloured background. In the picture on the right, some barely visible non-traversable points are circled with blud for clarity.

4.2.2.2 Skeletonisation

As described in Section 2.4.3, a morphological skeleton can be obtained using a number of techniques. For usage in this project, the iterative approach in an article by Bataineh[11] has been taken. The underlying idea is to repeatedly remove ('peel') the contour pixels of the skeleton until the skeleton consists solely of one-pixel wide strokes.

The skeletonisation algorithm contains of a main loop whose body includes a preprocessing step and a peeling step. This is executed repeatedly until the binary map does not change anymore between iterations. After the main loop, postprocessing is performed on the image before the final skeleton is obtained.

The preprocessing step of the main loop marks pixels on the boundary of the traversable areas (true areas of the binary map) as contour pixels; a procedure that includes adjustment for irregular ('noisy') edges. The second step of the main loop is then to peel off contour pixels that satisfy certain conditions. First of all, strokes that are one pixel wide are marked as finished so that they are kept as they are for all subsequent iterations. Then all other contour pixels are removed, given that they satisfy conditions ensuring that the topology of the skeleton is preserved.

After the main loop, a second loop that postprocesses the image is executed to take care of strokes that are two pixels wide and reduce them to one-pixel width. Again, this is repeated until no further changes are made to the skeleton.

After terminating the loop, so-called stairs in the image are removed in a single pass. Stairs in this context are diagonal strokes where every pixel has one other pixel connected horizontally and one pixel connected vertically. An overview of the full skeletonisation algorithm can be obtained in Algorithm 4.1 and an example of a completed skeletonisation procedure using this algorithm can be viewed in Fig. 4.1.

Algorithm 4.1 Skeletonisation algorithm

while Image changes **do**
    Preprocessing procedure
      – Mark pixels as contour
      – Remove noisy edges
    Peeling procedure
      – Mark one-pixel wide strokes as finished
      – Remove contour pixels given conditions on neighbours
end while
while Image changes **do**
    Two-pixel wide strokes removal
end while
One-pass stairs removal

4.2.2.3 Graph definition

When a skeleton has been obtained, a graph is created by following the pixels of the skeleton. The algorithm employed for this purpose scans all pixels of the skeletonised image and marks every observed pixel that is not part of the skeleton as processed. As soon as it finds a pixel that is part of the skeleton, it traverses the skeleton starting from that pixel. The traversal is performed by looking at the current pixel's 8-neighbourhood (the eight surrounding pixels) in every step, and then moving along the skeleton to a yet unvisited foreground pixel in the neighbourhood.

It starts by traversing the skeleton until it finds a pixel that represents a node. Such a pixel should have any number but two of the pixels in its 8-neighbourhood marked as true. If the number of skeleton neighbours is less than two, the pixel is the end of the skeleton and should thus define a node. On the other hand, if the number of skeleton neighbours is larger than or equal to three, the pixel represents a junction of the skeleton and should also define a node. Other skeleton pixels have exactly two skeleton neighbours and are part of a link between two pixels that define nodes.

When a first pixel representing a node is encountered, the algorithm creates a node in that position and restarts traversal of the skeleton from there. The algorithm maintains a list of 'outgoing' pixels that have exactly two true pixels in its 8-neighbourhood and is located adjacent to a pixel contained in a node. In every entry of this list, the node adjacent to the outgoing pixel is stored as well.

One-by-one the algorithm then processes the outgoing pixels by following the skeleton from that pixel until it finds another pixel that satisfies one of two conditions; it should either be part of the list of outgoing nodes, or it should have a number of true pixels in its 8-neighbourhood implying that a new node should be created at the location. In the case that a node in the list of outgoing pixels is encountered, a new edge is added between the associated node of the found outgoing pixel and the associated node of the outgoing pixel from which the traversal started. In the other case that a found pixel fulfils the criterion for defining a new node, such a node is created in that position and an edge is defined between the new node and the associated node of the outgoing pixel that the traversal started from. In this case, the algorithm additionally looks for new outgoing pixels from the new node and adds them to the list of outgoing pixels if they are present. In both cases, all traversed pixels are marked as processed and the outgoing pixels that have been covered are removed from the list.

Iterating until there are no more outgoing nodes left in the list, this algorithm will have created a graph of the connected component of the skeleton that was encountered. As all pixels that are traversed is marked as observed, the algorithm can check for more connected components of skeleton without processing the same connected component more than one time. It does so by traversing the remaining pixels of the image that are not yet marked as processed, until no such pixels exist anymore.

4.2.2.4 Local path finding

After a graph with edges and nodes has been defined, the minimal time it takes to transport the relevant vehicle between all pairs of nodes that are linked by an edge should be found. This is done by finding the shortest path in the original binary traversability map. The speed at which the terrain can be traversed is taken as constant over the whole traversable area of the map, and therefore it is equivalent to minimise the travelled distance and the travelled time.

To find the shortest path between two nodes, a check is first performed to see if a straight line between the nodes intersects any non-traversable area. The check is performed by checking every pixel that is marked by the Bresenham algorithm.[25] If all checked pixels are traversable, the shortest path is the straight line and it is stored, with the traversal time of the path (and edge) taken as the straight line distance divided by the terrain traversal speed.

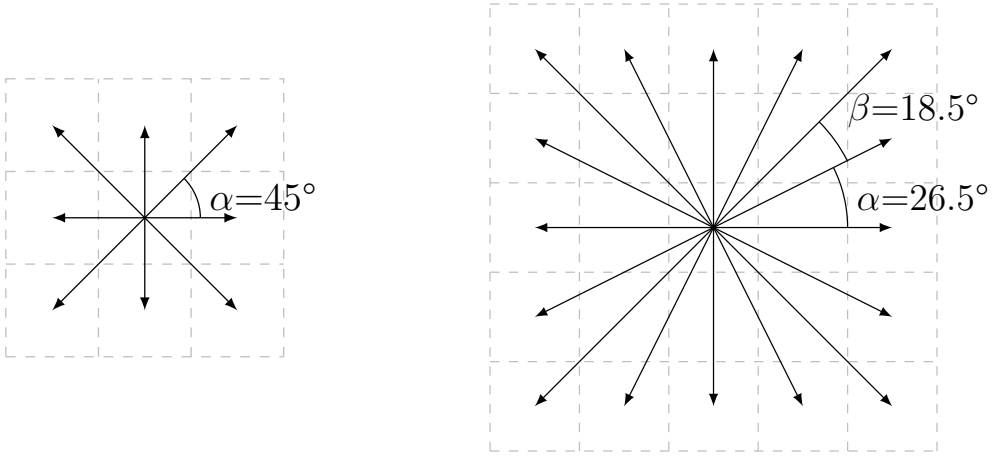
If instead any encountered pixel during the straight line checking is non-traversable, an implementation of the A* algorithm (see Section 2.5.2) is deployed for finding the shortest path between the two nodes. The straight-line distance to the target pixel divided by the maximum terrain traversal speed, plus the actual traversal time from the source node, is taken as heuristic for the implementation.

To increase the flexibility of movement compared to only allowing steps to the eight neighbouring pixels during execution of the algorithm, movement is allowed directly to any of the other 24 pixels in the 5-by-5 pixels square that is centred on the current pixel. This movement without checking if pixels in-between are traversable when moving to the outer pixels of the 5-by-5 square is justified by the fact that the binary traversability map was eroded in an earlier stage (see Section 4.2.1). This means that there cannot be any one-pixel wide non-traversable strokes in the traversability map, and thus the algorithm cannot jump over any non-traversable areas.

Allowing movement in only eight directions when traversing a grid as in this case can possibly yield a 7.97% longer path compared to the straight-line path, due to it being 45° between possible directions to move in. With allowing movement to 24 neighbouring pixels as described, there is at most 26.56° between possible directions, which yields a maximum error of 2.33%. This is illustrated in Fig. 4.2.

4.2.3 OpenStreetMap data processing

Since the OSM data is conveniently defined by nodes and edges (that are called ways in connection with OSM), the required processing for storing it in the same format as the resulting graph from the Vricon data is not very extensive. All OSM data (nodes and ways) that correspond to roads are extracted, and all other data is discarded. The OSM ways we are interested in have a tag with key ‘highway’ and they are converted to edges in the same format as is used for edges in the graph from the Vricon data. Ways without the highway tag, as well as nodes that are not connected by ways with the highway tag, are omitted. The relevant nodes are then also converted to the same format as the nodes obtained from the Vricon data.



(a) Possible 8-neighbourhood movements.

(b) Possible movements to the 16 pixels surrounding the 8-neighbourhood.

Figure 4.2: Illustration of angles between possible movements for the cases of only allowing movement to the 8-neighbourhood and allowing movement also to the 16 surrounding pixels.

The travel time of each edge, which can be traversed in a straight line between two nodes due to the nature of the OpenStreetMap data, is taken as the distance between the nodes, divided by the pre-set road speed. This road speed is typically larger than the terrain traversal speed (see Section 4.2.2.4), as is realistically the case when travelling in field.

To improve the subsequent merging of the resulting graphs from Vricon and OSM data that will be described in the following section, edges corresponding to long straight road sections are divided into several edges. It is performed by defining intermediate nodes on edges that corresponds to road sections longer than a given threshold, such that the distances between neighbouring resulting nodes does not exceed that same threshold. The original edge is then removed and new edges are added between neighbouring nodes.

4.2.4 Data merging

After processing the Vricon data and the OSM data separately, there are two disjoint sets of nodes, with corresponding edges, that all have a well-defined spatial position. In order to link the two disjoint graphs, an algorithm adds edges between every OSM node and its nearby neighbours from the set of Vricon nodes. For each OSM node, the whole set of Vricon nodes is observed, and an edge is defined to each of a predetermined number of Vricon nodes that are the closest and inside a given maximum range, with respect to the OSM node. Additionally, if no node is added within a quadrant as seen from the OSM node, an edge to the closest node in that quadrant is added if such a node exists within the given maximum range. This is done so that there is a spread in directions to move in from the node.

The paths and traversal times of the added edges that link nodes from the two data sources are then calculated in the same way as for edges from the Vricon data (see Section 4.2.2.4).

As the Vricon and OSM data have been merged to form a single graph, nodes representing the deployment sites are added to the graph as well. It is done in the same way as just described for each OSM node, with the difference that the deployment site nodes can be linked to all types of nodes. As this is completed, a full transportation graph is obtained, with all the needed information to search for optimal routes between pairs of deployment sites.

4.2.5 Route searching

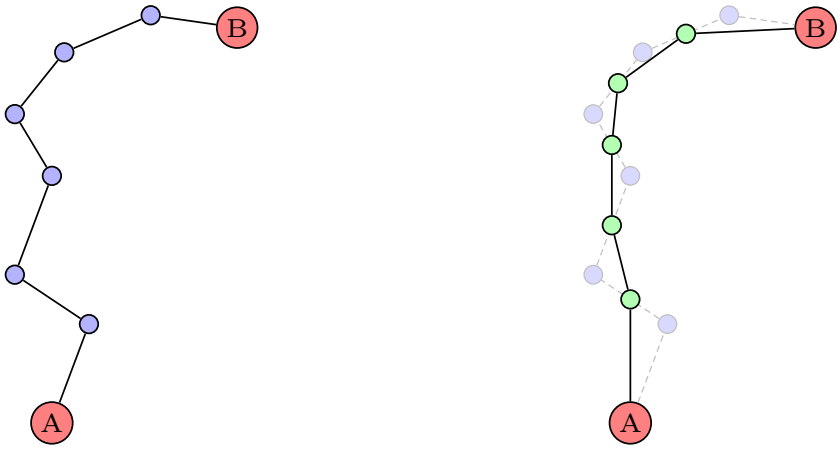
As the transportation graph is complete with components from the Vricon data, the OSM data and deployment sites, the traversal times of all edges in the graph are known. Since the nodes of the graph have a well-defined spatial location, it is easy to calculate a lower bound on the travel time between all pairs of nodes, given that the maximum speed a vehicle can travel with is known. Then an A* algorithm for finding the shortest (least time-consuming) routes between all pairs of deployment sites is implemented. The heuristic for each node is taken as the travelled time from the source node (deployment site), plus the straight-line distance to the target node (deployment site) divided by the maximum speed.

Under the restriction of movement in the transportation graph, this will yield the optimal route between all deployment sites. However, since the freedom of movement is not truly restricted to the exact paths of the transportation graph, the true optimal routes might differ from the found ones using the transportation graph. In order to address this non-optimality, an iterative postprocessing routine is deployed on the parts of routes that go through terrain.

4.2.6 Postprocessing routes

The main idea of the postprocessing is to add new nodes midway between the original nodes and then add new edges between the midway nodes. Then old nodes are omitted, except the source and target nodes, and new edges are added between the remaining neighbouring nodes. The first and last midway nodes are omitted completely so that the first and last new edges correspond to one and a half old edges each. An illustration of the process is presented in Fig. 4.3 where the path is clearly smoother after postprocessing.

This processing is done only for route segments in the terrain. If road nodes are present in the route, they will be considered source or target nodes in the postprocessing procedure if they are connected to a non-road node. In a typical case, a route consists of a terrain segment, followed by a road segment, followed by another terrain segment. In that case, the first terrain segment will be postprocessed with the first road node seen as the target node. Equivalently, the second terrain segment will be postprocessed with the last road node seen as source node.



(a) Route before postprocessing. **(b)** Route after postprocessing.

Figure 4.3: An illustration of the postprocessing of a route between a source node A and a target node B. The smoother turns can be observed in the postprocessed route, where the midpoints of all edges, except the first and the last, have been taken as new nodes.

If routes are really edgy, the postprocessing procedure can be iterated several times to obtain smoother routes. Postprocessing can however be rather costly, so there is a trade-off between further processing of the route and execution time, especially if paths between nodes far apart have to be searched for with the A* algorithm.

4.3 Schedule Optimisation

Once routes between deployment sites are found, optimising a deployment schedule is done by a genetic algorithm. In order to do this, the possible deployment sites are enumerated with indices from 1 to m, given there are m deployment sites in total. In the eyes of the genetic algorithm, the schedule is represented solely by a list of length N, containing the ordered set of deployment site indices. How a population is initialised, how selection works, how the genetic operators (crossover, mutation, elitism) is implemented and how the main loop combining these elements work is described in the following sections.

Other approaches for optimising schedules can possibly be taken, with ant colony optimisation methods[26], [27] and simulated annealing[28], [29] having been tested at the beginning of the project. They do not preserve good sequences of sites in a schedule as efficiently as a genetic algorithm and therefore tend to not yield as good results. Further discussion on possible alternatives is given in Section 6.4.2.

4.3.1 Initialising a population

In order to initialise a new population, schedules (chromosomes) are generated randomly and independently until the population has been filled to the desired pop-

ulation size. The first deployment site of every schedule is chosen randomly, with equal probability, among the m available ones. Then subsequent deployment sites are chosen randomly among all deployment sites that satisfies requirements with respect to the previous site; those requirements being that the site can be reached in the given maximum transportation time and that it is outside the required safety distance from the preceding site. This procedure is then repeated until N sites have been chosen, making up a valid schedule. Further on, more random schedules are created until the population is full.

4.3.2 Selection

Selection of individuals to use when creating the next generation is done via tournament selection. The procedure starts with choosing n_t of the schedules in the current population with equal probability (and replacement). Among the n_t chosen schedules, the one with best fitness (largest success probability) is returned with a probability p_t, typically with $p_\mathrm{t} > 0.5$.

If the schedule with best fitness is not returned, it is removed from the set of chosen schedules and the same process is repeated with the remaining set of schedules. That is, the second best schedule from the originally chosen set is returned with probability p_t. This is repeated until a schedule is returned or until the set of schedules consists of only one schedule, in which case that schedule is returned.

4.3.3 Genetic operators

When creating a new generation of a population, two or three genetic operators are used. These operators are crossover which is used on pairs of schedules, mutation that are used on individual schedules, and elitism that is used on the population as a whole. How they are implemented is described in the following three sections.

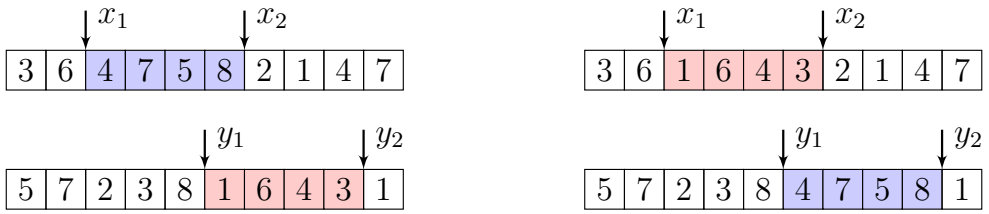
4.3.3.1 Crossover

A general crossover operator takes two schedules and exchanges parts of their sequences of deployment sites with one another. In this implementation, a two-point crossover that takes a randomly chosen segment of the first schedule and exchanges it with an equally long segment of the second schedule is utilised.

The schedules of length N can be 'cut' in-between deployment sites as well as at the end points, yielding $N + 1$ possible cutting points. The cutting points for the first schedule in the crossover is chosen randomly with equal probability for all cutting points, which yields a schedule segment with n entries between the cutting points, where n is between 0 and N. As this is done, the first cutting point in the second chromosome is chosen randomly from the first $N - n$ available cutting points. A restriction on these cutting points ensures that a connected segment of the second chromosome is chosen for the crossover, since the segment from the second schedule

must also be n entries long. Therefore, the second cutting point of the second schedule is implicitly given after the first cutting point is obtained.

The chosen segments are then exchanged with each other, creating new schedules. An illustration of this procedure can be seen in Fig. 4.4. An exception is if any of the new schedules would yield a deployment site following itself in the schedule. If that would happen, the crossover is omitted and the original chromosomes are returned instead.



(a) Before crossover. **(b)** After crossover.

Figure 4.4: Example of performing a crossover between two schedules of length 10. In the left picture are two chromosomes before crossover, where x_1 and x_2 have been chosen randomly as cutting points for the first chromosome. The first cutting point y_1 of the second chromosome has also been chosen randomly, while y_2 is given so that the segment between y_1 and y_2 has equal length to the segment between x_1 and x_2. The segments in-between cutting points are then swapped, and the result can be seen in the right picture.

4.3.3.2 Mutation

Mutation is executed on individual schedules by iterating through the schedule and independently replacing deployment sites with a probability p_m. If a site is mutated, it will be replaced by a new deployment site. That new site is chosen with equal probability among all sites that are not in the previous or following place in the schedule.

4.3.3.3 Elitism

The elitism operator is used in order to forcibly keep the schedule with the highest fitness in each generation into the following generation. It is performed by copying the best schedule from the previous generation into the first position of the new generation. This is performed after the new generation has been generated by the selection, crossover and mutation procedures, and implies that the first generated schedule of the new population is overwritten.

4.3.4 Main genetic algorithm loop

The above described procedures are executed iteratively in the main loop of the algorithm, where each iteration of the loop represents a generation of the popula-

tion. In order to create a new generation, two schedules at a time are chosen by tournament selection.

With a probability p_c, a crossover between the two selected schedules is performed. Whether or not crossover takes place, both schedules are mutated and inserted into the new population. This is repeated until the new population has been filled. Lastly, the elitism operator is used to include the current best schedule into the new population if it is activated.

The newly constructed population is then evaluated and subsequent generations are created until either a pre-set number of generations have been generated or until a satisfying result has been obtained. Before terminating, the best obtained schedule is returned. In Algorithm 4.2, an overview of the whole optimisation procedure can be observed.

Algorithm 4.2 Overview of the schedule optimisation genetic algorithm.

```
Randomly initialise a new population of schedules
while Termination condition is not fulfilled do
    Create empty new population
    while New population is not filled do
        Choose a schedule 'A' with tournament selection
        Choose a schedule 'B' with tournament selection
        if x ~ unif[0, 1] < p_c then
            Perform crossover between schedule A and B
        end if
        Mutate schedule A
        Mutate schedule B
        Insert schedule A and B into new population
    end while
    if Elitism is activated then
        Insert best schedule from current population into new population
    end if
    Replace current population with new population
end while
```

5
Implementation

The implementation described in this chapter is done to demonstrate the project's proposed methods for route finding and schedule optimisation. How the proposed method of Chapter 4 has been implemented to create a schedule optimisation application is described in the first section of the chapter.

A graphical user interface (GUI) that visualises the outcome of the optimisation procedure has been created and is presented in the second section. Such a GUI is important for a user to be able to comprehend the outcome of the schedule optimisation application. Furthermore, the third and last section of the chapter displays and explains some results of running the algorithm on a set of test cases.

5.1 Optimisation Method

This section describes how the proposed method of Chapter 4 has been implemented in Java code. The most important libraries and classes of the implementation are described in addition to the overall structure of the code project.

5.1.1 Terrain data parsing

The obtained geospatial data from Vricon (DSM and classification) is delivered in the GeoTIFF format.[30] In order to read the data and convert it to a Java format, the open source library GeoTools[31] has been used. Specifically, the `GeoTIFFReader` class is used to load data into the Java `WritableRaster` format.

The data is delivered in several files with at most 8192×8192 data points, corresponding to an area of about $16\,\mathrm{km}^2$ with a resolution of $0.5\,\mathrm{m}$. Since the application must handle several such tiles to cover larger areas, several tiles (resulting in several instances of `WritableRaster`) are imported separately.

To generate a complete traversability map according to Section 4.2.1, the first steps of determining if the slope and class both indicate a traversable point are performed separately for the imported tiles of map data. Then the resulting traversability maps are concatenated with each other and the final step of eroding the traversability map is performed on the concatenated map.

For large areas, keeping data in a single `WritableRaster` object is not possible, due to the format being backed internally by a one-dimensional array. Such arrays

can hold no more data entries than the maximum value of a Java integer, which is $2^{31} - 1$. With the given data resolution, this means it is only possible to store data for areas of up to about $500\,\mathrm{km}^2$. Therefore, OpenCV (see Section 2.7) has been used for managing raster data in the project. Consequently, the binary maps are converted to `Mat` objects, the basic data format of OpenCV, before concatenation. This structure is two-dimensional by nature and represents maps efficiently.

Following the procedure described in Section 4.2.1, a morphological skeleton is created from the binary traversability data. Instances of the custom `Node` and `Edge` classes are then created according to the procedure described in Section 4.2.2.3. As this is done, the shortest paths are added as properties of the edges according to Section 4.2.2.4.

5.1.2 Road data parsing

The OSM data utilised in the application is obtained from Geofabrik[32] that provides OSM data free of charge from all over the world. The data is downloaded in the compressed .pbf format, which can be parsed by the Osmosis[33] library. The primary class of interest in this library is `OsmosisReader` that creates Java objects of the Osmosis classes `Node` and `Way` from the loaded data.

Since these classes do not have all desired properties needed for the project, they are converted to the project's custom-made `Node` and `Edge` classes. An important aspect of this is converting the latitude-longitude coordinate representation of the Osmosis classes to the desired UTM coordinate representation. Furthermore, the processing steps of calculating traversal times and dividing long edges are performed according to Section 4.2.3.

5.1.3 Route generation and serialisation

Before finding routes, road and terrain data are merged according to Section 4.2.4. Namely, edges are added between nodes obtained from the OSM data and nodes obtained from the skeletonisation procedure. Additionally, deployment sites can then be added at this stage as well, also according to Section 4.2.4. As the full network with deployment sites and merged OSM and Vricon data is created, the A* algorithm is utilised to find routes between all pairs of deployment sites, according to Section 4.2.5.

At any time after creating the original graph from the Vricon and DSM data, the main object `GraphController`, which holds the full graph, traversability raster and all relevant properties, can be serialised (saved). It is done by making sure that the classes of all contained objects of the `GraphController` implement the Java interface `Serializable`. Since the `Mat` class from OpenCV does not implement this interface, it is wrapped in a container class that stores the data in an array of arrays before serialisation. The original data structure is then restored upon loading of the wrapper object.

An alternative to serialisation is to proceed immediately to the schedule optimisation. But since adding deployment sites take relatively short time in comparison to creating the full graph from raw input data, it is convenient to preprocess and serialise data before knowing the positions of deployment sites to use. The `GraphController` can then be loaded from memory when needed, deployment sites can be added, and routes can be found in relatively short time before continuing to the schedule optimisation.

5.1.4 Schedule optimisation

For storing deployment schedules, the class `Schedule` has been created. It contains data about all deployment sites and routes between them. A new `Schedule` instance is evaluated by the objective function (3.1) upon creation.

A main class `GAOptimizer` that holds the main genetic algorithm loop (see Section 4.3.4) has been created. An instance of this class is created with all parameters set, both concerning the desired schedule and the parameters of the genetic algorithm itself. A population of `Schedule` instances based on a given `GraphController` is created upon calling an initialisation method of `GAOptimizer`. The optimisation procedure is then controlled by the method `iterate` that iterates the population of schedules through a given number of generations. This method can be called any number of times, and in-between calls there are other methods available to obtain the currently best schedule, the overall best schedule found during all generations, and the success, survival and full coverage probabilities of said schedules.

To observe and control the optimisation procedure, an optimisation GUI has been created, in which the success, survival and full coverage probabilities of the best schedule in the population can be observed over time. Before initialising a population, the user can set a number of parameters in a panel on the right hand side of the GUI window. Then the user can choose to iterate the population through a given number of iterations, or to continuously iterate the population until the user presses a stop button. Additionally, the user can open the schedule visualisation GUI, that will be described in Section 5.2, by clicking a button. Furthermore, it is possible to open several tabs in the window, allowing several optimisation procedures to run simultaneously where each tab corresponds to one `GAOptimizer`. A screen capture of the GUI can be seen in Fig. 5.1.

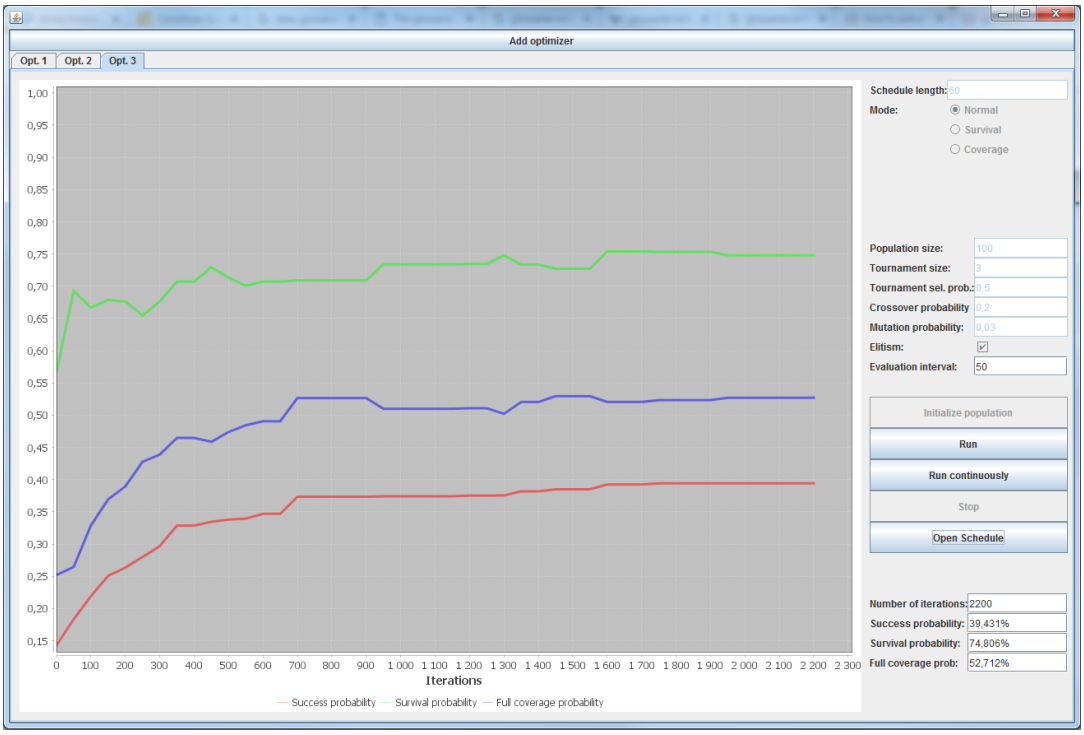


Figure 5.1: A screenshot of the optimisation GUI. In the main panel on the left hand side of the GUI window is a graph displaying the success, survival and full coverage probabilities in red, green and blue, respectively, as a function of the generation of the genetic algorithm. To the right is a control panel through which the user can set schedule and GA properties before initialisation of the population. With the buttons in the same panel, the optimisation procedure can then be controlled, and the user can choose to open the schedule visualisation GUI (see Section 5.2). By adding more tabs with the button at the top of the window, several optimisation procedures can be opened and run in different tabs of the window simultaneously.

5.2 Schedule Visualisation GUI

An important aspect when creating an optimising application, as described earlier in this chapter, is the ability to display information about the resulting schedule to a user. Therefore, a prototype GUI for visualising schedules has been created with the purpose to display an obtained deployment schedule to a user.

The main part of the GUI consists of a map displaying the utilised deployment sites and routes in-between them. When the GUI is opened, the route between the first and second deployment site in the schedule is shown on the map, and the user can then step between different routes with buttons in the GUI. The route that is currently selected is displayed in red and earlier routes in the schedule are shown in a less prominent green colour.

In addition to the map, there is an information panel on the right side of the GUI that displays information about the deployment sites that are connected by the currently selected route. The information is gathered in a table for each deployment site that shows the site's position, altitude and TAI coverage proportion among other things. There is additionally a table with data concerning the full schedule in the same information panel, such as the estimated success, survival and full coverage probabilities.

Below the information panel, there is another smaller map showing the DA in relation to the TAI. A screen capture of the full GUI can be seen in Fig. 5.2.

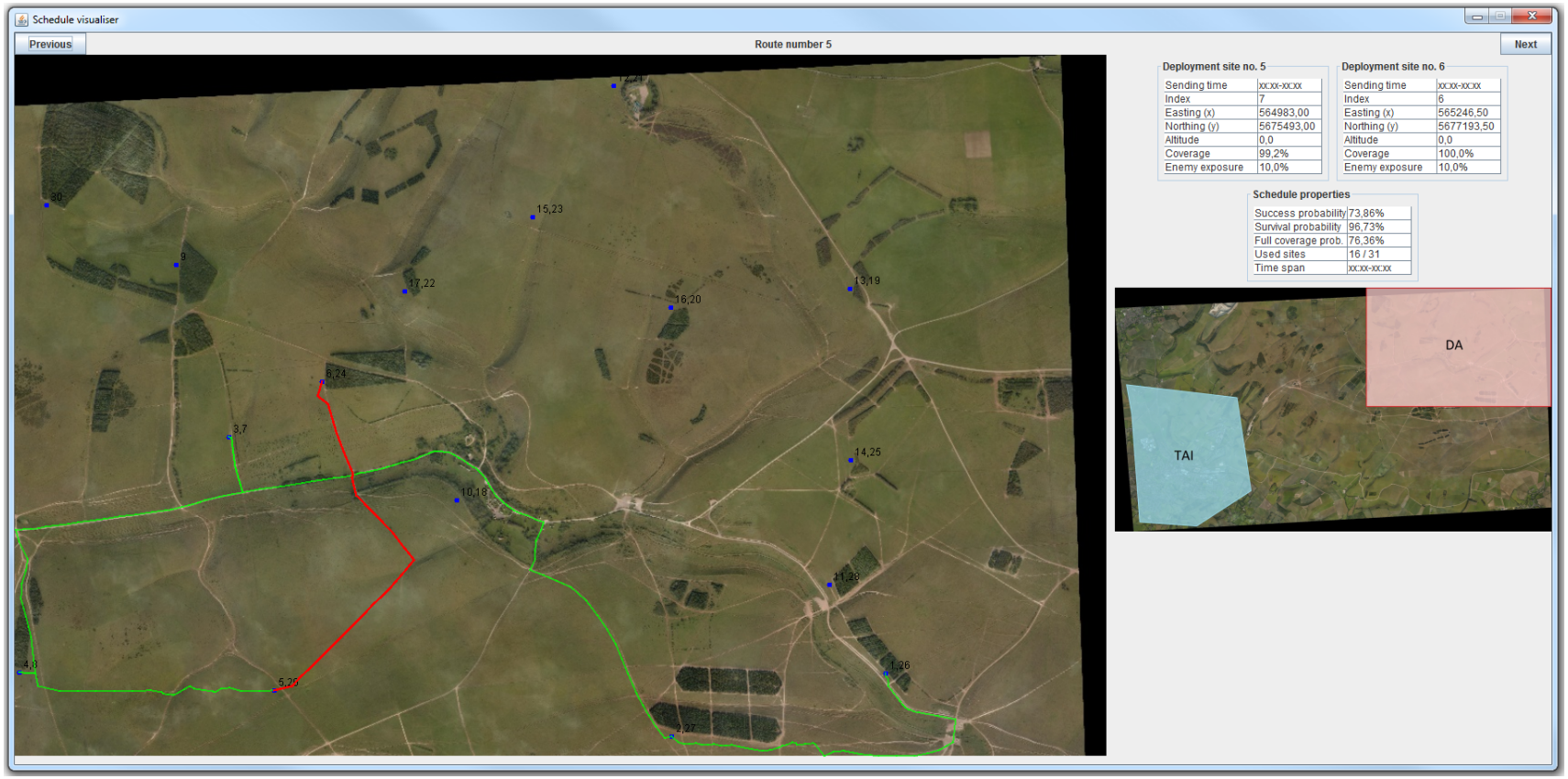


Figure 5.2: A screenshot of the schedule visualisation GUI. The large map to the left displays the DA with the currently selected route marked in red and previous routes marked in green. The user can choose route in the schedule using the 'Previous' and 'Next' buttons at the top. Information about the deployment sites connected by the current route, information about the schedule as a whole, and a smaller map displaying the DA in relation to the TAI can be seen to the right.

5.3 Results

The obtained results for optimisation can be observed to be heavily dependent on parameter settings. The results of running the full algorithm on a synthetic test case are considered. In this test case, the planning for one unit has been performed in a deployment area that is roughly 6 km × 3 km. In the area, 31 deployment sites, that all have a TAI coverage proportion q_{cov} between 0.7 and 1, have been placed.

To generate the transportation graph with this area takes roughly 6 minutes on the computer that has been used[1]. To load the transportation graph from file then takes about 35 seconds and to add the 31 deployment sites to the graph and search for the $\binom{31}{2} = 465$ routes between them takes about 80 seconds. Postprocessing the found routes takes about 2 minutes for one iteration. Creating a GA population with 100 individuals and evolving it for 10,000 generations take approximately 30 seconds, where the time for initialisation is negligible. These execution times should however be considered with care since there are a lot of factors that can influence them. They do, however, ´provide an idea of how running times relate to each other.

Results for this test case can be observed in Table 5.1, where 100 runs have been made for each of the three modes (normal mode, survival prioritisation mode and coverage prioritisation mode). It can be observed that normal mode yields best results for the success probability, survival prioritisation mode yields best results for survival probability and that coverage prioritisation mode yields best results for full coverage probability. These results are expected, and it can be further noted that the differences between modes for all four displayed values exceed the corresponding standard deviations.

[1] Quad-core 2.7 GHz CPU (Intel Core i7-6820HQ), 16 GB RAM, Windows 7 64-bit.

Table 5.1: Comparison of results for running the schedule optimisation genetic algorithm in different prioritisation modes for a fixed parameter setting. 100 runs were made for each mode, where each run consisted of initialising a population and evolving it for 100,000 generations. The number of successive sites in the schedule was 30, and there were 31 deployment sites with different properties to choose from in total. The first three columns represent the average success, survival and full coverage probabilities over the 100 runs, with the respective standard deviations in parentheses and the best average results for each of the three probabilities marked with green background. The last column indicates how many of the available 31 sites that was used for the schedule, with fewer used sites meaning there was a larger degree of re-use of the utilised sites. In the prioritisation modes, the exponent was set to $\alpha = 10$ for both cases. See Section 3.3 for a full description of the different modes and Sections 3.1 and 3.1 for descriptions of the probabilities.

Mode	p_{success}	p_{survival}	p_{coverage}	# used sites
Normal	0.804 (0.023)	0.913 (0.014)	0.880 (0.019)	13.48 (0.99)
Survival prioritisation	0.759 (0.031)	0.968 (0.001)	0.783 (0.032)	16.99 (0.84)
Coverage prioritisation	0.659 (0.047)	0.719 (0.048)	0.916 (0.016)	10.47 (0.85)

6

Discussion

This chapter aims to provide a discussion of different aspects of the project. A number of improvements for the proposed method are suggested, and some alternative approaches that could possibly replace parts of the proposed method are presented. The first section of the chapter synoptically discusses the obtained results, and preceding sections illuminate different parts of the project.

6.1 Obtained Results

The implemented method can be concluded to work well under the given circumstances. However, further analysis is required before implementing the method in a real planning application. An important aspect of such further analysis would be to make sure realistic input data is available. It is, for example, difficult to determine how realistic the synthetic deployment site data is, why deployment sites obtained from rigorous analysis should be utilised in further testing of the method.

By observing the results of testing different evaluation modes (see Table 5.1), it is clear that the two prioritisation modes can successfully steer the obtained schedules in the desired directions; that is, under the assumption that the modelled probabilities used in the objective function are reasonable. The most easily observed result that indicates that the prioritisation modes work are the re-usage of deployment sites; from the last column of the table, it can be seen that survival prioritisation mode results in more utilised deployment sites on average than normal mode, while the opposite holds for coverage prioritisation mode. Since it is easy to understand that re-using sites many times yield a higher risk of being located and attacked by the enemy, this is a clear indication of reasonability in the obtained results.

Most importantly when it comes to the results, however, is the performed analysis and the presented method itself. Since it cannot be stated as a fact that a certain obtained schedule is better than another, it is hard to estimate how good an obtained schedule really is. In any case, the presented method can be concluded to offer flexibility in that resulting schedules fulfil different properties depending on the input (such as choice of prioritisation mode).

6.2 Route Finding Improvement

In this section, a number of ways to improve the current route finding algorithm are presented. At first, some suggested extensions that provide robustness and additional flexibility are given. Then a way to account for differing traversal speeds in terrain and on roads is presented, before a description of how OSM data can be replaced by improved terrain classification data. Lastly, a couple of modifications that can possibly improve route finding are discussed.

6.2.1 Flexibility and robustness

A modification with potential to improve the overall quality of the full schedule optimisation application is to add the possibility to search for several routes between every pair of deployment sites. In that case, the subsequent schedule optimisation could choose between several routes to take between sites. This could potentially mean that the resulting schedules are more diversified and that traversals of busy road segments are better prioritised.

Several alternative routes can possibly be achieved by searching for one route at a time, where no edges can be shared among routes. Essentially, the traversed edges would be removed after each found route, and subsequent routes would be searched for among the remaining edges. In that way, the routes would be completely independent which is positive from a robustness perspective, so that other routes would still be present if one route is blocked.

An alternative could be to allow re-use of edges corresponding to terrain segments, but not allowing re-use of edges corresponding to road segments. This can be justified by the fact that road segments are typically where units are most exposed, and that they therefore should not be traversed too frequently. An example of how this can be done in either approach (allowing re-use of terrain segments or allowing no re-use of edges) is to search for more routes as long as the traversal time of said routes do not exceed the upper bound that is given by the mission properties.

External factors that might influence the possibility to travel a given route are (possibly dynamic) obstacles that are not known at the initial planning stage, with a typical example being other vehicles travelling on roads. On some narrow roads it might not be possible to meet with other vehicles and in extreme cases roads might be congested. Other external factors could be events such as fallen trees obstructing the way, or destroyed bridges or other passages. A strength of the taken approach, in the event of discovered obstacles, is that new routes can be recalculated quickly. A user can simply be enabled to remove edges of the graph that corresponds to destroyed passages or passages with obstacles, before routes are recalculated.

6.2.2 Speed variations

One major simplification during route finding is that the maximum speed for a vehicle is set to be constant. In reality, this is not the case since properties of the

terrain might influence the speed greatly. The speed of traversal might be much lower on a rough but traversable surface, compared to a flat open field such as an empty parking lot or a sport field. Due to the high resolution of the DSM data, there exists a possibility to estimate the maximum speed of traversal by observing height variations over short distances. Such a simple thing as the average slope over small areas might give an idea of how fast a unit can travel in that area.

A strength with the route finding approach is that it can be deployed even in a case where the possible traversal speed differs. In that case, a binary traversability map can still be used to create a transportation graph (see Section 4.2.2), before varying traversal speeds are taken into account in the local path search. A difference to the current approach (see Section 4.2.2.4) would be that travelling between two points in a straight line is not necessarily optimal. Therefore, all local paths would have to be determined by the A* algorithm, even if the straight-line path is possible.

Varying speeds on roads might also represent reality in a better way than is currently the case. Possible traversal speeds are in reality higher on large highways than on smaller dirt roads, which is why differing maximum traversal speeds should be allowed. A simple improvement might be to diversify the road network from OSM by utilising the values of the 'highway' tags (see Section 2.2). Additionally, height differences between connected nodes can be obtained from the DSM data and used to estimate the slope of road segments, which can influence possible traversal speeds.

6.2.3 Identifying road network in classification data

Ideally, one would not want to rely on OSM at all, since one can never be sure whether all roads in an area have been added to the database. If OSM is not to be used, road network data would have to obtained from Vricon's data, if other data sources are not to be used. Even though the quality of Vricon's data in this aspect is not sufficient at the moment, it is plausible to think that better quality classification data will be available in the near future. If that is the case, so that one or several reliable road classes are available in the data, a binary map of areas that consists of road segments could be created. Such a binary map could then be skeletonised, and the same graph defining procedure as used for the terrain (see Section 4.2.2.3) could be used to create a road network graph. That road network graph could then be merged with the terrain traversal graph, just as the OSM road network graph is merged with the terrain traversal graph.

6.2.4 Other improvements

One thing that is not accounted for is the time delays caused by vehicle manoeuvring. Turns and varying terrain means vehicles typically cannot maintain a constant speed, but must brake and accelerate regularly. While it remains a possibility to deploy more sophisticated algorithms to account for this problem, it should most often not be a problem that it is not explicitly handled. One should rather use an estimated maximum average speed for a certain type of terrain or road to account for

manoeuvring delays. Since movement is generally not time-critical except that units must reach its destination within a given maximum time, a slight underestimation of traversal speed instead of taking explicit care of turns and terrain shifts should not influence the performance of the application too much. To ensure a robust system, such an underestimation has to be done anyway since we cannot account for all external factors that influence traversal of routes.

The skeletonisation algorithm used in the project seems to produce rather strange results at occasions. It looks as if straight lines are often preferred over curves in the output skeletons, which means that the topology of binary maps are not preserved as well as desired. Using a different skeletonisation algorithm in a future implementation could therefore possibly improve the resulting routes.

In order to improve execution speed, there are two identified possible alterations to the A* algorithm that can be used. First, there is the possibility to implement a bidirectional A* algorithm.[34] This means to start two A* searches at once, one forward search from the search node towards the target node, and one backward search in the opposite direction. Then the algorithm can be terminated when a node is visited by both the forward and the backward search, since the concatenation of the route to that node from the source node and from the target node is then the shortest route between the source and the target node.

6.3 Schedule Optimisation Improvement

Improvements and extensions to the schedule optimisation genetic algorithm are the topics of this section, which consists of two parts. The first part concerns improvements and alterations that could possibly enhance performance of the genetic algorithm. The second part presents how the genetic algorithm can be modified to account for the case of several units being deployed in the same deployment area.

6.3.1 Genetic algorithm variation

There are a number of possible alterations to try and improve the schedule optimisation genetic algorithm. One possibility is to examine the effects of parameters for the GA in further detail. Given some test cases, different parameter settings could be iterated through, in order to try and find good choices.[35] Chances are however that the optimal choice of parameters varies depending on other inputs to the problem, such as desired schedule lengths and objective function parameters. To avoid the risk of determining falsely optimal parameters for all problems, one would have to test different settings for a large number of test cases with varying properties.

A common problem with genetic algorithms is that they tend to get stuck in local minima. There are a number of possible improvements to try and prevent that. One common method is to change the mutation probability depending on how the population evolves. Such an implementation would typically mean to increase the

mutation probability when the algorithm has not found a new best schedule in a given number of iterations.[36] The mutation probability can then be further increased if no better solutions are found, or restored to the original value if better solutions are indeed found.

Another possible solution to avoid getting stuck in local minima is to evolve several populations independently at first. Then one could mix these populations to generate new populations after a number of iterations. Blending populations like that would mean less homogeneous populations after mixing, which could benefit performance of the GA.

Utilising a local search algorithm at the end of the optimisation procedure is another addition that has potential to improve performance. Such an algorithm could, for example, take the best found schedule and try replacing all sites in that schedule individually. If a better schedule is then encountered, the process can be repeated with that schedule until it is not further improved by the local search algorithm.

6.3.2 Extension to several units per DA

A key assumption that has been made is that only one unit is used in each deployment area. Namely, disjoint deployment areas for each unit should be specified as input to the mission planning. A benefit of this separation of deployment areas is that changing conditions in one of the deployment areas only implies consequences for one of the units. The simplification it implies, in that planning can be done separately for each unit, is also a beneficial factor.

However, there could be advantages with allowing several units per DA. It could enable larger mobility and behaviour that is less predictable for the enemy. Another factor could be that different types of vehicles can be used in the same area. For example, a track-based vehicle can be used at deployment sites that are unreachable for a wheel-based vehicle, while wheel-based vehicles can be utilised at deployment sites that are easily accessible but far apart.

Allowing such a setup with several units in the same DA should be possible within the current schedule optimisation architecture. Routes can still be searched for just as before, possibly with different properties that correspond to different types of vehicles. For example, edges can be assigned one traversal time for vehicles of type ‘A’ and one traversal time for vehicles of type ‘B’.

The genetic algorithm can then be modified so that chromosomes correspond to all units’ schedules at once, so that every n:th entry in the schedule corresponds to one unit in a mission with n units, see Fig. 6.1. Checks would have to be performed when applying genetic operators, so that several units are not present at the same deployment site at once. This could be performed by simply checking that there are at least n entries in the chromosome before successive occurrences of the same deployment site.

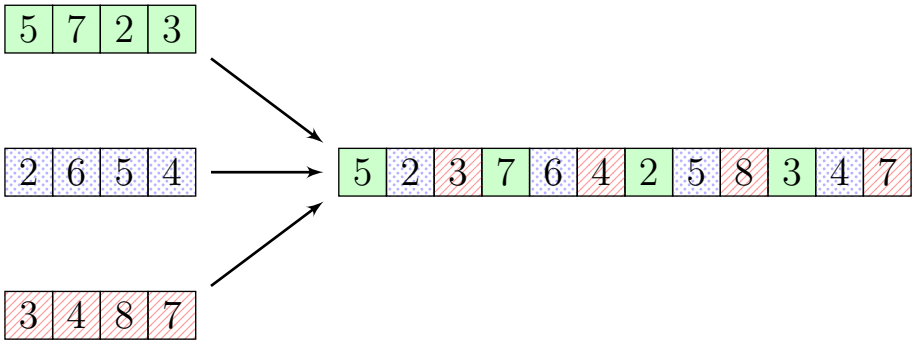


Figure 6.1: Illustration of how to combine the schedules of three units into a single chromosome. Different backgrounds correspond to different units.

6.4 Alternative Approaches

This section discusses several alternative approaches that can replace parts of the proposed method to achieve good deployment schedules. The first section concerns alternative ways to obtain routes between deployment sites, while the second section concerns alternative methods that could optimise sequences of deployment sites during the planning of a mission.

6.4.1 Route searching alternatives

A naive approach to the problem of route finding is to generate a traversability map and use the A* algorithm to search for routes directly between all pairs of deployment sites. That would never yield routes that take longer time to traverse than routes in the current application, but it would be computationally heavier. The computation time would also increase more rapidly with growing distances between deployment sites. Even though the A* algorithm is used in the current implementation, the routes it is searching for are relatively short. As the execution time of the A* algorithm typically scales with the square of the distance between points, keeping these distances short is critical to maintain reasonable execution times.

Additionally, using the A* algorithm to find routes directly between deployment sites would mean that a smaller part of the procedure can be done before deployment sites are known. In the current implementation, the computationally heaviest parts of the route finding can be performed before deployment sites are known. It would also mean that the application would not be as robust, since recalculation of routes when obstacles are discovered would not be as efficient.

6.4.2 Schedule optimisation alternatives

A number of alternative approaches to using a GA were considered for schedule optimisation in the beginning of the project. The two other approaches that were

tested using Matlab was ant colony optimisation (ACO) and simulated annealing.

ACO methods are commonly used for solving optimisation problems with spatial dependencies,[37]–[39] with probably the most famous being the Travelling Salesman Problem.[40], [41] Therefore, such an implementation was initially thought to be a good idea for optimising deployment schedules. However, testing using Matlab indicated that genetic algorithms performed considerably better. In addition, simulated annealing was tested with Matlab as a possible schedule optimisation method. The performance levels were however similar to the ACO testing so simulated annealing was discarded.

An intrinsic feature of the genetic algorithm is that it generally maintains well-performing sequences of its chromosomes, which is important in planning a deployment schedule. In basic implementations of ACO and simulated annealing, this is not the case, which is thought to be the primary reason that the methods are outperformed by the GA. In order to create an ACO or simulated annealing implementation that outperforms the GA, some modification has to be done to make sure that good sequences are exploited.

6.5 Schedule Quality Evaluation

The topic of this section is to discuss how the quality of obtained deployment schedules is evaluated. Firstly, the choice of objective function to use in optimisation is looked into. Then some additional aspects that the current evaluation model might miss are discussed in the second part of the section.

6.5.1 Choice of objective function

Modelling a scalar objective function for a problem with as many desired properties as in this case is intricate. Other choices for a scalar objective function than the probability of a successful mission are certainly possible, but could come with the downside that values of the objective function might not be easily interpretable, which is a strength of the taken approach. Even for the case of prioritising survival or full coverage in the optimisation process, the individual probabilities for survival and full coverage can be displayed and aid in understanding the optimisation procedure. The major downside is that it is difficult to define a realistic model for the probability.

One alternative approach is to define a number of properties to optimise and use some technique for multi-objective optimisation[42] to simultaneously optimise those properties. However, that would still yield the problem of choosing what properties to include. In the current implementation, while it might not be clear in detail how interaction between different properties impacts the model, individual properties have a clear definition (see Sections 3.1 and 3.2) and can be adjusted easily depending on what is desired for a mission.

6.5.2 Additional aspects

A number of simplifications have been done during the project, which might give non-accurate or non-optimal results. One such simplification is that it has been assumed that the only relevant property of a deployment site with respect to the TAI is the covered proportion q_{cov}. In a real application, it cannot be stated that this proportion is the only property that makes a deployment site good from a TAI coverage perspective. Free line-of-sight to lower heights above ground in a larger portion of the volume above the TAI is typically beneficial, among other things.

Another aspect that has not been covered is that vehicles can leave marks in the terrain. Even though a vehicle is generally more difficult to spot in the terrain compared to on a road, marks in the terrain might mean that it is dangerous to return to places where marks have been left.

6.6 Limiting Assumptions

An important thing to consider when planning movement for radar units, which has not been covered during the project, is the ability to communicate with own troops and allies. As a unit is operating, it should have an established data link to communicate information about detected missiles in the airspace. Essentially, this means that the surroundings of a unit must allow radio communication with a base station for the operation. In special cases, wired connection might be possible but must be considered an exception. Voice communication with the operator is a possible backup option that is typically not preferred. No matter what mode of communication is used, it should be accounted for when choosing deployment sites. However, since communication is a key factor for deployment sites, a simple solution is to not present deployment sites where communication is insufficient to the schedule optimisation algorithm.

Other requirements than to simply move between successive deployment sites might exist as well. For example, the unit and its vehicle might have to refuel during long missions. That can be done in one of two ways; either a tank truck can meet up with the vehicle along some of its routes when it is necessary, or the vehicle have to move to some given fuel station at given intervals. The former option would typically mean less time consumption for the unit, but requires coordination between the radar unit and the tank truck. What is clear is that either case should ideally be accounted for in the planning stage of a mission. If it has to be integrated in the current schedule planning or if it can be done in a separate stage afterwards is a topic of further analysis.

6.7 User Communication

One important aspect of a completed mission planning is that it should be comprehensible for a user. The schedule visualisation GUI created to display deployment

sites and routes on the map (see Section 5.2) is a step in this direction, but could be further improved. For instance, features such as the ability to zoom in the map are not present in the GUI, and some data is synthetic at present. Part of the problem is that no proper deployment site interface has been available, so that some data is synthetic, if it is present at all.

Additionally, an analysis should be done on what information is really relevant for a user. Technical details should typically not be displayed for a user that cannot understand the meaning behind such details. In general, presented information should be prioritised so that the user can focus on what really matters. However, since the method should be integrated into large planning applications for possible future usage, the visualisation parts should probably be developed as a part of such incorporating applications.

7

Conclusion

A method, divided into the two main parts of route finding and schedule optimisation, for acquiring optimised deployment schedules have been proposed. The method has furthermore been implemented in a proof-of-concept Java application that yields reasonable and desirable results under the given circumstances. For further testing and development of the proposed method, a number of suggestions are given, which includes improvements of both the route finding part and the schedule optimising part of the method. Also, what additional data that is required as input for testing the method in a more realistic setting has been explained.

The used skeletonisation method (see Section 4.2.2) for creating a transportation graph from raster data enables routes between deployment sites to be searched for quickly, as soon as the more costly initial processing is completed. A number of alterations are suggested for improving found routes in a future implementation. Most notably, these suggestions include allowing varying the possible speed depending on road and terrain types, and enabling searching for several alternative (and independent) routes to take between each pair of deployment sites. Improvements that can possibly speed up execution of the method are additionally presented, such as implementing bidirectional A* search. See Section 6.2 for a more comprehensive explanation of such improvements.

For schedule optimisation, the genetic algorithm has been concluded to work well. A number of possible improvements are however suggested for the schedule optimisation, with focus lying on methods to prevent the algorithm from getting stuck in local minima (see Section 6.3.1). Furthermore, suggestions on how to extend the genetic algorithm in order to allow several radar units to operate in the same deployment area are given in Section 6.3.2.

How deployment schedules can be evaluated is another topic that has been covered during the project. This, however, needs further investigation to ensure that the utilised models resemble reality. Nevertheless, the implemented approach with optimising a modelled success probability does seem to yield reasonable results, and the scalar-valued probability is easily interpretable. Furthermore, the implemented prioritisation modes makes tweaking of resulting schedules according to desired properties of the mission simple, and the simulation results in Section 5.3 suggest that the proposed method is efficiently optimising various objective functions. Therefore, alterations of the objective function in a future implementation should be possible without having to modify the optimisation method, except for the evaluation of schedules.

Bibliography

[1] Saab. (2018). Arthur – Weapon Locating System, [Online]. Available: https://saab.com/land/istar/weapon-locating-system/arthur/ (visited on 12th Nov. 2018).

[2] N. G.-I. Agency. (2018). Digital Terrain Elevation Data, [Online]. Available: https://www.nga.mil/ProductsServices/TopographicalTerrestrial/Pages/DigitalTerrainElevationData.aspx (visited on 8th Nov. 2018).

[3] Vricon. (2018). Company, [Online]. Available: https://www.vricon.com/company/overview/ (visited on 12th Nov. 2018).

[4] ——, (2018). Vricon DSM, Global elevation model with 0.5m resolution, [Online]. Available: https://www.vricon.com/wp-content/uploads/2017/06/Vricon_DSM_print.pdf (visited on 12th Nov. 2018).

[5] F. Aas Isaksen and K. Nilsen Brusevold, 'Mission Planning with High-Resolution Geographical Data', Bachelor's thesis, Østfold University College, 15th May 2018.

[6] J. Iliffe and R. Lott, '2.2 Coordinate Systems', in *Datums and Map Projections for Remote Sensing, GIS and Surveying (2nd Edition)*. Whittles Publishing, 2008, ISBN: 978-1-904445-47-0.

[7] B. L. Decker, 'World geodetic system 1984', Defense Mapping Agency Aerospace Center St Louis Afs Mo, Tech. Rep., 1986.

[8] J. W. Hager, J. F. Behensky and B. W. Drew, 'The Universal Grids: Universal Transverse Mercator (UTM) and Universal Polar Stereographic (UPS). Edition 1', Defense Mapping Agency Hydrographic/Topographic Center Washington DC, Tech. Rep., 1989.

[9] J. W. Hager, L. L. Fry, S. S. Jacks and D. R. Hill, 'Datums, ellipsoids, grids, and grid reference systems', Defense Mapping Agency Hydrographic/Topographic Center Washington DC, Tech. Rep., 1992.

[10] R. M. Haralick, S. R. Sternberg and X. Zhuang, 'Image analysis using mathematical morphology', *IEEE transactions on pattern analysis and machine intelligence*, no. 4, pp. 532–550, 1987.

[11] B. Bataineh, 'An Iterative Thinning Algorithm for Binary Images Based on Sequential and Parallel Approaches', *Pattern Recognition and Image Analysis*, vol. 28, no. 1, pp. 34–43, 2018.

[12] C. R. Maurer, R. Qi and V. Raghavan, 'A linear time algorithm for computing exact Euclidean distance transforms of binary images in arbitrary dimensions', *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 25, no. 2, pp. 265–270, 2003.

[13] P. Maragos and R. Schafer, 'Morphological skeleton representation and coding of binary images', *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 34, no. 5, pp. 1228–1244, 1986.

[14] T. Y. Zhang and C. Y. Suen, 'A fast parallel algorithm for thinning digital patterns', *Communications of the ACM*, vol. 27, no. 3, pp. 236–239, 1984.

[15] E. W. Dijkstra, 'A note on two problems in connexion with graphs', *Numerische mathematik*, vol. 1, no. 1, pp. 269–271, 1959.

[16] P. E. Hart, N. J. Nilsson and B. Raphael, 'A formal basis for the heuristic determination of minimum cost paths', *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[17] J. H. Holland, 'Genetic algorithms', *Scientific american*, vol. 267, no. 1, pp. 66–73, 1992.

[18] D. Whitley, 'A genetic algorithm tutorial', *Statistics and computing*, vol. 4, no. 2, pp. 65–85, 1994.

[19] C. Darwin, 'On the origin of species by means of natural selection, or the preservation of favoured races in the struggle for life', 1859.

[20] I. The MathWorks. (2018). MATLAB, [Online]. Available: `https://se.mathworks.com/products/matlab.html` (visited on 23rd Oct. 2018).

[21] O. Corporation. (2018). What is Java technology and why do I need it?, [Online]. Available: `https://java.com/en/download/faq/whatis_java.xml` (visited on 23rd Oct. 2018).

[22] OpenCV. (2018). About, [Online]. Available: `https://opencv.org/about.html` (visited on 11th Oct. 2018).

[23] J. Nicholson, *The Concise Oxford Dictionary of Mathematics*. 2014, ISBN: 9780199679591.

[24] J. R. Benton, S. S. Iyengar, W. Deng, N. Brener and V. Subrahmanian, 'Tactical route planning: new algorithms for decomposing the map', *International Journal on Artificial Intelligence Tools*, vol. 5, no. 01n02, pp. 199–218, 1996.

[25] J. E. Bresenham, 'Algorithm for computer control of a digital plotter', *IBM Systems journal*, vol. 4, no. 1, pp. 25–30, 1965.

[26] M. Dorigo and G. Di Caro, 'Ant colony optimization: a new meta-heuristic', in *Proceedings of the 1999 congress on evolutionary computation-CEC99 (Cat. No. 99TH8406)*, IEEE, vol. 2, 1999, pp. 1470–1477.

[27] M. Dorigo and M. Birattari, 'Ant colony optimization', in *Encyclopedia of machine learning*, Springer, 2011, pp. 36–39.

[28] K. A. Dowsland and J. M. Thompson, 'Simulated annealing', in *Handbook of natural computing*, Springer, 2012, pp. 1623–1655.

[29] S. Kirkpatrick, C. D. Gelatt and M. P. Vecchi, 'Optimization by simulated annealing', *science*, vol. 220, no. 4598, pp. 671–680, 1983.

[30] N. Ritter and M. Ruth, *GeoTIFF Format Specification, GeoTIFF Revision 1.0*, 31st Oct. 1995. [Online]. Available: `https://cdn.earthdata.nasa.gov/conduit/upload/6852/geotiff-1.8.1-1995-10-31.pdf` (visited on 18th Jun. 2018).

[31] GeoTools. (2018). About GeoTools, [Online]. Available: `http://geotools.org/about.html` (visited on 11th Oct. 2018).

[32] Geofabrik. (2018). Who we are, [Online]. Available: `http://download.geofabrik.de/` (visited on 11th Oct. 2018).

[33] OpenStreetMap. (2018). Osmosis, [Online]. Available: `https://wiki.openstreetmap.org/wiki/Osmosis` (visited on 11th Oct. 2018).

[34] T. Ikeda, M.-Y. Hsu, H. Imai, S. Nishimura, H. Shimoura, T. Hashimoto, K. Tenmoku and K. Mitoh, 'A fast algorithm for finding better routes by AI search techniques', in *Vehicle Navigation and Information Systems Conference, 1994. Proceedings., 1994*, IEEE, 1994, pp. 291–296.

[35] J. J. Grefenstette, 'Optimization of control parameters for genetic algorithms', *IEEE Transactions on systems, man, and cybernetics*, vol. 16, no. 1, pp. 122–128, 1986.

[36] M. Srinivas and L. M. Patnaik, 'Adaptive probabilities of crossover and mutation in genetic algorithms', *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 4, pp. 656–667, 1994.

[37] A. Lazarowska, 'Ship's trajectory planning for collision avoidance at sea based on ant colony optimisation', *The Journal of Navigation*, vol. 68, no. 2, pp. 291–307, 2015.

[38] S. Pothiya, I. Ngamroo and W. Kongprawechnon, 'Ant colony optimisation for economic dispatch problem with non-smooth cost functions', *International Journal of Electrical Power & Energy Systems*, vol. 32, no. 5, pp. 478–487, 2010.

[39] A. Rossi and G. Dini, 'Flexible job-shop scheduling with routing flexibility and separable setup times using ant colony optimisation method', *Robotics and Computer-Integrated Manufacturing*, vol. 23, no. 5, pp. 503–516, 2007.

[40] A. Colorni, M. Dorigo, V. Maniezzo *et al.*, 'Distributed optimization by ant colonies', in *Proceedings of the first European conference on artificial life*, Paris, France, vol. 142, 1991, pp. 134–142.

[41] M. Dorigo, V. Maniezzo and A. Colorni, 'Ant system: optimization by a colony of cooperating agents', *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, vol. 26, no. 1, pp. 29–41, 1996.

[42] G. Chiandussi, M. Codegone, S. Ferrero and F. E. Varesio, 'Comparison of multi-objective optimization methodologies for engineering applications', *Computers & Mathematics with Applications*, vol. 63, no. 5, pp. 912–942, 2012.