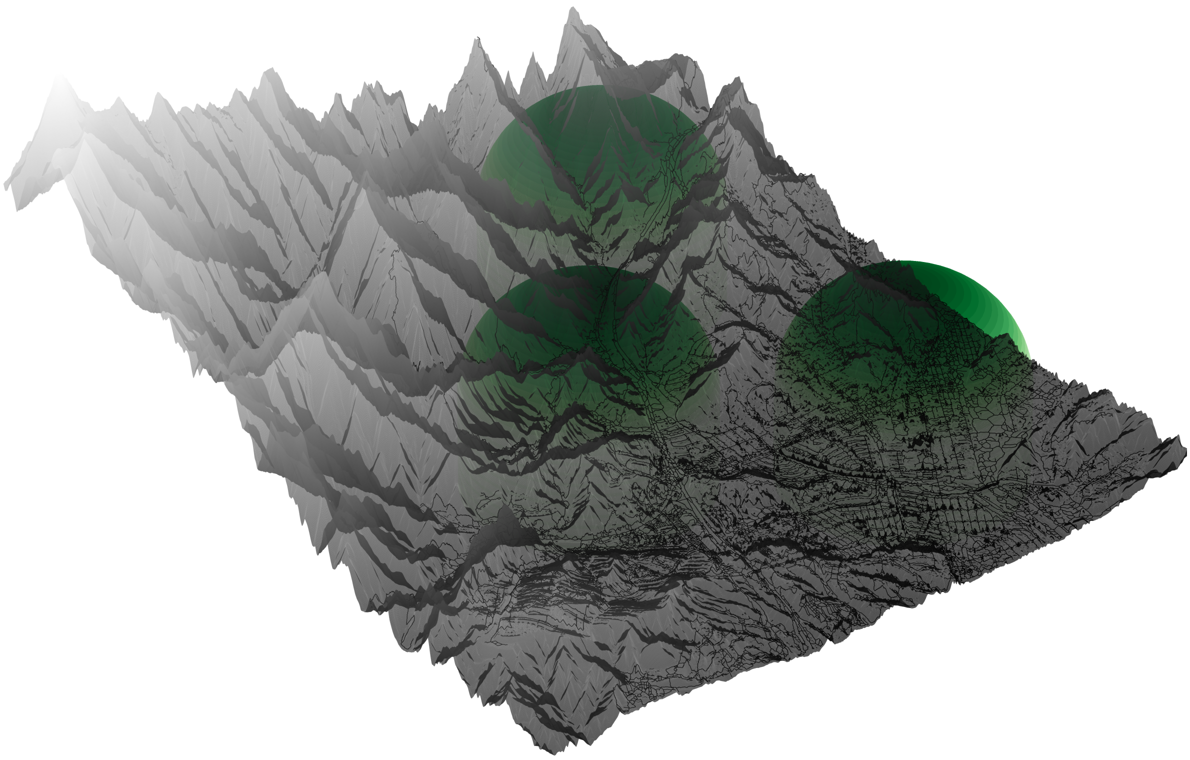
CHALMERS
UNIVERSITY OF TECHNOLOGY



Search-on-the-Move Radar: Multi-Route Planning using Machine Learning

Master's thesis in Complex Adaptive Systems

Jonatan Bertolozzi
Christian Jenei

DEPARTMENT OF SPACE, EARTH & ENVIRONMENT
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

Search-on-the-Move Radar: Multi-Route Planning using Machine Learning

Jonatan Bertolozzi
Christian Jenei

Department of Space, Earth and Environment
Division of Physical Resource Theory
Complex Systems
Chalmers University of Technology
Gothenburg, Sweden 2025

Search-on-the-Move Radar:
Multi-Route Planning using Machine Learning
Jonatan Bertolozzi
Christian Jenei

© Jonatan Bertolozzi, Christian Jenei, 2025.

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

Master's Thesis 2025
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: Digital Surface Model (DSM) data with an overlaid traversability graph generated by our graph builder. The visualization includes three radar domes to illustrate the multi-unit routing problem.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Search-on-the-Move Radar:
Multi-Route Planning using Machine Learning
Christan Jenei
Jonatan Bertolozzi
Department of Space, Earth and Environment
Chalmers University of Technology

Abstract

Search-on-the-move is the practice of maintaining surveillance while radar-equipped vehicles remain in motion. We study the variant in which several radar-equipped ground vehicles cooperate in real time, composing a set of interdependent routes that jointly protect a defended asset. Because the heuristic routing objective is non-differentiable, gradient-based optimization methods cannot be directly applied. Instead, we propose a neuroevolutionary method that couples Graph Neural Networks (GNNs) with covariance matrix adaptation evolutionary strategy (CMA-ES).

Digital Surface Model (DSM) height data are transformed into traversability graphs through morphological image processing, terrain-class masks, and skeletonisation. From the graph a scenario graph is derived according to the operations intent. The GNN assigns scores to nodes and edges, indicating their strategic relevance; a greedy Dijkstra-based method then connects high-score nodes into complete patrol routes. A domain-specific fitness function evaluates the resulting multi-route plan, rewarding wide geographic dispersion, time-synchronised cooperation, and uninterrupted radar coverage, particularly to detect and track potential threats along avenues of approach to the Keep-Out Zone (KOZ)

Across three contrasting landscapes our system produced two-vehicle patrols in seconds, consistently holding KOZ blind-time below set threshold. The resulting route plans were subsequently assessed with a vulnerability-analysis, revealing terrain and planning weaknesses prior to deployment, demonstrating the method's broader utility. The results demonstrate that neuroevolution is an efficient method for solving complex, non-differentiable search-on-the-move problems and offers practical potential for diverse military planning applications.

Keywords: Neural Network, Graph Neural Network, NeuroEvolution, Genetic Algorithm, Covariance Matrix Adaptation, Evolution Strategy, Digital Surface Model, Radar, Route Planning, Search-On-The-Move.

Acknowledgements

We would like to extend our deepest gratitude to our supervisors, Håkan Warston and Niklas Doverbo, at Saab Surveillance. Special thanks to Håkan Warston for his invaluable discussions and teaching of radar and military tactics theory, as well as for being our primary contact throughout this journey. His guidance has been instrumental in shaping this thesis. Niklas Doverbo provided essential support with system requirements and navigated any hindrances of our work.

We are also sincerely thankful to our examiner, Claes Andersson, from the Department of Space, Earth, and Environment, for taking the time to oversee this thesis. His insightful and discussions surrounding this work, including those in the context of evolutionary systems, have added a fascinating dimension to the project.

A heartfelt thanks goes to Saab Surveillance, including Håkan and Niklas, for granting us the opportunity to work with such interesting and challenging problems. Being able to explore cutting-edge techniques at Saab, particularly in the fields of radars and machine learning, has been an incredibly rewarding and enjoyable experience.

We would also like to express our gratitude to Chalmers University of Technology for providing a wonderful and enriching environment during our time as students. The knowledge and experiences gained at Chalmers have been invaluable.

Finally, we extend our appreciation to the previous contributors whose work laid the foundation upon which this thesis is built. Your efforts have significantly guided and inspired our own.

Jonatan Bertolozzi
Christian Jenei
Gothenburg, June 2025

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in subject order:

NN	Neural Network
ANN	Artificial Neural Network
MLP	Multilayer Perceptron
GNN	Graph Neural Network
GAT	Graph Attention Network
GCN	Graph Convolutional Network
ES	Evolutionary Strategies
GA	Genetic Algorithm
CMA-ES	Covariance Matrix Adaptation Evolution Strategy
SepCMA-ES	Separable Covariance Matrix Adaptation Evolution Strategy
NEAT	NeuroEvolution of Augmenting Topologies
DSM	Digital Surface Model
CRS	Coordinate Reference System
GeoTIFF	Geospatial Tagged Image File Format
OSM	OpenStreetMap
BFS	Breadth-First Search
LOS	Line of Sight
DA	Defended Asset
KOZ	Keep-Out Zone

Nomenclature

Below is the nomenclature of indices, sets, parameters, and variables used throughout this thesis.

Indices

i, j	Generic indices for nodes/pixels or their neighbors in a graph/skeleton
ℓ	Layer index in neural networks (e.g., $\ell = 1, 2, \ldots, L$)
m	Dimension or rank index (e.g., for output size or transform matrices)
n	Iteration index (e.g., for morphological skeletonization or training samples)
r	Route index (e.g., for route R_r)
t	Time-step or algorithm iteration counter

Sets

C	Coverage polygon of a node
C_r	Ideal circular coverage area with radius r_{cov} (notation from source text)
$\mathcal{V}$, $\mathcal{E}$	Node set $\mathcal{V}$ and edge set $\mathcal{E}$ in a graph
$\mathcal{E}_Z$	Set of coverage events occurring in zone Z
$\mathcal{N}(v)$	Neighbor set of node v
$P^{(t)}$	Population at iteration t (e.g., in Genetic Algorithms or CMA-ES)
DA	Polygon representing the Defended Asset zone
KOZ	Polygon representing the Keep-Out Zone
REST	Polygon representing the Remaining Area zone
R_r	The r-th route, an ordered list of nodes

$S(X)$	Morphological skeleton of X
$\mathcal{V}_{\text{must}}$	Subset of graph nodes that must be visited (pathfinding)
Z	Generic zone polygon (DA, KOZ, or REST)

Parameters

α_{time}	Coefficient for route duration variance penalty (also α_{rtp})
α_Z	Fraction of zone Z covered by a node's visibility polygon (node feature, e.g., α_{DA})
β_{time}	Coefficient for exponential excess duration penalty (also β_{exp})
B_{opp}	Bonus coefficient for opposite-angle simultaneous coverage
$c_\sigma, c_\mu, c_1, c_c, d_\sigma$	Typical CMA-ES learning-rate parameters
$d(n_i, n_j)$	Distance of the edge between nodes n_i and n_j
δ_{angle}	Angle difference threshold for synergy calculation (distinguishing same vs. different angles)
δ_t	Time window threshold for considering coverage events simultaneous
ϵ	Small constant to prevent division by zero in cluster penalty calculation
γ_{time}	Rate for exponential excess duration penalty (also γ_{rate})
$h_{\text{curvature}}$	Curvature offset in viewshed calculations, $\frac{r^2}{2\,R_{\text{eff}}}$
k	Atmospheric refraction coefficient (typ. $k = 4/3$)
k_{route}	Number of nodes in a specific route (used for averaging, e.g. centroid calculation)
λ	Population size (in Genetic Algorithms / CMA-ES)
m_Z	Normalized magnitude of the offset vector from zone Z's centroid to coverage intersection centroid (node feature, e.g., m_{DA})
μ	Number of selected parents for recombination
μ_{eff}	Effective number of parents (weighted) in CMA-ES
$P_{\text{clus\_coeff}}$	Coefficient for the per-route cluster penalty (used as P_{clus} in the numerator of the P_r^{clus} formula in source text)
P_{same}	Penalty coefficient for same-angle simultaneous coverage
R	Total number of routes in a solution
R_e	Earth's radius
R_{eff}	Effective Earth radius, $R_{\text{eff}} = k\,R_e$
r_{cov}	Radius of the ideal radar coverage circle (used in C_r)

ρ_{cov}	Coverage ratio: actual intersection area with the ideal radar circle divided by the circle's area (node feature)
σ_{shape}	Polsby–Popper circularity of the coverage polygon (node feature, $0 \leq \sigma_{\text{shape}} \leq 1$)
T_{th}	Time duration threshold for penalizing long routes
θ_Z	Normalized angle of the offset vector for zone Z within its respective zone (node feature, e.g., θ_{DA})
v_{max}	Local maximum travel speed at a node (node feature, km/h)
w_{cov}	Weighting factor for coverage ratio ρ_{cov} in event quality
w_{exp}	Weighting factor for the exploration reward
w_{shape}	Weighting factor for Polsby–Popper circularity σ_{shape} in event quality
w_Z	Zone-specific weight for zone $Z \in \{\text{DA}, \text{KOZ}, \text{REST}\}$ (e.g., w_{DA})
x, y	Normalized x and y coordinates of a node (in $[0, 1]$, node feature)

Variables

$\mathbf{a}^{(l)}$	Activation for layer l in a neural network
α, θ	Azimuth and elevation angles (viewshed general)
α_{vu}	Normalized attention scores in Graph Attention
$\mathbf{b}^{(l)}$	Biases for the l-th layer of a neural network
$\mathbf{C}$	CMA-ES covariance matrix
ΔT	Excess duration of the longest route over T_{th}
$\Delta\theta$	Absolute difference in angles between two simultaneous coverage events
$\Delta x, \Delta y$	Offset components for coverage placement (e.g., $\text{IC}_x - \text{ZC}_x$)
e_{vu}	Unnormalized attention scores in Graph Attention
ev_i, ev_j	Coverage events (indexed i, j when comparing two events)
$\mathbf{h}_v^{(l)}$	Node embedding of node v at layer l (e.g., in GNNs)
$H_{\text{terrain}}(x, y)$	Terrain elevation at coordinate (x, y)
$(\text{IC}_x, \text{IC}_y)$	Coordinates of the centroid of the intersection $C \cap Z$
$\mathbf{m}, \sigma$	CMA-ES mean and global step-size
$\mu_{T_{\text{end}}}$	Mean of route durations T_{end}^r (denoted as μ in source text for route durations, distinguished from CMA-ES μ)
$\overline{d}$	Average distance of a route's nodes from its centroid
P_r^{clus}	Clustering penalty for route r

P_{clus}	Total clustering penalty for a solution (sum of P_r^{clus})
P_{dur}	Exponential penalty for excess route duration
P_T	Total time-related penalty ($P_{\text{var}} + P_{\text{dur}}$)
P_{var}	Variance penalty for route durations
$\mathbf{p}_\sigma, \mathbf{p}_c$	Step-size and covariance evolution paths in CMA-ES
Q_Z	Total raw quality of coverage events in zone Z
R_Z	Exploration reward for zone Z
$S_\cup$	Overall score from synergy, raw quality, and exploration across all zones
S_Z	Synergy score for zone Z
σ_T^2	Variance of route durations T_{end}^r
$t(n)$	Arrival time at node n
t_i	Arrival time at i-th node of a route (also used as generic event time t_i, t_j in source text for events ev_i, ev_j)
T_{end}^r	Total travel duration of route r
$T_{\max}$	Maximum travel duration among all routes in a solution
τ	Travel time for an edge
θ_i, θ_j	Coverage placement angles of events ev_i, ev_j (distinct from general viewshed θ)
$\overline{v}$	Average travel speed over an edge
$\mathbf{W}^{(l)}$	Weights for the l-th layer of a neural network
$\bar{x}, \bar{y}$	Coordinates of the spatial centroid of a route's nodes
$\mathbf{x}_i, \mathbf{z}_i$	i-th candidate solution and standard normal sample (CMA-ES)
(x_r, y_r, h_r)	Radar position and height above terrain (distinct from normalized node x, y)
$\mathbf{z}^{(l)}$	Linear transform output for layer l in a neural network
z_{eff}	Effective altitude after curvature, $z - h_{\text{curvature}}$

Contents

List of Figures

List of Tables

1 Introduction

The planning of a military mission is a highly complex and dynamic process that requires accurate coordination between different operational units, including command, artillery, and sensor units. These units must operate together to maximize effectiveness while adapting to the continuously and fast paced changes that military planning requires. Mission planning is typically conducted in advance but must remain and allow for adjustments based on real-time observations and newly gathered intelligence.

One critical part of mission planning involves the deployment and movement of land-based sensor units, such as radar systems used for surveillance and target detection. Systems such as Saab's Giraffe 1X radar (G1X) play an important role in monitoring areas of interest and maintaining high airspace awareness in different military missions. The G1X is a lightweight high performance radar system that weights less than 150 kg total [2]. This weight class ensures a diverse set of deployment possibilities as it can be transported by utility vehicles such as trucks. Its mobility and agility allow it to maintain operability while relocating, supporting the possibilities of quickly reevaluate the routing during transportation. The concept of dynamic relocation while performing a intelligence gathering is called search-on-the-move. However, this capability introduces a core planning concern: how can such a system plan and execute safe, informative and efficient routing through complex terrain under uncertainty and time pressure.

Sensor survivability is a major concern in this context. Since radar emissons can be detected and geolocated by adversaries, longer exposure in one location can compromise the safety of a unit. Therefore, a successful routing policy must minimize the unit's exposure while maintaining coverage and line of sight to the operationally relevant areas. This also raises a number of critical concerns that must be adressed: How can we model terrain aware exposure and visibility of a sensor dynamically while moving. Further the tradeoffs between sensor coverage and path efficient routing must be considered under the constraints of real time optimization.

However, maintaining operational effectiveness with these types of moving systems comes with several challenges. Sensor units must be routed strategically to maximize coverage while minimizing the risk of detection by enemy forces. This typically involves complex schemes of monitoring the same airspace using different sensors, this since radar emissions can be intercepted by an adversary and used to locate the sensor units. Therefore, these units can not stay stationary for too long, to minimize the risk of the adversary being able to calculate an accurate estimate of its location, and hence endanger the safety or survival of this unit. This criterion is especially relevant for search-on-the-move routing of the G1X system since it is capable of maintaining operability during relocation. With an ever increasing demand for innovative defense and intelligence solutions it calls for an agile real-time routing system that enables continuous intelligence gathering during which you are able to minimize or gain and understanding of the exposure of the operating unit.

The deployment of sensor units is further complicated by environmental factors such as terrain, vegetation, and man-made objects such as buildings and infrastructure. High resolution geospatial data, like that developed by Saab in cooperation with Vricon, allows for precise mapping of the terrain and objects, significantly improving the capabilities for an accurate routing. However, such precise data also provides challenges for a real-time decision making system since it poses computational challenges to operate at a high resolution. Therefore, it has also been part of this thesis to investigate the possibilities of efficiently processing large volumes of terrain data while maintaining rapid optimization and fast inference times for routing.

2
Background

2.1 Aim

The primary aim of this thesis is to develop and evaluate a novel neuroevolutionary framework for agile, real-time, and multi-vehicle route planning for land based search-on-the-move (SOTM) radar systems. Specifically, this research addresses the challenge of optimizing non-differentiable routing objectives. The new contribution is the coupling of Graph Neural Networks (GNNs), for learning strategic terrain relevance, with the Covariance Matrix Adaptation Evolution Stratergy (CMA-ES), for gradient free optimization. The goal is to create a decision support system that enables dynamic repositioning based on operational needs, and environmental constraints. The outcome and novelty of this thesis is a proof-of-concept system that implements the framework and demonstrates its efficacy in generating and evaluating routing strategies.

The goal is to create a proof-of-concept decision support system capable of dynamic and agile replanning in response to evolving operational needs and environmental constraints

2.2 Limitations

This research, is aiming for a a proof-of-concept solution to provide verification for the methodologies employed and the results obtained. The study is based on a finite set of Digital Surface Models (DSMs), representing a specific number of distinct terrains for training and validation. While these terrains offer diversity, the models' exposure is inherently limited to these environments, which may affect the broader generalization of findings to all possible operational landscapes.

Certain environmental variables were simplified or excluded from the scope of this work. Notably, dynamic factors such as changing weather conditions, which can significantly impact sensor performance and ground traversability, were not modeled. Similarly, the load-bearing capacity, a critical factor for real-world vehicle mobility, was not considered. These simplifications mean that the developed planning capabilities operate under idealized environmental assumptions.

A significant operational constraint was the lack of access to a dedicated high-performance computational or training cluster. This resource limitation influenced the choice of optimization algorithms and the extent of experimentation possible. For instance, it necessitated the use of more memory-efficient variants of optimization strategies, such as Separable CMA-ES, rather than more exhaustive methods like full CMA-ES. This also constrained the number of training iterations, the scale of hyperparameter searches, and the diversity of scenarios that could be practically generated and tested.

2.3 Related Work

2.3.1 Earlier Work

This thesis is a part of a series of thesis work conducted at Saab Surveillance, Gothenburg, focused on applying high-resolution geospatial data to scheduling and deployment planning problems. Previous work in this series has established some foundation for some of the methodologies used in this Thesis. The first thesis [19] addressed schedule planning for individual artillery hunting radar systems (ARTHUR), using genetic algorithms (GA) and skeletonization for traversability graphs to optimize deployment schedules that balanced surveillance with survivability by minimizing detection probabilities. Subsequent work [18] extended this by once again using GAs to solve a optimization problem. This time static deployment for entire air defence companies, considering multiple sensor and fire units to maximize overall operational capacity.

Collectively, these preceding theses demonstrated the utility of evolutionary algorithms using high-resolution geospatial data for planning optimization. However, they primarily focused on sequentially or static planned deployments, respectively. This present thesis continues this research by addressing the more complex challenge of real-time, cooperative multi-route planning for SOTM systems, which introduces a new set of requirements. Previous methods using direct optimization, where generating or adapting solutions require a full-scale re-optimization. Such an approach would not be feasible for the requirements of a low latency SOTM problem. Therefore, this thesis investigates the strategy of a pre trained model. This enables computationally inexpensive inference required to be real time adaptable, by decoupling the optimization from the planning.

2.3.2 Neuroevolution

Neuroevolution refers to the application of evolutionary algorithms to optimize artificial neural networks (ANNs). Rather than relying on gradient-based methods such as backpropagation, neuroevolution evolves a population of candidate networks over several generations based on a fitness function. This process does not require differentiable loss functions or smooth loss landscapes, making it naturally suited to non-differentiable or noisy optimization tasks.

2.3.2.1 Motivation and Historical Context

Early neuroevolution methods primarily focused on small-scale networks or narrowly defined tasks. A landmark development was the introduction of NeuroEvolution of Augmenting Topologies (NEAT) [25], which employed evolutionary mechanisms to iteratively expand network architectures along with optimizing weights. By exploring both topology and parameter spaces, NEAT illustrated neuroevolution's ability to discover innovative architectures that might not emerge from conventional gradient-based searches.

2.3.2.2 Approaches and Encodings

- **Weight Evolution Only:** With a fixed network architecture, evolutionary operators mutate real-valued weights encoded in a genome. This approach is particularly advantageous when gradient information is unreliable or when the optimization task is non-differentiable.
- **Topology & Weight Evolution:** More sophisticated methods, such as NEAT, allow for the simultaneous evolution of network topologies and weights. This dual evolution can lead to the discovery of novel architectures that enhance model performance beyond what traditional gradient descent may achieve.

2.3.2.3 Recent Advances in Deep Neuroevolution

Recent studies have demonstrated that even high-capacity deep neural networks can be effectively evolved using gradient-free genetic algorithms. For instance, work on *Deep Neuroevolution* shows that a simple genetic algorithm can successfully train deep reinforcement learning agents on challenging tasks with millions of parameters [26]. Moreover, gradient-free neuroevolutionary approaches benefit from an exploration-oriented global search strategy that renders them less prone to getting trapped in spurious local minima. Such advantages have been highlighted in recent research efforts [30, 29, 11], which demonstrate that neuroevolution can excel in scenarios where gradients are unreliable or the loss landscape is particularly rugged.

3

Theory

3.1 Digital surface model

In the collection of data for large terrain regions, challenges arise due to the curvature and irregularity of the Earth's surface. While approximating the terrain as a flat surface for small scales is trivial, the inaccuracies of this approximation grows with the scale of the terrain target. This demands more precise methods to manage and represent the data.

A common and very useful file format to deal with these challenges and to accurately store geographically derived data is using the Geographically tagged image file format (GeoTIFF). The format allows the image data to be paired with metadata that is used for georeferencing. This includes storing the coordinate reference system (CRS), map projection, and transformation parameters, which ensures accurate representation of the data.

To represent terrain in file format, it is common practice to use digital surface models (DSM), this is used to capture the Earths surface elevation and including permanent object that covers it, for instance buildings and vegetation. The data is represented by a raster grid where each pixel corresponds to a spatial resolution, throughout this report the Vricon data has a resolution of 0.5 meters per pixel. This level of detail enables identification of small terrain features and obstacles that impact both radar positioning and a capability of accurately simulate the viewshed, discussed in 3.6. For comparison, Digital Terrain Elevation Data (DTED) level 2, which has been the standard for many military applications, offers approximately 30 meters spacing [7], while even enchanced DTED level 3 provides only a 10 meters resolution. The 60 fold improvement in resolution from DETD level 2 to Vricons 0.5 meter data, transforms whats possible in terrain analysis and route planning. Small but significant features such as buildings like buildings, tree lines or narrow roads are entirely missed or distorted in lower resolution.

To build an idea of what the terrain looks like it is not only sufficient to use only DSM data, since there are properties of the terrain that can not be inferred from only the elevation. Such as which medium the terrain consists of which in turn is highly important to produce estimates of other properties, such as load bearing capacity of the terrain. To solve this problem a classification information is needed. Vricon along with the DSM data also provides terrain classification information, with different classes, see table 3.1 for types of terrain classification.

Class	Description
No data	-
Uncertain	-
Ground	Grass, dirt, etc.
Low vegetation	0.5 - 2 m
Medium vegetation	2 - 5 m
High vegetation	> 5m
Man-made	Buildings etc.
Water	-
Paved surface	Roads etc.

Table 3.1: Presents the Vricon terrain classification classes, and describes what is contained withing each class.

3.2 Image Processing

Most of the methods used to process the data are binary morphological operations, the point here is that we are interested in a rough estimate of which parts of the terrain that is traversable, it is not optimal but working with a binary definition of traversability significantly reduces the complexity of our problem and enables simpler image processing operations.

3.2.1 Dilation and Erosion

Binary dilation is a morphological transformation [15], given a binary image A and a structuring element B the dilation is defined as

$$A \oplus B = \{x | B_x \cap A \neq \emptyset\}, \tag{3.1}$$

where x is the coordinate B_x is the shifted structuring element. The operation places the structuring element B at every pixel in A and adds x to the output if at least one pixel in B_x overlaps with a pixel in A. This results large objects expanding and becoming more visible also filling small holes in the image. On the contrary binary erosion is instead defined as

$$A \ominus B\{x | B_x \cap A^c = \emptyset\} \tag{3.2}$$

where A^c is the complement of A. Meaning that no part of B should intersect with the background, A^c, therefore a partial intersection of A is removed. This results in thinning of lines, objects and deletion of objects smaller than the structuring element. The choice of structuring element is essential to define the desired function of both operations.

3.2.2 Opening and closing

Opening and closing are two useful operations derived from the combinations of above mentioned dilation and erosion [15]. These are used to remove noise, smoothing boundaries and removing holes in images.

$$A \circ B = (A \ominus B) \oplus B, \tag{3.3a}$$
$$A \bullet B = (A \oplus B) \ominus B. \tag{3.3b}$$

The opening ($\circ$) of a image A shrinks objects by erosion to remove noise and then restores main shape with dilation. This preserves overall shape but small elements are deleted. The closing ($\bullet$) is the reverse operation order and fills small gaps while retaining the original shape.

3.3 OpenStreetMap

OpenStreetMap (OSM) is a open-source project aimed at creating a comprehensive global map database based through user contribution [12]. The OSM database is structured by three fundamental elements. First the nodes which are point features defined by geographic coordinates, latitude and longitude. Second the ways which are linear or area features composed of ordered nodes, and are used to represent entities such as roads, building footprints or land use boundaries. Third and final are the relations, these are more advanced features that describe relationships between multiple elements, such as multipolygons with holes or traffic rules at a specific intersection. Each element can have multiple tags or so called key-value pairs that describe the feature's attributes. For instance, a road might have the tag highway and maxspeed to indicate the class and speed limit and number of lanes to document the configuration.
The database and accessing its data is freely available without copyright, license or usage fees for any purpose and is covered by the Open Data Commons Open Database License (ODbL) [23]. This has lead to a large ecosystem of applications and tools built based on OSM.
Despite the highly detailed information, it relies on individual contributors therefore there is no completeness and correctness guarantee. However the fetched data from the database can conveniently be used to supplement and update the terrain information and gather other useful properties in the area of interest. As an example this could be road information such as speed limit that the original terrain classification 3.1 does not provide.

3.4 Morphological skeleton

Morphological skeleton is a transformation that reduces a binary shape to a thin line representation, i.e. the skeleton, while preserving its topological and geometric properties. The skeleton can be seen as the medial axis, i.e. the set of points that have at least two closest points to the shapes boundary, that maintains the connectivity and basic structure. The formal definition of the skeleton $S(X)$ of a binary image X can be written as

$$S(X) = \bigcup_{n=0}^{\infty} (X \ominus nB) \setminus ((X \ominus nB) \circ B) \tag{3.4}$$

where B is the structuring element used for the operations. $(X \ominus nB)$ denotes erosion of X by B applied n times, see section 3.2.1. The set difference ensures that only the boundary pixels at each stage contribute to the skeleton. This process iteratively removes the boundary pixels until only the skeleton is leaved.

Algorithm 1 Skeletonization of Binary Images

Require: Binary image I.
Ensure: Skeletonized image S.
  Choose Method:
  if Thinning **then**
    while Changes in I **do**
      Remove boundary pixels while preserving connectivity.
    end while
  else if Medial Axis **then**
    Compute distance transform of I.
    Retain ridge points as the skeleton.
  end if
  Post-process S to clean and smooth the skeleton.
  return S.

3.5 Graph Algorithm

In order to transform the skeletonized image into a graph-based representation suitable for navigation, the algorithm systematically identifies junctions and links via the pixel values. Instead of stopping at endpoints that do not support further navigation decisions, junctions are identified by examining the pixel neighbors within the skeleton. These junctions represent locations where the route diverges and navigation choices must be made.

The algorithm begins by scanning every pixel in the binary skeleton image. A pixel is considered a junction if it is white and has more than two white neighbors in its 8-connected neighborhood. This criterion is meet, this pixel is recorded as a node.

To maintain the link between these junctions, the algorithm constructs a pixel-level connectivity map. Here, all skeleton pixels are temporarily treated as nodes in a secondary graph, with edges reflecting direct neighbor-to-neighbor connections. This pixel map visualizes the interconnection and provides a reference structure for a breadth-first search (BFS). The BFS traversal enables the algorithm to connect junctions directly, establishing edges between them in the final graph.

The following pseudocode outlines this method, from identifying junctions and building pixel-level connectivity to performing BFS-based exploration that connects junctions:

Algorithm 2 Graph Construction from Skeletonized Image

```
Input: Binary skeleton image S
Output: Graph G with junctions
                                    ▷ 1. Identify Junctions and Build Pixel Graph
for each (x, y) in S where S[x, y] = 1 do
    if neighbors(x, y) > 2 then
        Add junction node n at (x, y) to G
    end if
    for each neighbor (x', y') where S[x', y'] = 1 do
        Connect (x, y) ↔ (x', y') in pixel graph P
    end for
end for
                                    ▷ 2. Connect Junctions via BFS
for each junction u in G do
    Perform BFS from u in P to find paths to other junctions
    Add edges (u ↔ v) to G for all junctions v reached
end for
return G
```

3.6 Viewshed Algorithm

A viewshed algorithm estimates which regions of the landscape are directly visible from a radar's position by casting rays through a digital surface model (DSM) and checking for line-of-sight (LOS) obstructions. The algorithm accounts for both the curvature of the Earth and atmospheric refraction to generate realistic visibility approximations over large distances.

Radar systems, especially those operating over extended ranges, must incorporate Earth's curvature into the LOS calculations. Without this consideration, the viewshed would tend to overestimate visible areas as distance increases. In addition, atmospheric refraction slightly bends radar waves toward the ground, effectively increasing the radar's visible range. This phenomenon is modeled by adjusting Earth's radius as follows:

$$R_{\text{eff}} = kR_e,$$

where R_e is the Earth's radius and k is a refraction coefficient typically approximated as

$$k = \frac{4}{3}$$

under standard atmospheric conditions [1]. Given a radar positioned at coordinates (x_r, y_r) and at a height h_r above the terrain, the algorithm computes visibility along discrete azimuth α and elevation θ directions. For a range increment r, the point along the ray is determined by:

$$
\begin{aligned}
x &= x_r + r\cos(\alpha)\cos(\theta), \\
y &= y_r + r\sin(\alpha)\cos(\theta), \\
z &= h_r + r\sin(\theta).
\end{aligned}
$$

To account for the effect of Earth's curvature, the ray's height is adjusted by:

$$h_{\text{curvature}} = \frac{r^2}{2R_{\text{eff}}},$$

yielding an effective altitude

$$z_{\text{eff}} = z - h_{\text{curvature}}.$$

At each computed point, the terrain elevation $H_{\text{terrain}}(x, y)$ obtained from the DSM is compared with z_{eff}. If $H_{\text{terrain}}(x, y)$ exceeds z_{eff}, the ray is deemed obstructed; otherwise, it continues to the next range increment.

Reference Plane Algorithm

In scenarios where sightline sampling points do not conveniently align with the Digital Surface Model (DSM) grid, the standard line-of-sight (LOS) approach can become computationally intensive. A more efficient alternative, known as the reference plane algorithm [28], is designed to accelerate viewshed computation, particularly when dealing with such non-aligned sightlines. By establishing an auxiliary grid, let us denote it as G_{aux}, which shares the same dimensions as the DSM. However, instead of storing raw terrain elevations, G_{aux} is populated with a critical elevation for each grid point (i, j), denoted $H_{\text{crit}}(i, j)$. This critical elevation represents the minimum height a target at the X-Y location of (i, j) would need to possess to be visible from the observer's position S, considering all blocking elements by terrain encountered along the path from S up to that cell. This concept of storing an evolving effective horizon can be viewed as quite similar to the Line-of-Sight (LOS) function described in [28], which summarizes intervisibility information. The auxiliary grid G_{aux} is populated progressively as the viewshed analysis proceeds outward from the observer.

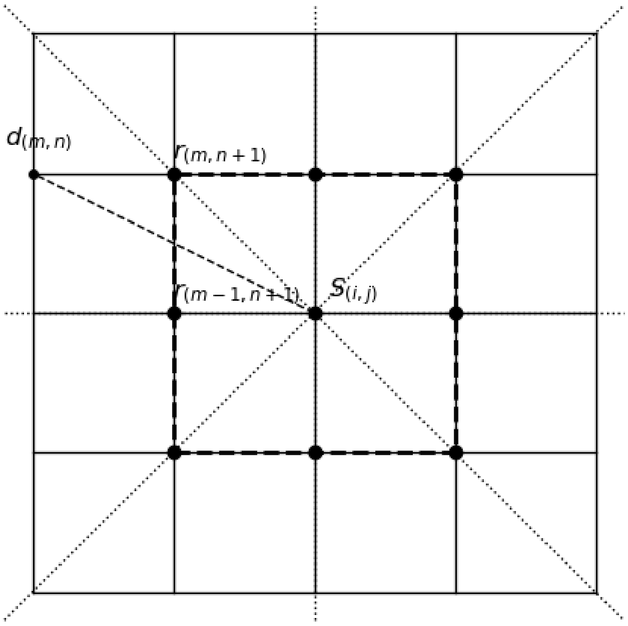


Figure 3.1: An illustration of the reference plane algorithm. The viewpoint $S_{i,j}$ is the observer's position. Points $r_{(m,n+1)}$ and $r_{(m-1,n+1)}$ serve as reference points for determining the visibility of the destination $d_{(m,n)}$ on the auxiliary grid.

The process, as illustrated in Figure 2.1 where the observer S is denoted by $S_{i,j}$, involves defining a reference plane to determine the visibility of a destination point $d(m,n)$. This plane is constructed using three points: firstly, the observer's position S; secondly and thirdly, two reference points whose Z-coordinates are derived from the critical elevations stored in G_{aux}. Specifically, these are a point P_1 at the X-Y coordinates of $r(m,n+1)$ with elevation $H_{\text{crit}}(m,n+1)$, and a point P_2 at the X-Y coordinates of $r(m-1,n+1)$ with elevation $H_{\text{crit}}(m-1,n+1)$, as shown in the figure. The grid locations $r(m,n+1)$ and $r(m-1,n+1)$ are strategically chosen as they lie on the auxiliary grid in cells positioned behind the destination $d(m,n)$ relative to the observer S. The H_{crit} values at these locations thus captures the cumulative blocking effects determined up until those points. The destination point $d(m,n)$ is hence determined visible if its actual terrain elevation, $H_{\text{DSM}}(m,n)$ obtained directly from the DSM, lies on or above this constructed reference plane. This simplifies the visibility assessment from the standard sight line computations with a single plane intersection test. Also sped up by utilising the intermediate computation stored in the auxiliary grid, G_{aux}.

3.7 Pathfinding Algorithms

3.7.1 Dijkstra's Algorithm

One of the most well-known algorithms for finding the shortest path from a single source node to all other nodes in a graph with nonnegative edge weights is *Dijkstra's algorithm* [8]. Given a weighted graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ and a source node $v_s \in \mathcal{V}$, the goal is to determine the minimum cumulative cost of reaching every other node $v \in \mathcal{V}$ from v_s.

Dijkstra's algorithm maintains a set of visited nodes for which the minimum-cost path from v_s has been definitively determined, and a set of unvisited nodes for which costs are still preliminary. Initially, only the source node v_s is known to have a preliminary distance of zero, while all other nodes are assigned a large initial distance.

At each step, the algorithm:
1. Selects the unvisited node u with the smallest current distance (i.e. the node that appears cheapest to reach from v_s).
2. Moves u into the visited set. This action finalizes the cost of reaching u.
3. Updates the distances of neighbors of u. Specifically, if the path through u to a neighbor w offers a lower cost than the currently recorded cost for w, that cost is replaced with the new value.

This process repeats until either all nodes become visited or the target node has its distance finalized. Since it always expands the globally smallest distance node first, Dijkstra's algorithm guarantees an optimal solution for graphs with nonnegative edge weights. Using a binary heap, the time complexity is $O(|E| \log |V|)$ in typical implementations [9].

3.7.2 Greedy Path-Finding Algorithm

While Dijkstra's algorithm finds a shortest path in terms of a predefined cost metric, a related but more heuristic method for constructing routes through multiple specific nodes is the greedy path-finding approach. In particular, suppose there is a subset of nodes $\mathcal{V}_{\text{sub}} \subseteq \mathcal{V}$ that must be visited by the final path. A greedy strategy can build this path iteratively:
1. Start from an initial node $v_{\text{start}} \in \mathcal{V}_{\text{sub}}$.
2. At each step, choose the next node $\in \mathcal{V}_{\text{sub}}$ that has the least cost to the current node.
3. Mark the chosen node as visited and append it to the route.
4. Continue until all nodes $\mathcal{V}_{\text{sub}}$ have been incorporated.

3.8 Artificial Neural Networks

Artificial Neural networks (ANNs) are a class of parametric machine learning models inspired by the structure and function of biological neural systems. They are composed of interconnected layers of neurons, which transform an input vector through successive linear and non-linear operations to produce an output prediction [21]. Formally, let $\mathbf{x} \in \mathbb{R}^{d_0}$ denote the input to the network. A feed-forward NN with L layers produces an output $\mathbf{y}$ by repeatedly applying affine transformations followed by non-linear activation functions:

$$\mathbf{z}^{(l)} = \mathbf{W}^{(l)}\mathbf{a}^{(l-1)} + \mathbf{b}^{(l)}, \quad \mathbf{a}^{(l)} = f\Big(\mathbf{z}^{(l)}\Big), \quad l = 1, 2, \ldots, L,$$

where:

- $\mathbf{a}^{(0)} = \mathbf{x}$ is the input,
- $\mathbf{W}^{(l)} \in \mathbb{R}^{d_l \times d_{l-1}}$ and $\mathbf{b}^{(l)} \in \mathbb{R}^{d_l}$ are learnable parameters for the l-th layer, where $\mathbf{W}^{(l)}$ (the weights) determine how strongly each input influences the output, and $\mathbf{b}^{(l)}$ (the biases) allow the activation to shift, enabling the model to fit the data [21],
- $f(\cdot)$ is a non-linear activation function such as the rectified linear unit (ReLU), sigmoid, or hyperbolic tangent (tanh).

The final layer $\mathbf{a}^{(L)}$ serves as the output layer of dimension d_L.

Training and Loss Functions

Neural networks typically learn to tune their parameters to minimize a loss function $\mathcal{L}$ via gradient-based optimization [21]. For a dataset $\{(\mathbf{x}^{(n)}, \mathbf{t}^{(n)})\}_{n=1}^{N}$, where $\mathbf{t}^{(n)}$ is the target output for input $\mathbf{x}^{(n)}$, the overall loss is often defined as the average over individual sample losses:

$$\mathcal{L}(\Theta) = \frac{1}{N} \sum_{n=1}^{N} \ell\Big(f(\mathbf{x}^{(n)}; \Theta), \mathbf{t}^{(n)}\Big),$$

where $f(\mathbf{x}; \Theta)$ denotes the network's output for input $\mathbf{x}$ given parameters Θ, and

$$\Theta = \Big\{\mathbf{W}^{(l)}, \mathbf{b}^{(l)}\Big\}_{l=1}^{L}$$

contains all weights $\mathbf{W}^{(l)}$ and bias $\mathbf{b}^{(l)}$ across the L layers of the network.

Training is commonly performed using gradient descent or other variants such as Stochastic Gradient Descent (SGD), Adam, or RMSprop [21]. These algorithms iteratively update the parameters in the direction opposite to the gradient of the loss function, aiming to reduce $\mathcal{L}$:

$$\Theta_{\text{new}} = \Theta_{\text{old}} - \eta \, \nabla_{\Theta} \, \mathcal{L}(\Theta), \tag{3.5}$$

where η is the learning rate controlling the step size in the optimization.

The backpropagation algorithm computes the gradients $\nabla_{\Theta} \, \mathcal{L}(\Theta)$ by applying the chain rule through the network layers, determining how small changes in each parameter—such as an early-layer weight $\mathbf{W}^{(l)}$—affect the overall loss.

Universal Approximation Theorem

Theorem 1. ***Universal Approximation:*** *Let $C(X, \mathbb{R}^m)$ denote the set of continuous functions from a subset $X \subseteq \mathbb{R}^n$ to $\mathbb{R}^m$. Suppose $\sigma \in C(\mathbb{R}, \mathbb{R})$ is an activation function such that $(\sigma \circ \mathbf{x})_i = \sigma(x_i)$ for each component i. Then σ is* not polynomial *if and only if, for* every *$n \in \mathbb{N}$, $m \in \mathbb{N}$, every compact set $K \subset \mathbb{R}^n$, every continuous target function $f \in C(K, \mathbb{R}^m)$, and every $\varepsilon > 0$, there exist:*

$$k \in \mathbb{N}, \quad A \in \mathbb{R}^{k \times n}, \quad \mathbf{b} \in \mathbb{R}^k, \quad C \in \mathbb{R}^{m \times k}$$

such that the neural network

$$g(\mathbf{x}) = C\Big(\sigma(A \cdot \mathbf{x} + \mathbf{b})\Big)$$

satisfies

$$\sup_{\mathbf{x} \in K} \Big\| f(\mathbf{x}) - g(\mathbf{x}) \Big\| < \varepsilon.$$

In other words, if σ is any non-polynomial activation function, then a sufficiently wide (i.e. for large enough k hidden units) single-hidden-layer feed-forward network with activation σ can approximate any continuous function on a compact domain K to arbitrary precision [17]. Polynomial activation functions would restrict the network's output to remain a polynomial of its inputs, whereas a non-polynomial σ enables a far richer class of functions to be represented [17, 21].

3.8.1 Graph Neural Networks

Graph Neural Networks (GNNs), generalize traditional neural network to handle data that is structured as graphs. A graph $G = (V, E)$ is defined by a node set V and an edge set $E \subseteq V \times V$. Each node $v \in V$ can have an associated feature vector $\mathbf{x}_v \in \mathbb{R}^d$. The primary goal of a GNN is to learn either node-level representations or an overall representation for the entire graph by iteratively passing and aggregating information between neighboring nodes.

Message Passing. A fundamental approach in GNNs is the message passing mechanism, often described in terms of three functions: *message*, *aggregation*, and *update* [31]. Let $\mathbf{h}_v^{(l-1)}$ denote the embedding of node v at layer $(l-1)$. Initially $\mathbf{h}_v^{(0)} = \mathbf{x}_v$. At the l-th layer, each node v gathers information from its neighbors $\mathcal{N}(v)$ to update its embedding $\mathbf{h}_v^{(l)}$

1. *Message Function M*: defines how each neighbor u computes the message $\mathbf{m}_{u \to v}^{(l)}$ that it sends to v, typically a function of $\mathbf{h}_u^{(l-1)}, \mathbf{h}_v^{(l-1)}$, and possibly edge features $\mathbf{e}_{u,v}$. Formally,

$$\mathbf{m}_{u \to v}^{(l)} = M\Big(\mathbf{h}_u^{(l-1)}, \mathbf{h}_v^{(l-1)}, \mathbf{e}_{u,v}\Big).$$

2. *Aggregation Function A*: combines the messages from neighbors $\mathbf{m}_{u \to v}^{(l)}$ for all $u \in \mathcal{N}(v)$, where $\mathcal{N}(v)$ is the set of neighbors of node v. This step is often

permutation-invariant (such as summation or averaging), ensuring that the order in which neighbor messages are processed does not affect the result, since graph neighbors have no inherent ordering:

$$\mathbf{m}_v^{(l)} = A\Big(\{\mathbf{m}_{u\to v}^{(l)} \mid u \in \mathcal{N}(v)\}\Big).$$

3. *Update Function U*: uses the aggregated message $\mathbf{m}_v^{(l)}$ and the previous embedding $\mathbf{h}_v^{(l-1)}$ to compute the new embedding $\mathbf{h}_v^{(l)}$:

$$\mathbf{h}_v^{(l)} = U\Big(\mathbf{h}_v^{(l-1)}, \mathbf{m}_v^{(l)}\Big).$$

where U is a learnable (parametrized) function, such as a feed-forward neural layer.

By stacking multiple layers, a node's representation at higher layers effectively encodes information from increasingly larger subgraphs, capturing both local and global properties. This is due to the growing receptive field. For a GNN with L message passing layers, the embedding $h^(L)_v$ of a node v is influenced by the information from all nodes within its L-hop neighborhood. That is, after one layer, $h^(1)_v$ incorporates information from v itself and its direct neighbors (1-hop) [22]. After two layers, $h^(2)_v$ incorporates information from nodes up to two hops away, and so on. Therefore, depth of the network directly controls the size of the subgraph, i.e. the receptive field, that contributes to each node's final representation.

3.8.1.1 Graph Attention Networks

Graph Attention Networks (GATs), introduced by Veličković et al.[27], incorporate a learnable attention mechanism into the message passing framework. This allows each node to dynamically assign different levels of importance to its neighbors during feature aggregation, thereby enabling a more flexible and expressive update rule compared to uniform or predefined distance-based weighting.

Attention Coefficients The operation of a single GAT layer begins by transforming the input features of each node v into
Consider a single GAT layer where each node v has an embedding $\mathbf{h}_v^{(l-1)}$. First, a linear transformation is applied to map the node's features into a new d'-dimensional space:

$$\mathbf{h}'_v = \mathbf{W}\,\mathbf{h}_v^{(l-1)},$$

where $\mathbf{W} \in \mathbb{R}^{d' \times d}$ is a learnable parameter matrix. Next, for each pair of connected nodes v and u, a shared parameter vector $\mathbf{a} \in \mathbb{R}^{2d'}$ is used to compute an unnormalized attention score:

$$e_{vu} = \mathrm{LeakyReLU}\Big(\mathbf{a}^\top [\mathbf{h}'_v \,\|\, \mathbf{h}'_u]\Big),$$

where $\|$ denotes vector concatenation, and LeakyReLU is the non-linear activation function. To make the scores comparable across different neighborhoods, they are normalized using the softmax function, yielding the attention coefficients:

$$\alpha_{vu} = \frac{\exp(e_{vu})}{\sum_{k\in\mathcal{N}(v)} \exp(e_{vk})}.$$

These nromalized coefficients, α_{vu}, reflect the relative importance of neighbor u features in the context of updating the representation of node v.

Feature Aggregation via Attention. Once the attention coefficients are computed, the updated embedding $\mathbf{h}_v^{(l)}$ for node v is obtained by a weighted linear combination of its neighbors' transformed features:

$$\mathbf{h}_v^{(l)} = \sigma\Big(\sum_{u \in \mathcal{N}(v)} \alpha_{vu}\, \mathbf{h}'_u\Big),$$

where σ is a non-linear activation function (e.g. ReLU). Intuitively, α_{vu} determines the relative importance of node u's features when updating v. This allows the model to selectively aggregate information based on learned relationships between nodes.

Multi-Head Attention. To stabilize the self-attention learning process, GAT often employs multi-head attention. That is, the above procedure is replicated K times with independent parameter sets $\{\mathbf{W}^{(k)}, \mathbf{a}^{(k)}\}_{k=1}^{K}$, and the results are then aggregated—either concatenated or averaged—to form the final embedding:

$$\mathbf{h}_v^{(l)} = \begin{cases} \|_{k=1}^{K}\, \sigma\Big(\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^{(k)}\, \mathbf{W}^{(k)} \mathbf{h}_u^{(l-1)}\Big), & \text{(concatenation)} \\ \dfrac{1}{K} \displaystyle\sum_{k=1}^{K} \sigma\Big(\sum_{u \in \mathcal{N}(v)} \alpha_{vu}^{(k)}\, \mathbf{W}^{(k)} \mathbf{h}_u^{(l-1)}\Big), & \text{(averaging)} \end{cases}$$

where $\|$ denotes concatenation, $\alpha_{vu}^{(k)}$ is the attention coefficient for the k-th head, and averaging simply takes the mean over heads. This enables each head to focus on different feature subspaces and types of attention patterns [21].

3.9 Genetic Algorithm

Genetic Algorithms (GAs) are adaptive heuristic search techniques inspired by Darwin's principle of natural selection that falls under the Evolutionary Algorithm (EA) umbrella [16, 5]. They are used to solve optimization and search problems by evolving a population of candidate solutions referred to as individuals or genomes over multiple generations. Each individual represents a potential solution encoded in a specific format, for example, a string of bits or a vector of real numbers, analogous to the composition of a chromosome. A GA generally consists of the following steps:

1. **Initialization:** Generate an initial population of solutions using some distribution, often uniform or random, but which can be adapted to problem-specific strategies to guide the search of solutions.
2. **Evaluation:** Compute each individual's fitness based on a predefined objective function.
3. **Selection:** Choose a subset of individuals from the population based on their fitness scores. The goal is to balance exploitation of high-fitness individuals with exploration to maintain diversity (see 3.9.4).

4. **Crossover:** Combine pairs of selected individuals to produce *offspring* solutions.
5. **Mutation:** Make small random alterations to some offspring.
6. **Replacement:** Form the new population for the next generation.

This cycle repeats until a termination condition is satisfied, such as reaching a maximum number of generations or achieving a target fitness value.

3.9.1 Fitness Evaluation

Given an individual x from the search space $\mathcal{X}$, the fitness function $f : \mathcal{X} \to \mathbb{R}$ maps x to a scalar value that reflects how well x meets the optimization objective.

3.9.2 Crossover

Crossover, is the mechanism by which two parent solutions combine their genetic makeup to produce child or offspring solutions. The rationale is that good solutions may yield even better offspring by mixing their characteristics.
To illustrate, let us consider two parent genomes, each represented by a binary string of length 8:

$$\text{Parent}_1 = 1\,0\,1\,1\,0\,0\,1\,0, \quad \text{Parent}_2 = 1\,1\,1\,1\,0\,0\,0\,0.$$

One-point crossover at position 4 might produce:

$$\text{Child}_1 = 1\,0\,1\,1 \mid 0\,0\,0\,0, \quad \text{Child}_2 = 1\,1\,1\,1 \mid 0\,0\,1\,0,$$

where the vertical bar $|$ indicates the crossover position. The first 4 bits of Parent_1 are combined with the last 4 bits of Parent_2 to create Child_1, and vice versa for Child_2.

3.9.2.1 Simulated Binary Crossover (SBX)

Simulated Binary Crossover (SBX) is commonly used when solutions are represented by real-valued vectors (as opposed to binary strings). Developed by Deb and Beyer [6], SBX simulates the behavior of single-point crossover in binary-coded genetic algorithms for continuous parameter spaces. Given two parents $\mathbf{x}^{(1)} = (x_1^{(1)}, x_2^{(1)}, \ldots, x_n^{(1)})$ and $\mathbf{x}^{(2)} = (x_1^{(2)}, x_2^{(2)}, \ldots, x_n^{(2)})$, SBX produces two children $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$ for each variable (index i) using:

$$\beta_q = \begin{cases} (2u)^{\frac{1}{q_c+1}}, & \text{if } u \leq 0.5, \\ \left(\frac{1}{2(1-u)}\right)^{\frac{1}{q_c+1}}, & \text{if } u > 0.5, \end{cases}$$

where $u \in [0, 1]$ is drawn uniformly at random, and $q_c > 0$ is the distribution index. The value β_q acts as a spread factor:

- If $u \leq 0.5$, then $\beta_q = (2u)^{1/(q_c+1)} < 1$, which biases offspring toward the region between the parents.
- If $u > 0.5$, then $\beta_q = \left[2(1-u)\right]^{-1/(q_c+1)} > 1$, which allows offspring to explore beyond the parents.

A larger q_c concentrates most β_q values near 1, producing children close to their parents, increasing exploitation, while a smaller q_c permits wider variation, increasing exploration.
Once β_q is computed for each variable i, the children are generated as:

$$y_i^{(1)} = \tfrac{1}{2}\Big[(1+\beta_q)\,x_i^{(1)} + (1-\beta_q)\,x_i^{(2)}\Big], \quad y_i^{(2)} = \tfrac{1}{2}\Big[(1-\beta_q)\,x_i^{(1)} + (1+\beta_q)\,x_i^{(2)}\Big].$$

By iterating this for all dimensions i, we obtain two new offspring vectors $\mathbf{y}^{(1)}$ and $\mathbf{y}^{(2)}$. This process balances parental traits with controlled exploration of the search space.

3.9.3 Mutation

Mutation introduces random changes to an individual's genome. This is to maintain diversity in the population and for exploring parts of the solution space that may not be reachable by crossover alone. Typically, each gene is flipped or perturbed with a small probability p_{mut}. For binary-encoded solutions:

$$\text{Genome} = g_1\, g_2\, \dots\, g_n,$$

a simple mutation operator might flip each bit g_i with probability p_{mut}:

$$g_i' = \begin{cases} 1 - g_i, & \text{with probability } p_{\text{mut}}, \\ g_i, & \text{with probability } (1 - p_{\text{mut}}). \end{cases}$$

3.9.4 Selection

Selection is the process of choosing individuals from the current polulation that will undergo crossover and mutation to form the next generation. The goal of selection is to exploit high-fitness individuals by giving them greater reproductive opportunities, and to explore the search space by maintaining diversity. Having a balance between exploitation and exploration prevents premature convergence on suboptimal solutions.

Roulette Wheel (Fitness-Proportionate) Selection. Each individual i is assigned a slice of a roulette wheel proportional to its fitness f_i. The probability of selecting individual i is

$$P_i = \frac{f_i}{\sum_{j=1}^{N} f_j},$$

where N is the population size. A random spin of the wheel determines which individual is selected, so higher-fitness individuals are more likely to be chosen. A drawback of this method is that fitness values must be non-negative; if negative fitnesses occur, scaling or transformation of the fitness function is required.

Tournament Selection. A subset of k individuals is chosen uniformly at random from the population (commonly $k = 2$ or 3). The individual with the highest fitness in this subset is selected to enter the mating pool. This process is repeated N times, with replacement until the mating pool contains the desired number of parents. Tournament size k provides a convenient adjust selection pressure.

Rank Selection. Individuals are first ranked by fitness from best to worst. Selection probabilities are then assigned based on rank rather than raw fitness, for example via a linear or exponential scheme. Rank selection mitigates the risk of a few high-fitness individuals dominating the population too early, preserving diversity and exploration potential.

3.9.5 Elitism

Elitism is a strategy in Genetic Algorithms that preserves a fixed number of the best-performing individuals from one generation to the next. This practice ensures that the highest quality solutions are not lost due to the stochastic nature of crossover and mutation. Formally, let the population at generation t be denoted as $P^{(t)}$ with size N. After evaluating the fitness $f(x)$ for each individual $x \in P^{(t)}$, the top E individuals (where E is a predetermined elitism rate, representing a small percentage of the population size N) are selected as:

$$P^{(t)}_{\text{elite}} = \{x \in P^{(t)} : \text{rank}(x) \leq E\}.$$

These elite individuals are directly copied into the next generation $P^{(t+1)}$ without modification. The remaining $N - E$ slots in $P^{(t+1)}$ are then filled by offspring generated through the standard processes of selection (from $P^{(t)}$), crossover, and mutation.

3.10 Covariance Matrix Adaptation Evolution Strategy

CMA-ES is a stochastic, population-based optimization method designed for real-valued search spaces. It maintains and iteratively updates a multivariate normal distribution over the parameter space, adapting both its mean and covariance [14, 13].

Given an objective function

$$\min_{\mathbf{x} \in \mathbb{R}^n} \quad f(\mathbf{x}),$$

CMA-ES works by sampling multiple candidate solutions from a learned distribution, evaluating their fitnesses, and then updating the distribution parameters to bias future sampling toward regions of higher expected fitness.

3.10.1 Distribution and Sampling

At iteration t, CMA-ES models the parameters $\mathbf{x} \in \mathbb{R}^n$ with a multivariate normal distribution:

$$\mathbf{x} \sim \mathcal{N}\Big(\mathbf{m},\ \sigma^2\mathbf{C}\Big),$$

where

- $\mathbf{m} \in \mathbb{R}^n$ is the mean (center of the distribution),
- $\sigma > 0$ is a global step-size,
- $\mathbf{C} \in \mathbb{R}^{n\times n}$ is the covariance matrix.

To draw a candidate solution $\mathbf{x}_i$, one samples

$$\mathbf{z}_i \ \sim\ \mathcal{N}(\mathbf{0},\mathbf{I}), \quad \text{and sets} \quad \mathbf{x}_i \ =\ \mathbf{m} \ +\ \sigma\,\mathbf{B}\,\mathbf{D}\,\mathbf{z}_i,$$

where $\mathbf{B}$ is an orthonormal matrix of eigenvectors of $\mathbf{C}$, and $\mathbf{D}$ is a diagonal matrix of eigenvalue square roots, so $\mathbf{C} = \mathbf{B}\,\mathbf{D}^2\,\mathbf{B}^\top$. This transformation rotates and scales the standard normal sample $\mathbf{z}_i$ according to the learned covariance structure $\mathbf{C}$, ensuring that $\mathbf{x}_i$ follows $\mathcal{N}(\mathbf{m},\ \sigma^2\mathbf{C})$.

3.10.2 Update Procedure

After sampling a population of λ solutions $\{\mathbf{x}_1, \ldots, \mathbf{x}_\lambda\}$ and evaluating $f(\mathbf{x}_i)$, CMA-ES ranks them by fitness. Let $\mathbf{x}_{(1)}, \ldots, \mathbf{x}_{(\mu)}$ be the top μ solutions, where $\mu \leq \lambda$.

3.10.2.1 Mean Update

After sampling and evaluating the population, the top μ highest-fitness solutions $\Big\{\mathbf{x}_{(1)}, \mathbf{x}_{(2)}, \ldots, \mathbf{x}_{(\mu)}\Big\}$ are selected for recombination. The new mean is defined as a weighted average of these solutions:

$$\mathbf{m}_{\text{new}} \ =\ \sum_{i=1}^{\mu} w_i\,\mathbf{x}_{(i)},$$

where the weights $\{w_i\}$ sum to 1 and typically satisfy $w_1 \geq w_2 \geq \cdots \geq w_\mu$. A useful auxiliary variable is

$$\mathbf{y} \ =\ \frac{\mathbf{m}_{\text{new}} - \mathbf{m}}{\sigma},$$

which measures the update in "standardized" coordinates.

3.10.2.2 Evolution Paths

Here, $c_\sigma \in (0,1]$ and $c_c \in (0,1]$ are the learning rates for the step-size and covariance evolution paths, respectively, and

$$\mu_{\text{eff}} \ =\ \frac{\Big(\sum_{i=1}^{\mu} w_i\Big)^2}{\sum_{i=1}^{\mu} w_i^2}$$

is the variance-effective selection mass.

3.10.2.2.1 Step-size path $\mathbf{p}_\sigma$. This tracks the direction of successful moves in whitened space:

$$\mathbf{p}_\sigma \;\leftarrow\; (1 - c_\sigma)\,\mathbf{p}_\sigma \;+\; \sqrt{c_\sigma\,(2 - c_\sigma)\,\mu_{\text{eff}}}\;\mathbf{C}^{-1/2}\,\mathbf{y}.$$

3.10.2.2.2 Covariance path $\mathbf{p}_c$. This is used to update the covariance matrix:

$$\mathbf{p}_c \;\leftarrow\; (1 - c_c)\,\mathbf{p}_c \;+\; \mathbf{h}_\sigma\,\sqrt{c_c\,(2 - c_c)\,\mu_{\text{eff}}}\;\mathbf{y}.$$

This conditional update via $\mathbf{h}_\sigma$ prevents the path from accumulating spurious steps when the step-size path length deviates significantly from its expected value, which would otherwise corrupt the covariance estimate. Here, $\mathbf{h}_\sigma \in \{0, 1\}$ is set to 0 whenever $\|\mathbf{p}_\sigma\|$ significantly exceeds its expectation, halting $\mathbf{p}_c$ updates in those cases.

3.10.2.3 Step-Size Adaptation

The global step-size σ is adapted by comparing $\|\mathbf{p}_\sigma\|$ to the expected norm of a random vector in $\mathbb{R}^n$. In one common form:

$$\sigma \;\leftarrow\; \sigma\,\exp\!\Bigl(\tfrac{c_\sigma}{d_\sigma}\,\bigl(\tfrac{\|\mathbf{p}_\sigma\|}{E\|\mathbf{z}\|} - 1\bigr)\Bigr),$$

where d_σ is a damping parameter and $E\|\mathbf{z}\|$ is the average length of a standard normal vector in n dimensions.

3.10.2.4 Covariance Matrix Adaptation

Finally, $\mathbf{C}$ is updated using a rank-1 and rank-μ correction:

$$\mathbf{C} \;\leftarrow\; (1 - c_1 - c_\mu)\,\mathbf{C} \;+\; c_1\,\mathbf{p}_c\,\mathbf{p}_c^\top \;+\; c_\mu \sum_{i=1}^{\mu} w_i\,\mathbf{y}_i\,\mathbf{y}_i^\top,$$

where $\mathbf{y}_i = (\mathbf{x}_{(i)} - \mathbf{m})/\sigma$, and $c_1, c_\mu \in (0, 1]$ are the learning rates for the rank-1 and rank-μ updates, respectively. One then performs an eigen-decomposition of $\mathbf{C}$ (updating $\mathbf{B}$ and $\mathbf{D}$) so that future sampling can exploit the current covariance structure.

3.10.3 Separable Covariance Matrix Adaptation

In high-dimensional settings, maintaining a full covariance matrix is expensive, both in memory $\mathcal{O}(n^2)$ and runtime. The separable or diagonal CMA-ES simplifies the model by assuming

$$\mathbf{C} = \operatorname{diag}(c_1, c_2, \ldots, c_n),$$

thus ignoring all off-diagonal elements [24].

3.10.3.1 Sampling (Diagonal Case)

Because $\mathbf{C}$ is diagonal,

$$\mathbf{x}_i \;=\; \mathbf{m} \;+\; \sigma\left(\sqrt{c_1}\,z_{i1},\,\ldots,\,\sqrt{c_n}\,z_{in}\right)^\top, \quad \text{with each } z_{ij} \sim \mathcal{N}(0, 1).$$

3.10.3.2 Diagonal Covariance Update

The update rule affects only diagonal entries,

$$c_j \;\leftarrow\; (1 - c_1 - c_\mu)\, c_j \;+\; c_1\, (\mathbf{p}_{c,j})^2 \;+\; c_\mu \sum_{i=1}^{\mu} w_i\, (\mathbf{y}_{i,j})^2,$$

which greatly reduces memory and computation. However, no cross-correlation between dimensions is learned, making $\mathbf{C}$ restricted to axis-aligned ellipsoids.

3.10.4 Effect on the Search Space

3.10.4.0.1 CMA-ES. The algorithm can rotate and shape the search distribution in any direction (via eigenvectors $\mathbf{B}$), thus effectively handling correlated parameters or skewed/rotated landscapes. The cost is maintaining an $n \times n$ matrix and computing eigen-decompositions each generation [24].

3.10.4.0.2 Separable CMA-ES. By ignoring covariances (off-diagonal terms), the algorithm scales well in $\mathcal{O}(n)$ memory and simpler updates. This is crucial for very large n. However, if the problem exhibits strong parameter correlations, the diagonal approach may converge more slowly, as it cannot capture rotated ridges or valleys in the objective function [24].

3.10.5 Algorithmic Outline

Algorithm 3 provides a high-level CMA-ES pseudocode. Although it shows the full covariance version, the diagonal variant simply replaces the matrix operations with elementwise updates on the diagonal.

Algorithm 3 CMA-ES (Covariance Matrix Adaptation Evolution Strategy)

```
Given: λ (population size), μ < λ, recombination weights {w_i}
Initialize m, σ, C ← I, p_σ ← 0, p_c ← 0
while not terminate do
    for i ← 1 to λ do
        x_i ← sample_multivariate_normal(mean = m, cov = σ^2 C)
        f_i ← f(x_i)
    end for
    sort {x_1, ..., x_λ} by f_i
    m' ← Σ_{i=1}^{μ} w_i x_(i)                          ▷ new mean from best μ
    p_σ ← update_ps(p_σ, (m' − m)/σ)
    p_c ← update_pc(p_c, p_σ, m', m)
    C ← update_C(C, p_c, {x_(i)}, m)
    σ ← update_sigma(σ, ‖p_σ‖)
    m ← m'
end while
return m                                                 ▷ best-known solution
```

4

Objective Function Modeling

4.1 Scenario Graph and Feature Extraction

To evaluate and optimize multi-route coverage strategy solutions, the spatial and geometric properties of the terrain and coverage must be represented in a discretized form. To achieve this, we construct a scenario graph, derived from an original traversability graph and the radar polygons for each node. Each node and edge in the scenario graph contains numerical features, allowing for fast evaluation within the objective function (§4.2).

4.1.1 Coverage Zones and Scenario Graph

We begin by defining three strategically significant zones, represented as polygons on the terrain:

- **Defended Asset (DA)**: A critical region requiring protection.
- **Keep-Out Zone (KOZ)**: A surrounding buffer region around the DA, intended to prevent adversary penetration.
- **Remaining Area (REST)**: All remaining terrain beyond DA and KOZ.

Each node within the traversability graph is associated with a coverage polygon, representing the area observable or coverable from that node. To form the scenario graph, we preserve node coordinates (x, y), maximum travel speed for the vehicle on the terrain (v_{max}), and edge distances from the original traversability graph. We then derive additional scenario-specific features based on geometric intersections between node coverage polygons and the defined zones. These additional features capture how effectively each node contributes to the coverage requirements of the different zones, thus forming the basis for subsequent objective evaluations.

4.1.2 Node Feature Definitions

Each node in the scenario graph carries 14 distinct features. The first three features originate directly from the traversability graph: the normalized coordinates x and y (both ranging from 0 to 1), and v_{max}, representing the local maximum travel speed at the node in km/h.

The remaining eleven features derive from geometric analysis of the node's coverage polygon and its intersections with the three defined strategic zones:

The coverage effectiveness features α_{DA}, α_{KOZ}, and α_{REST} represent the fraction of each respective zone, Defended Asset (DA), Keep-Out Zone (KOZ), and Remaining Area (REST) covered by the node's visibility polygon.

The compactness and utilization of coverage are characterized by σ_{shape}, the Polsby–Popper circularity score measuring shape compactness (ranging from 0 to 1), and ρ_{cov}, the coverage ratio defined as the actual intersection area with the ideal radar coverage circle divided by the area of that circle.

Finally, the positional coverage metrics for each zone include the magnitudes m_{DA}, m_{KOZ}, m_{REST}, representing the normalized offset vectors from each zone's centroid to the centroid of their respective coverage intersections, and the corresponding normalized angles θ_{DA}, θ_{KOZ}, θ_{REST}, each normalized to the interval $[-1, 1]$.

4.1.2.1 Zone Area Coverage

For each node, we compute the proportion of the DA, KOZ, and REST areas intersected by its coverage polygon. Specifically, for each zone polygon Z (DA, KOZ, REST) and coverage polygon C of a node, we define:

$$\alpha_{\text{Zone}} = \frac{\text{Area}(C \cap Z)}{\text{Area}(Z)}.$$

These fractions quantify each node's capability to cover specific strategic zones.

4.1.2.2 Coverage Shape Metrics

To characterize the shape and concentration of each node's coverage polygon, we introduce two metrics:

4.1.2.2.1 Polsby–Popper Circularity (σ_{shape}) Circular coverage polygons minimize the maximum distance from the node to any point within the covered area, promoting uniform visibility and reducing blind spots. Therefore, the use Polsby–Popper circularity is used to quantify how close the polygon C is to an ideal circle:

$$\sigma_{\text{shape}} = \frac{4\,\pi\,\text{Area}(C)}{\text{Perimeter}(C)^2},$$

yielding values in $[0, 1]$, where a higher score indicates more compact, circular coverage shapes.

4.1.2.2.2 Coverage Ratio (ρ_{cov}) To quantify how effectively a node utilizes its theoretical radar capability, we define the coverage ratio as the proportion of the node's actual coverage polygon area relative to the area of an idealized coverage circle representing the radar system's ideal coverage based on the radars range, and also assuming no obstructions. Formally, let C be the node's actual coverage polygon and C_r an ideal circular coverage area with radius r, centered at the radar position of C. The coverage ratio is thus given by:

$$\rho_{\text{cov}} = \frac{\text{Area}(C \cap C_r)}{\text{Area}(C_r)}.$$

A higher ratio indicates that the node's actual coverage closely matches the radar system's maximum potential, whereas a lower ratio reflects underutilization of the radar's ideal coverage capacity.

4.1.2.3 Coverage Placement Metrics (Magnitude and Angle)

To determine the alignment and displacement of coverage relative to each zone's centroid within each strategic zone, we calculate a normalized offset vector between the centroid of the intersection area ($C \cap Z$) and the centroid of the zone Z. This vector provides two metrics:

Let IC denote the centroid of $C \cap Z$ and ZC the centroid of zone Z. We define the offset components as:

$$\Delta x = \mathrm{IC}_x - \mathrm{ZC}_x, \quad \Delta y = \mathrm{IC}_y - \mathrm{ZC}_y.$$

The magnitude is then normalized by half the diagonal length of zone Z's bounding box:

$$m_{\text{Zone}} = \frac{\sqrt{\Delta x^2 + \Delta y^2}}{\frac{1}{2}\,\mathrm{diag}(Z)}.$$

Normalizing by half the diagonal length scales the distance to a standard range of $[0, 1]$, since the maximum distance from the zone centroid to a corner of its bounding box equals half the diagonal, providing a consistent geometric reference across zones of different sizes.

The angle is computed using the standard arctangent function, normalized to lie within $[-1, 1]$:

$$\theta_{\text{Zone}} = \frac{\arctan 2(\Delta y, \Delta x)}{180}.$$

These metrics succinctly represent the directional and positional attributes of coverage within each zone.

4.2 Objective Function

This section presents the objective function designed to evaluate and compare different multi-route solutions in our coverage planning problem. The objective function outputs a single numerical fitness value: higher values indicate better solutions. Conceptually, the function incentivizes (1) effective and complementary coverage of important zones, (2) avoiding redundant coverage at the same time and area, (3) penalizing overly long or unbalanced routes, and (4) discouraging excessively clustered solutions. The fitness is computed in four main parts:

1. Route Arrival Times & Event Generation
2. Synergy, Coverage Quality, and Exploration
3. Time Penalties
4. Cluster Penalties

Finally, the negative of the penalties is subtracted from the total score (the sum of arrival-time rewards, synergy, coverage quality, and exploration terms) to produce the final scalar fitness value.

4.2.1 Route Arrival Times and Timeline Looping

Let $\{\mathrm{R}_1, \ldots, \mathrm{R}_r\}$ be the set of r routes. Each route R_r is an ordered list of nodes $(n_1, n_2, \ldots, n_k)$. For each route:

1. **Travel Times**: We begin by computing an arrival time t_i at each node n_i. We set $t_1 = 0$ (i.e., the arrival time at the first node is zero). Then for every pair (n_i, n_{i+1}), we retrieve the edge distance $d(n_i, n_{i+1})$. Then v_1 and v_2 are the speeds (in km/h) at nodes n_i and n_{i+1}, we define
$$\overline{v} \;=\; \frac{v_1 + v_2}{2}, \quad \tau \;=\; \frac{d(n_i, n_{i+1})}{\overline{v}},$$
and thus
$$t_{i+1} \;=\; t_i + \tau.$$
2. **Route Duration**: The time at the final node of each route is its *route duration*, denoted T^r_{end}. Let
$$T_{\max} \;=\; \max_r \Big(T^r_{\mathrm{end}}\Big)$$
be the maximum duration among all routes.
3. **Looping**: If routes are not the same length, shorter routes are "replayed" to fill the full time horizon $T_{\max}$. Suppose route r has duration T^r_{end}. The number of full loops is
$$\left\lfloor \frac{T_{\max}}{T^r_{\mathrm{end}}} \right\rfloor,$$
with a leftover time window for the partial loop. Each loop repeats the node visits of route r, offset in time by multiples of T^r_{end}. This ensures that the entire time interval $[0, T_{\max}]$ can have repeated coverage from shorter routes.

4.2.2 Synergy, Coverage Quality, and Exploration

4.2.2.1 Coverage Events by Zone

We focus on coverage in three distinct zones: the Defended Asset (DA), the Keep-Out Zone (KOZ), and the Remaining Area (REST). For each route r and node n within that route, we examine three specific node features, the zone area fraction α_Z, coverage placement magnitude m_Z, and coverage placement angle θ_Z, that collectively indicate potential coverage contributions in each respective zone ($Z \in \{\mathrm{DA, KOZ, REST}\}$). If a node has a coverage area fraction and placement magnitude in a given zone (i.e., $\alpha_Z > 0$ and $m_Z > 0$), an event is created at the time $t(n)$ when route r arrives at node n.

4.2.2.2 Quality of Each Coverage Event

The coverage quality of each event integrates three key node attributes: the zone coverage effectiveness, defined as the product $\alpha_Z \times m_Z$; the Polsby–Popper Circularity σ_{shape}; and the coverage ratio ρ_{cov}. These attributes are combined using respective weights w_{shape} and w_{cov}, resulting in a final event quality measure:

$$\text{Quality} = (\alpha_Z \times m_Z) + w_{\text{shape}} \cdot \sigma_{\text{shape}} + w_{\text{cov}} \cdot \rho_{\text{cov}}.$$

All coverage events are then grouped by zone. Let $\mathcal{E}_Z$ denote the set of coverage events occurring in zone Z. In particular, we maintain three event sets:

$$\mathcal{E}_{\text{DA}}, \quad \mathcal{E}_{\text{KOZ}}, \quad \mathcal{E}_{\text{REST}}.$$

4.2.2.3 Synergy: Same vs. Opposite Angles

Once there's a list of coverage events for each zone, we compare events *within the same zone* that happen close in time. Specifically, for two events ev_i and ev_j belonging to different routes $i \text{and} j$, occurring at times t_i and t_j, they are compared if $|\, t_i - t_j \,| \leq \delta_t$. Each event has a coverage placement angle θ in $[-1, 1]$ where we define

$$\Delta\theta = \Big|\theta_i - \theta_j\Big|.$$

We then apply:

1. **Same Angle Penalty**: If $\Delta\theta < \delta_{\text{angle}}$, we subtract:

$$P_{\text{same}} \ \times \ \frac{\text{Quality}_i + \text{Quality}_j}{2}.$$

   This penalizes redundant coverage in the same region within the same zone at nearly the same time.
2. **Opposite Angle Bonus**: If $\Delta\alpha$ is close to 1 (within some tolerance), we add:

$$B_{\text{opp}} \ \times \ \frac{\text{Quality}_i + \text{Quality}_j}{2}.$$

   This rewards coverage coming from opposing angles in the same zone at about the same time.

Summing these pairwise increments (positive or negative) yields a *synergy score* for each zone:

$$S_Z[\text{Zone}] \ \in \ \mathbb{R}.$$

4.2.2.4 Summation of Raw Quality Per Zone

In addition to synergy, each zone accumulates the *raw coverage quality* of all events. If events(Zone) is the set of coverage events for that zone, the total raw quality is simply,

$$Q_Z[\text{Zone}] = \sum_{\text{ev} \in \mathcal{E}_Z)} \text{Quality}(\text{ev}).$$

4.2.2.5 Coverage-Based Exploration

To encourage routes that provide more *diverse* coverage over both *time* and *angle*, the objective function measures an *exploration fraction.* We define:

- A total time horizon of $[0,\, T_{\max}]$. We break it into bins of size 30 minutes (e.g., 12 bins for a 6-hour horizon).
- The angle range $[-1, +1]$ is binned into 8 sub-ranges (9 edges).

Hence, each zone has a 2D grid of $(\text{time\_bin}, \text{angle\_bin})$. We mark a cell "covered" if an event occurs with time and angle falling into that bin. The *exploration fraction* for zone Z is:

$$\texttt{Exploration Fraction per Zone} = \frac{\text{Covered bins}}{\text{Total bins}}.$$

Multiplying by the *exploration weight* w_{exp} gives an *exploration reward* added to the zone's score.

4.2.2.6 Final Zone Scores and Sum

After synergy, raw quality, and exploration, we weight each zone's partial score by a *zone weight*:

$$\begin{aligned}
\text{ZoneScore(DA)} &= w_{\text{DA}}\Big(S_{\text{DA}} + Q_{\text{DA}} + R_{\text{DA}}\Big),\\
\text{ZoneScore(KOZ)} &= w_{\text{KOZ}}\Big(S_{\text{KOZ}} + Q_{\text{KOZ}} + R_{\text{KOZ}}\Big),\\
\text{ZoneScore(REST)} &= w_{\text{REST}}\Big(S_{\text{REST}} + Q_{\text{REST}} + R_{\text{REST}}\Big).
\end{aligned}$$

Summing them yields a *union coverage synergy score*:

$$S_{\cup} \;=\; \text{ZoneScore(DA)} + \text{ZoneScore(KOZ)} + \text{ZoneScore(REST)}\,.$$

4.2.3 Time Penalties

Even with strong coverage scores, we wish to discourage routes that are excessively long or imbalanced. Let T^r_{end} be the duration of route r. Then:

1. **Variance Penalty**: We compute

   $$\mu = \frac{1}{R}\sum_{r=1}^{R} T^r_{\text{end}}, \quad \sigma_T^2 = \frac{1}{R}\sum_{r=1}^{R}\Big(T^r_{\text{end}} - \mu\Big)^2.$$

   The penalty is

   $$P_{\text{var}} = \alpha_{\text{time}} \times \sigma_T^2,$$

   where α_{time} is a tunable coefficient (α_{rtp}). This punishes large discrepancies in route durations.

2. **Exponential Penalty for Excess Duration**: If the longest route takes $\max(T^r_{\text{end}})$ hours and a *threshold* T_{th} is set (e.g., 6 hours), then

$$\Delta T = \max\Big(0,\ \max_r(T^r_{\text{end}}) - T_{\text{th}}\Big).$$

The function then imposes an exponential penalty:

$$P_{\text{dur}} = \beta_{\text{time}}\Big(\exp\Big(\gamma_{\text{time}}\ \times\ \Delta T\Big) - 1\Big),$$

where $\beta_{\text{time}} = \beta_{\text{exp}}$ and $\gamma_{\text{time}} = \gamma_{\text{rate}}$. As a route's length grows beyond T_{th}, the penalty rises exponentially.

We sum these two pieces to obtain:

$$P_T\ =\ P_{\text{var}} + P_{\text{dur}}.$$

4.2.4 Cluster Penalty

As a final check, we penalize routes that remain in a very limited geographic region. For each route r, let $\{(x_i, y_i)\}$ be the normalized 2D coordinates of its nodes. We compute:

1. The *centroid* $(\bar{x}, \bar{y})$ of that route's nodes:

$$\bar{x} = \frac{1}{k}\sum_{i=1}^{k} x_i, \quad \bar{y} = \frac{1}{k}\sum_{i=1}^{k} y_i.$$

2. The *average distance* $\overline{d}$ of each node (x_i, y_i) from the centroid:

$$\overline{d} = \frac{1}{k}\sum_{i=1}^{k}\sqrt{(\,x_i - \bar{x}\,)^2 + (\,y_i - \bar{y}\,)^2}.$$

If $\overline{d}$ is very small, the route is effectively "clustered" in a tight area. We define a per-route penalty:

$$P_r^{\text{clus}} = \frac{P_{\text{clus}}}{\overline{d} + \epsilon},$$

where ϵ is a small constant to avoid division by zero. The sum across all routes,

$$P_{\text{clus}} = \sum_{r=1}^{R} P_r^{\text{clus}},$$

grows large if multiple routes are each confined to a small region.

4.2.5 Final Fitness Function

Bringing all the components together, the final objective function (fitness) is:

$$\boxed{\text{Fitness}\ =\ \underbrace{S_{\cup}}_{\substack{\text{Coverage synergy}\\ +\text{ Raw quality}\\ +\text{ Exploration}}}\ -\ \underbrace{P_T}_{\substack{\text{Variance +}\\ \text{Exponential penalty}}}\ -\ \underbrace{P_{\text{clus}}}_{\substack{\text{Penalty on}\\ \text{tight clustering}}}\ .}$$

Interpretation:

- $S_\cup$ rewards good coverage in the DA, KOZ, and REST zones, synergy bonuses for opposite angles, lower synergy for duplicate angles in the same time window, and broad coverage across time/angle bins (exploration).
- P_T punishes imbalanced or excessively long route durations.
- P_{clus} punishes routes that concentrate their nodes in too small a region, thus failing to spread coverage.

A higher final fitness value indicates a solution that:

1. Covers each zone effectively (maximizing coverage quality).
2. Blances coverage angles over time (rewarding exploration and opposite-angle synergy, penalizing redundant same angles).
3. Keeps route lengths moderate and consistent (to avoid large variance or very long routes).
4. Distributes nodes spatially, avoiding overly tight clustering.

5

Methods

5.0.1 Scenario Graph Generation and Feature Engineering

To create a manageable way of representing the environment for the neural network, a multi-stage preprocessing pipeline was developed. This pipeline transforms raw geospatial data into a scenario graph $G = (V, E)$ where nodes V represent junctions and edges E represent traversable paths, both with simulated features. The objective of this process is to encapsulate essential terrain, visibility, and mobility characteristics that inform the neural network's decision-making and also the fitness evaluation (Section 4.2.5).

5.0.1.1 Terrain Traversability Mapping

The initial of the pipeline handles the derivation of a traversability map from the raw geospatial datasets. High-resolution Digital Surface Model (DSM) data (Section 3.1) and land classification rasters are first aligned and clipped to a common area of interest. Terrain slope, computed from the DSM, serves as a primary indicator of traversability, with areas exceeding a predefined slope threshold deemed impassable to approximate vehicle mobility constraints. This slope-based mask is then refined using a sequence of binary morphological operations as illustrated in the morpohological processing stage (panel 2) of Figure 5.1. Both erosion and dilation are used to keep traversable areas spatially coherent and also removing noisy artifacts from the slope calculation.

Figure 5.1: Traversability map generation pipeline. From left to right: (1) Initial traversability map derived from DSM data using slope gradients and land classification to assess off-road mobility; (2) morphological processing (erosion and dilation) to enclose traversable regions; (3) integration of OpenStreetMap (OSM) road data to ensure accurate representation of on-road paths; (4) final traversability map combining processed terrain and OSM information.

The structuring elements for these operations are chosen to approximate the vehicle's footprint, thereby filtering out passages too narrow for navigation. Semantic constraints from the land cover classification (e.g., water bodies, dense vegetation) are subsequently integrated by logical intersection to further refine the traversable regions. Where available, road network data from OpenStreetMap (OSM, Section 3.3) is incorporated via logical union, ensuring known navigable routes are preserved.

5.0.1.2 Graph Structure Derivation

The final binary traversability mask is then converted into a graph structure suitable for path planning. This involves morphological skeletonization (principles described in Section 3.4) to reduce traversable areas to one-pixel-wide centerlines representing the core connectivity of the environment. The rationale for skeletonization is to significantly reduce the search space for pathfinding algorithms while preserving the topological structure of the traversable terrain.
From this skeleton, a high-level graph G is constructed. Nodes in G are strategically placed at junctions (pixels with more than two neighbors), as these represent critical decision points for navigation. To establish connectivity between these high-level nodes, an intermediate pixel-level graph of the entire skeleton is temporarily formed. Breadth-First Search (BFS) is then employed from each high-level node on this pixel graph to find the shortest paths (in terms of pixel count) to other reachable high-level nodes. These discovered paths become the edges in the final graph G.

5.0.1.3 Node and Edge Feature Engineering

The constructed graph G is subsequently enriched with a set of node and edge attributes critical for the task. This process leverages the original DSM, land classification, and OSM data, alongside specialized geospatial analyses.
For each node $v_i \in V$:

- **Geographical Position and Maximum Speed:** The node's real-world coordinates (geo_pos) are derived from its pixel location using the DSM's geotransformation. Its maximum traversal speed (max_speed) is determined by querying a generated speed layer at the node's location. This speed layer itself integrates terrain class-dependent speed values with explicit speed limits from OSM road data (Section 3.3), providing a mobility term.

- **Radar Viewshed Features:** A critical set of features pertains to visibility. For each node, a viewshed polygon representing the area visible from that node (assuming a radar placement) is computed using GDAL's implementation of a reference plane algorithm (detailed in Section 3.6), considering Earth's curvature, atmospheric refraction, radar height, target height, and maximum range. Due to the potential memory burden of storing many large polygons, viewsheds are stored externally in a shelve database, with nodes retaining only a reference. From these viewsheds and the defined operational zones (DA, KOZ, REST, from Section 4.1), we derive the node features used directly by the fitness function: area coverage fractions (α_{Zone}), coverage shape compactness (σ_{shape}), coverage ratio (ρ_{cov}), and coverage placement magnitudes and angles ($m_{\text{Zone}}, \theta_{\text{Zone}}$), as detailed in the objective function modeling (Section 4.2). These features provide the neural network and fitness function with quantitative measures of each node's potential contribution to surveillance objectives.

For each edge $e_{ij} \in E$:

- **Traversal Time:** Estimated as the edge path length divided by the minimum of the maximum speeds of its incident nodes. This serves as a primary weight for pathfinding.

The entire feature extraction pipeline, including DSM merging, classification integration, speed layer generation, and parallelized viewshed computation for all nodes, produce the final .

5.0.2 Scenario construction

The scenario generation process is centered on a structured representation of three spatial regions: Defence Asset (DA), Keep Out Zone (KOZ) and Rest. The creation pipline begins with the extraction of the spatial bounds from the DSM data, so that the georeference aligns with the graph environment. Within these bounds, a scenario center point is selected, as seen later in 6 the center has been arbitrarily chosen. In most cases a parametrized, rotated rectangle, with configurable size and orientation to support a variety of scenario setups. The KOZ is then defined as a buffer zone surrounding the DA, reflecting the area within which the presence of a threat could compromise the the DA due to line of sight exposure, the adversarys range, or interception limitations. The remaining region is designated as the REST zone. Integrating this with the traversablitity graph and the outputs from the feature extraction pipeline results in the final scenario graph G'. This graph G' is then used in the optimization process. While the current approach uses simple geometric referenced objects for simplicity, the system is designed to accommodate arbitrary polygonal [10] inputs, enabling the support for a wide set of scenarios. However in this report we have chosen to investigate a certain simplified subset of these.

5.1 Model Architecture

The model, Search-on.the-move (SOTM), is designed to perform joint node and edge prediction on the routing graph environment, which is then used to construct multi-routing for multiple systems. The architecture, illustrated in Figure 5.2, contains three main components: a GATv2 embedding backbone (GATStack), a GCN based branch for node level prediction (GCNBranch), and finally an attention based MLP for on edge prediction (EdgeAttentiveMLP). The selection of the different layers and modules is motivated by the need to balance representational power, with the ability to leverage the node and edge features.

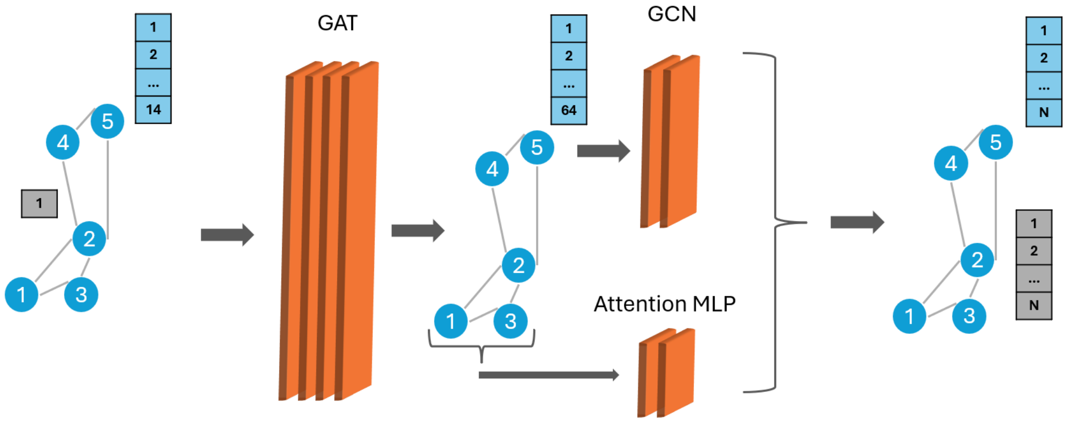


Figure 5.2: Overview of the graph-based model architecture. The input graph, with node features of dimension 14, is first processed through four Graph Attention Network (GAT) layers to project them into a 64-dimensional latent space. The updated graph is then passed through two Graph Convolutional Network (GCN) layers to map node features to an output space of dimension N, corresponding to the number of predefined routes. In parallel, an Attention MLP operates on valid edges (node pairs with connections) to produce N-dimensional edge-level outputs. The final graph consists of node and edge features tailored for downstream route-specific tasks.

The model accepts graph data with node features of dimesion $d_{node_{in}} = 14$ and edge features $d_{edge_{in}} = 1$. It outputs node and edge scores, both with an output dimension matching the number of routes being planned, all tests in this report consider 2 routes.

5.1.1 GATv2 Backbone with Edge Features

To capture local structure and context-dependent relationships in the graph, we adopt a stack of Graph Attention Network v2 (GATv2) layers [3]. This GATStack is configured with 3 layers, where each layer uses 4 attention heads and processes node features into a hidden dimension of 32 per head. Each GATv2 layer computes node embeddings using:

$$\mathbf{h}_i^{(l+1)} = \sigma\left(\sum_{j \in \mathcal{N}(i)} \alpha_{ij}^{(l)} \cdot \mathbf{W}^{(l)}\mathbf{h}_j\right),$$

where α_{ij} is an attention weight that incorporates both $\mathbf{h}_i$, $\mathbf{h}_j$, and the edge feature $\mathbf{e}_{ij}$. A important design choice is the concatenation of outputs from the attention heads. Concatenation of the attention heads produces a separate feature vector which allows the model to learn distinct subspaces of interaction per head meaning that the features in the graph are supplemented by spatial, directional and functional roles of the neighbors in the graph. This leads to the output dimension of $32 \times 4 = 128$ from the first GATv2 layer. Following each GATv2 concolution, an Exponential linear unit (ELU) is applied. A dropout rate of 0 was set, i.e no dropout. The final output of the GATStack, h, is a tensor of node embeddings with dimension 128. Not allowing for concatenation would average away these feature properties.

5.1.2 Node output from GCN Branch

To generate node level scores, the embeddings h are passed through a GCNBranch. This module consists of a sequence of GCNConv layers. In this case it has one layer with 16 units. The activation function used in this branch is the ReLU function. Optionally dropout is applied at the output stage to regularize predictions but all tests ran with dropout set to 0.

$$n_{\text{out},i} = f_{\text{MLP}}\left(\mathbf{h}_i^{(L)}\right), \quad \text{where} \quad \mathbf{h}_i^{(L)} = [\text{GCN}_L \circ \cdots \circ \text{GCN}_1](\mathbf{H}, \mathbf{A})_i \tag{5.1}$$

Here, GCN_ℓ denotes the ℓ^{th} graph convolutional layer, and f_{MLP} is a linear transformation mapping the final embedding to a scalar output $n_{\text{out},i} \in \mathbb{R}$. These outputs are later thresholded during route construction to determine which nodes to include in the route building process (see Section 5.1.5).

5.1.3 EdgeAttentiveMLP for edge prediction

At the same time, the node embedding h (dimension 128) from the GATStack are used to predict edge scores via the EdgeAttentiveMLP module. For each edge (u, v) in the graph, the embeddings h_u and h_v are first stacked. Then the 128 dimensions are linearly projected to an hidden dimension of 32. An multi-head attention mechanism with 4 heads is applied to this two-token sequence. This sequence represents the source and destination node embeddings of a certain edge. Here the projected embeddings serve as query, key and value. The output for each edge is then reduced by mean pooling across the two tokens, resulting in a 32 dimensional vector per edge. These are passed through a MLP, which consists of 2 layers with 32 and 16

neurons, followed by a ReLU. The final layer of the MLP maps the 16 dimensional features to an output dimension of 2. Once again representing the edge score for each route. Additionally No dropout is used.
This mechanism allows the model to reason symmetrically and contextually about edge importance using shared attention over both endpoints. Unlike simpler concatenation schemes, this approach preserves order invariance and allows to model more complex interactions.

5.1.4 Joint Forward Pass

The final SOTM model thus produces node outputs of shape $[N, 2]$ and edge outputs of shape $[E, 2]$, which are then fed to the route building process. Here N and E are the number of nodes and edges in the graph and 2 is the number of routes.

5.1.5 Route building

The node features from the network are then passed to determine the best of these nodes. The main goal is to select a subset of nodes whose importance score is high enough. Instead of picking a fixed number of nodes, say the top K nodes, it tries to find a score threshold such that the number of nodes fall within a range $r = [n_{min}, n_{max}]$. This is threshold is calculated using a binary search algorithm, looking at the range of possible node scores.

The search initializes a lower and upper bound for the potential threshold, which is the minimum and maximum observed scores respectively. First a candidate threshold *mid* is selected as the midpoint of the score bounds. The number of nodes, c, with scores $\geq mid$ are counted. If $n_{min} \leq c \leq n_{max}$, the threshold is accepted and the search stops. However if $c > n_{max}$, the threshold is too low selecting too many nodes. In this case, to find a lower more inclusive threshold the search space is narrowed by setting the lower bound to *mid*. Similarly, if $c < n_{min}$ the threshold is selecting too few nodes, and the upper bound is set to *mid*. This iterates until a *mid* is found that ensures the count c falls within the desired range r.

After this process all the high importance nodes have been determined. Crucially, these nodes are already sorted in descending order based on their scores from the network, see Section 5.1.4. The node with the highest score is selected as the starting point of the route, and the following nodes are visited in order of decreasing importance. Before constructing the route, the graph, previously derived in Section 5.0.1.2, is weighted using the edge predictions from the network output. As mentioned in Section 5.1.3, the edge scores reflect the goodness of transitioning between two nodes.

With the ordered list of must-visit nodes and the weighted graph, the route construction follows. For each pair of nodes in the ordered sequence, an optimal path segment is computed using Dijkstras algorithm, Section , on the full graph, guided by the learned edge cost. Each new path segment is then stitched together to form the full route. While this method of connecting an sequence of nodes with the shortest paths doesnt guarantee global optimlarity for the final route, it's important to keep in mind that the complexity of the routing problem has been significantly reduced by the model's inital node selection. This substantially reduces the search space and provides a qualitatively and computational feasible approximation, as will be shown in Section 6.

5.1.6 Fitness Evaluation

To guide the Covariance Matrix Adaptation Evolution Strategy (CMA-ES) or Genetic Algorithm (GA) in discovering effective routing solutions, a composite fitness function was carefully designed. Its' role is to quantify the overall quality of a candidate set of routes, which are generated by the GNN model and constructed by the route-building algorithm mentioned in the above section. This fitness function translates operational objectives into a single scalar value, effectively framing the optimization problem. The specifics of how this function calculates coverage synergy, raw quality, exploration, and applies penalties for problems like route imbalance or excessive clustering are described in Chapter 4 (Objective Function Modeling, Section 4.2).

For the actual training and experiments presented in Section 6, specific values were after testing chosen for the fitness function's parameters to match operational priorities and constraints. The initial step of evaluating route arrival times and durations, which then forms the basis for generating coverage events, was carried out as outlined in Section 4.2.

The main positive part of the fitness score comes from how well the routes cover the area, work together, and explore. As described in Section 4.2.2, this was set up with particular weights for the different zones: the Defended Asset (DA) was given a zone weight $w_{DA} = 0.05$, the Keep-Out Zone (KOZ) a weight $w_{KOZ} = 1.0$, and the Remaining Area (REST) a weight $w_{REST} = 3.0$. This means that coverage in the REST area was considered three times more important than in the KOZ, pushing for broader surveillance while still ensuring the KOZ was well-protected. The parts that make up the quality of coverage for each event (Section 4.2.2.2), how much of a zone is covered effectively ($\alpha_Z \times m_Z$), how compact the coverage shape is (σ_{shape}), and the coverage ratio (ρ_{cov}) were all given equal importance with weights $w_{shape} = 1.0$ and $w_{cov} = 1.0$.

To encourage routes that provide more diverse coverage over both time and angle, the exploration component (detailed in Section 4.2.2.5) used a total time horizon that was divided into 30-minute bins. The reward for exploring new time-angle combinations was increased by multiplying the fraction of unique bins covered by a factor of 1.5.

Penalties, which are formulated in Sections 4.2.3 and 4.2.4, were put in place to discourage routes with undesirable characteristics. For time penalties, this included a variance penalty, where a weight $\alpha_{rtp} = 1.5$ was applied to the variance of how long different routes took, to encourage all agents to have similar workloads. Also, if routes went on for longer than a 2-hour threshold, an exponential penalty for this excess duration was applied, with a growth rate $\gamma_{rate} = 0.5$ and a scaling factor $\beta_{exp} = 0.2$. To stop routes from clustering too much in one small area, a spatial clustering penalty (Section 4.2.4) was applied, which was proportional to the inverse of each route's mean node-centroid distance, using a cluster penalty factor of 2.0. This helped to make sure patrol efforts were spread out more.

The model was trained on several different scenarios, with different terrains or setups for the DA and KOZ, the fitness scores from each individual scenario were combined using a shifted harmonic mean. Concretely, given n scenario fitnesses $\{f_i\}_{i=1}^{n}$, we first compute a shift

$$s = \max\Big(-\min_i f_i + \varepsilon,\, 0\Big),$$

and define shifted values

$$f_i' = f_i + s.$$

The harmonic mean is then computed as

$$\tilde{H} = \frac{n}{\sum_{i=1}^{n} \frac{1}{f_i' + \varepsilon}} - s,$$

where ε is a small constant (e.g. 10^{-8}) for numerical stability. This formulation retains the harmonic mean's strong penalization of low scenario fitnesses, gracefully handles negative f_i so that if all f_i are sufficiently negative, $\tilde{H}$ remains negative, and avoids a simple clamp to zero thus preserving the correct sign.

This way of combining scores was chosen to help the system learn solutions that would work well across a variety of conditions. This particular setup of the objective function (Section 4.2.5) is what produced the final scalar fitness value used by CMA-ES or GA to adjust the GNN's parameters.

5.1.7 Neural Network model optimization

The parameters of the neural network model were optimized using two distinct evolutionary computation techniques. This approach, often termed neuroevolution (as discussed in Section 2.3.2), was chosen for its efficiency in navigating complex, non-differentiable, and high-dimensional search spaces. Specifically, a Genetic Algorithm (GA) and a variant of the Covariance Matrix Adaptation Evolution Strategy (Section 3.10) (SepCMA-ES) were employed and compared for the primary task of evolving the network weights.(Section 6.1).

5.1.8 Genetic Algorithm training

To optimize the neural network's parameters using a GA they were flattened into a single, real-values one-dimensional tensor, representing an individual in the GA population, Section 3.9. The Objective for the GA is to maximize the fitness score obtained when evaluating candidate solutions, see Section 4.

The GA was configured with a population size of 30 individuals. Elitism3.9.5 was used, preserving the top 20% (i.e., 6 individuals) of the current population directly inserted into the next generation. Tournament selection, with a tournament size of $k = 3$, was used to select parent individuals for reproduction. Crossover between selected parents was performed to form offspring for the next generation. As mentioned in Section 3.9.2.1, SBX was chosen as the crossover operator applied with a crossover probability of 100% and a distribution index $\eta = 15$. Gaussian mutation was then applied where each gene (parameter) in the genome of a offspring had a 10% probability of being perturbed. If selected for mutation, the perturbation would be drawn from a Gaussian distribution $\mathcal{N}(0, \sqrt{0.05})$.

The inital population was generated by taking the networks initial parameters and adding small Gaussian noise, $\mathcal{N}(0, \sqrt{0.01})$. The GA was run for 50 generations. The fitness was determined by unflattering the genome into the SOTM model and performing a forward pass to obtain node and edge scores, which were used to build a route, detailed in Section 5.1.5, and finally evaluated by the objective function.

5.1.9 Separable Covariance Matrix Adaptation Evolution Strategy training

Since the neural network contains $n \approx 80000$ parameters the optimization choice using the CMA-ES method leads to a covariance matrix with size $n \times n$. This gives rise to both memory and computational complexity problems. Therefore this method was adapted to only consider the variance, i.e. the diagonal of the covariance matrix, reducing the amount of operations otherwise required in such a high-dimensional space. We call this the Separable CMA-ES (Sep-CMA-ES). The core challenge addresses by Sep-CMA-ES in this context is the non-differentiable nature of the overall fitness function (Section 4.2.5). The described non-differentiability is a result of the route building process, where discrete node selection via the binary threshold search and using Dijkstra's algorithm (Section 5.1.5). Since Sep-CMA-ES doesn't require the gradient it is suited to handle non-smooth optimization landscapes.

The search of the continuous parameter space samples candidate solutions from evolving multivariate normal distributions that model the parameter space. The distributions mean, variances and step-size are updated based on the fitness of the sampled candidates λ, modeling the local structure of the fitness landscape. The update of the mean between iterations is done via a weighted recombination of the top half performing candidate samples μ. The updated of the step-size σ and the variances of $\mathbf{C}$ is guided by the evolutionary direction $\mathbf{p}_\sigma$ and $\mathbf{p_c}$ respectively, see Section 3.10.2. This is conceptually similar to how momentum and adaptive learning rates function in gradient-based optimization, but without the gradient requirement.

For the Sep-CMA-ES runs, the mean vector $\mathbf{m}$ of the search distribution was initialized using the network's initial parameters, the step-size σ was set to 0.1, and the variances of $\mathbf{C}$ were all set to 1.0. The population size λ of 50 candidate solutions was generated in each generation. From each generation, $\mu = \lambda/2 = 25$ parents were selected using the standard CMA-ES weighting schemes, described in Section 3.10.2.2. The learning rates for the evolutionary paths and the updates to the step-size and diagonal covariance matrix elements were determined using the formulas detailed in [14], which depend on the problem dimensionality n and the effective number of parents μ_{eff}. The optimization was then run for 50 generations. After deeming the Sep-CMA-ES to be more effective than the GA, the final model that was evaluated in Chapter 6 was trained solely using Sep-CMA-ES for 100 generations, using the same parameters as described above.

6

Results

6.1 Training Process Evaluation

Here we present the training outcomes of two multi-route training scenarios for two model variations, the *Compact* model (83,108 parameters) and the *Extended* model (90,916 parameters), each optimized using a Genetic Algorithm (GA) and the Separable Covariance Matrix Adaptation Evolution Strategy (SepCMA-ES). Both training scenarios (see 4.1 for details on scenarios) were run for 50 generations with a population size of 20, as detailed in 5.

Figure 6.1 shows the evolution of average fitness (blue dots) for both GA and SepCMA-ES on the Extended (top row) and Compact (bottom row) models. In the SepCMA-ES plots (right column), the best fitness per generation (orange dots) and the changing σ value (background gradient) are highlighted. Table 6.1 summarizes the total training time, final global best fitness, and the time at which it was reached.

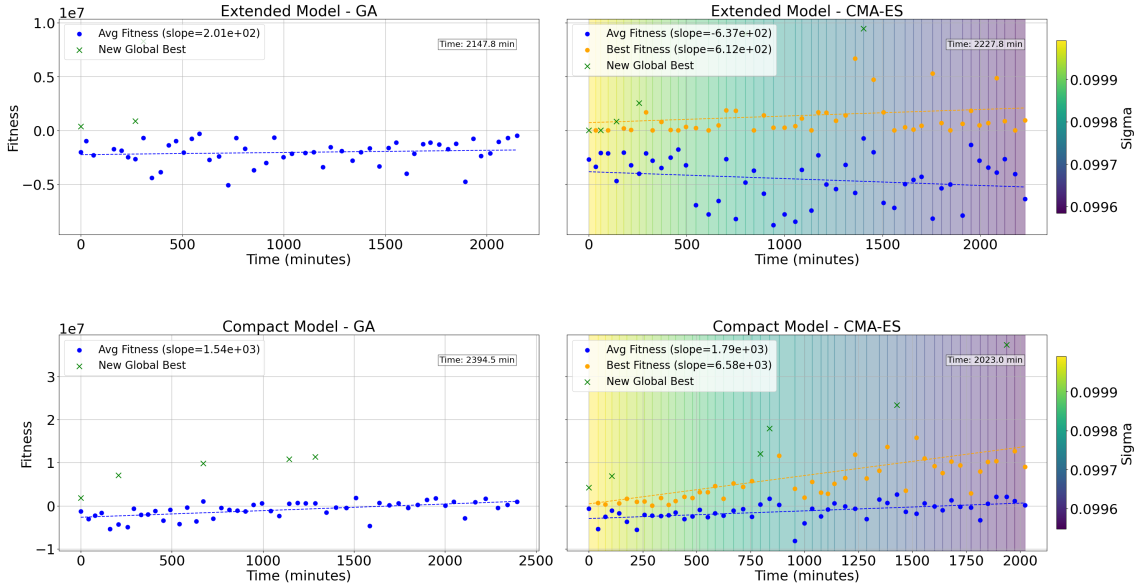


Figure 6.1: Training performance for the *Extended* (top row) and *Compact* (bottom row) GNN models using GA (left) and SepCMA-ES (right). Blue dots indicate average fitness, green crosses mark new global bests, and orange dots (SepCMA-ES) show best fitness values. Dashed lines represent linear regression trendlines. For SepCMA-ES, the background color shifts from yellow to purple, decreasing variance of the search distribution.

Table 6.1: Training statistics for GA and SepCMA-ES on the Extended and Compact models. Shown are the total training time, final global best fitness, and the time to reach that best. Experiments were run on a Dell system with an Intel Core i9 @ 2.40 GHz (Comet Lake), 64 GB DDR4 RAM, and an NVIDIA Quadro RTX 5000 Max-Q GPU.

Model	**Total Time (min)**	**Final Global Best**	**Time to Final Best (min)**
GA-Extended	2147.75	8.27×10^6	307.87
SepCMA-Extended	2227.77	9.45×10^6	1402.90
GA-Compact	2394.47	1.14×10^7	1287.55
SepCMA-Compact	2023.02	3.74×10^7	1938.05

Table 6.2 further indicates that SepCMA-ES surpassed GA's global best fitness after about 140 minutes for the Extended model, while for the Compact model, SepCMA-ES outperformed GA from the start.

Table 6.2: Time (in minutes) until SepCMA-ES exceeds GA's global best fitness for each model.

Model	**Time (min)**
Extended	140.13
Compact	0.00

The results using GA on the Extended model (top-left subplot in Fig. 6.1), indicate global bests were found in the first few hundred minutes, after which improvements mostly stagnated (slope: $+2.01 \times 10^2$ fitness units per minute). In contrast, on the Compact model (bottom-left), GA achieved a steeper improvement rate (slope: $+1.54 \times 10^3$ fitness units per minute) and a higher final fitness (1.14×10^7) compared to the Extended model's 8.27×10^6, showing that GA performed better on the smaller Compact model. This indicates that GA got better solutions more quickly before plateauing.

For SepCMA-ES, the Extended model (top-right) shows a slight decline in the average fitness slope (-6.37×10^2 fitness units per minute) while the best fitness steadily improves ($+6.12 \times 10^2$ fitness units per minute), yielding a final fitness of 9.44×10^6 that outperforms GA-Extended. The adaptive σ parameter (visualized by a gradient from yellow to purple) illustrates the algorithm's progressive local refinement.

On the Compact model (bottom-right), SepCMA-ES achieved the highest overall performance with a final fitness of 3.74×10^7 and a steep best-fitness slope ($+6.58 \times 10^3$ fitness units per minute), reflecting early exploration followed by focused refinement. The steadily decreasing σ confirms effective exploitation of the Compact model's capacity. Overall, while the GA method discovers solutions earlier, SepCMA-ES continues to improve over time, eventually surpassing GA.

6.2 Scalability & Inference Efficiency

To assess how the extended model (trained via SepCMA-ES) scales with increasing graph size, we conducted a series of experiments on progressively smaller versions of our largest scenario graph. Specifically, we began with a graph of 48 493 nodes and iteratively reduced it in steps of 1 000 randomly selected nodes, each time measuring the time required to perform inference and the time required to construct a route from those outputs (see Figure 6.2).

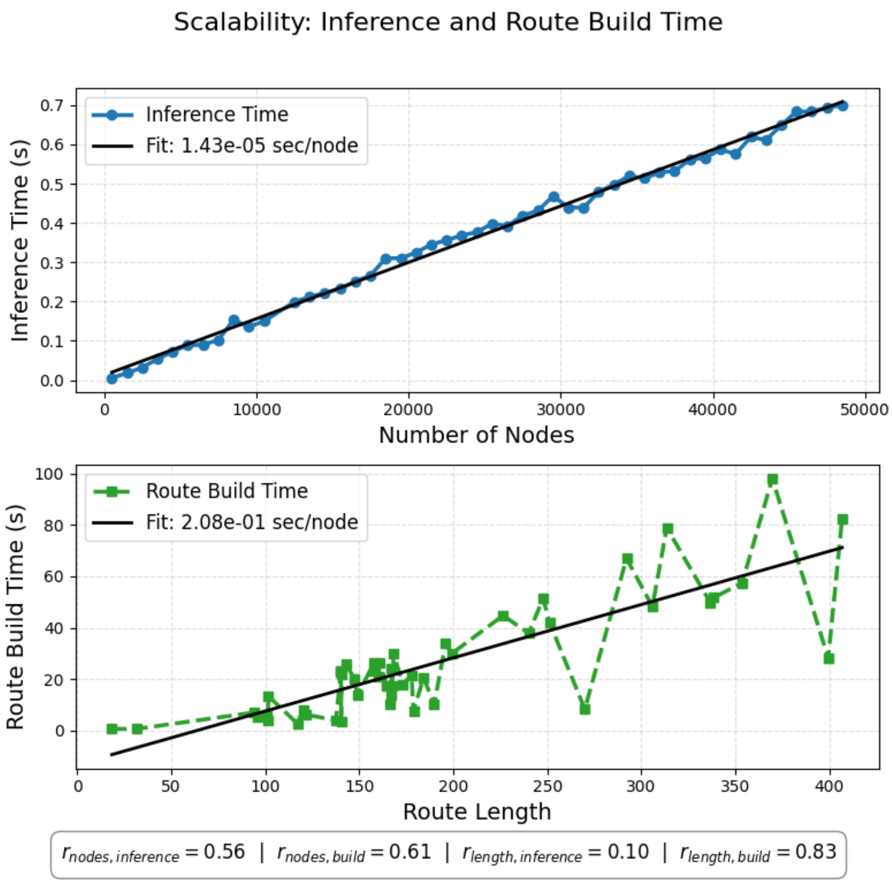


Figure 6.2: Scaling of inference time and route build time with respect to node count and route length.

The inference time exhibits an approximately linear trend across all graph sizes, though the correlation is moderate ($r = 0.56$), indicating that factors beyond node count—such as edge density and implementation overhead—also play a role. In our GNN architecture (see Section 5.1), each Graph Convolutional Network (GCN) layer incurs a theoretical cost of $\mathcal{O}(L\, m\, F + L\, n\, F^2)$, where n is the number of nodes, m the number of edges, F the feature dimension, and L the number of layers [4]. Because F and L are fixed in our experiments and m grows roughly linearly with n, the message-passing term $\mathcal{O}(L\, m\, F)$ dominates, yielding the observed near-linear scaling. Graph Attention Network (GAT) layers similarly cost $\mathcal{O}(|V|\, F\, F' + |E|\, F')$ per attention head, scaling linearly with the number of heads and with edge count [27].

In contrast, the routing step involves a binary search procedure to determine a dynamic threshold, which in one instance operates on the order of $\mathcal{O}(n \log n)$ due to the need to sort or compare all node scores (see 5). Furthermore, the route construction requires repeated shortest-path computations for connecting selected nodes, where each such computation uses a standard Dijkstra-like algorithm on a graph with n nodes and E edges—scales on the order of $\mathcal{O}(n \log n + E)$. Analysis reveals a positive linear trend with the node count ($r = 0.61$) and an even stronger correlation with route length ($r = 0.83$). A linear trend line fitted to the routing time indicates a growth rate of 2.08×10^{-1} seconds per node in the route.

6.3 Training Scenario Solutions

The model was jointly trained across three distinct terrains: Salisbury Plains (UK), an urban-coastal region of Los Angeles (US), and the wooded tundra landscape of the Vidsel Test Range (SE). Each terrain was discretized into traversability graphs containing approximately 7,000 nodes (Los Angeles), 17,000 nodes (Salisbury Plains), and 48,000 nodes (Vidsel). Distinct polygon zones were defined for Defended Asset (DA), Keep-Out Zone (KOZ), and REST in each scenario (see Table 6.3 for detailed polygon area statistics).

Table 6.3: Terrain and scenario statistics

Terrain	Nodes	Edges	Map area (km^2)	DA area (km^2)	KOZ area (km^2)	REST area (km^2)
Los Angeles	8 457	23 655	78.038	0.005	13.146	64.887
Salisbury Plains	17 097	55 415	99.813	0.005	29.129	70.679
Vidsel Test Range	48 493	122 560	125.485	0.005	29.129	96.351

The model was optimized to reward solutions that achieved simultaneous coverage of complementary angles within a 30-minute threshold, while penalizing redundant angles Node quality scores combined shape and spatial coverage equally weights 1.0 each. Coverage in the REST zone was weighted highest (3.0), followed by KOZ (1.0), with minimal emphasis placed on the DA (0.05). Exploration across zones was encouraged by multiplying the fraction of the zone's spatiotemporal coverage by 1.5.

Additionally, route durations were constrained by penalizing both duration variance (weighted by 1.5) and excessive duration beyond 2 hours, with an exponential penalty (growth rate 0.5, scaled by 0.2). To avoid overly clustered routes, spatial clustering was penalized proportionally to the inverse of each route's mean node-centroid distance (cluster penalty factor of 2.0).

These criteria were combined into a single scenario fitness score and then aggregated via a shifted harmonic mean across scenarios. Training employed the compact model variant optimized using SepCMA-ES, consistently achieving superior convergence rates and fitness results, as detailed in Section 6.1.

6.3.1 Salisbury Plains

Figure 6.3 presents a sequence of time–stamps of a *two–vehicle* solution predicted by the trained SOTM model. The green and orange routes cooperate to protect the Defended Asset (DA) by patrolling its encompassing Keep-Out Zone (KOZ) while surveilling the outer REST area. At each instant the vehicles maintain complementary sensor footprints, and jointly seal potential threat corridors into the Keep-Out Zone.

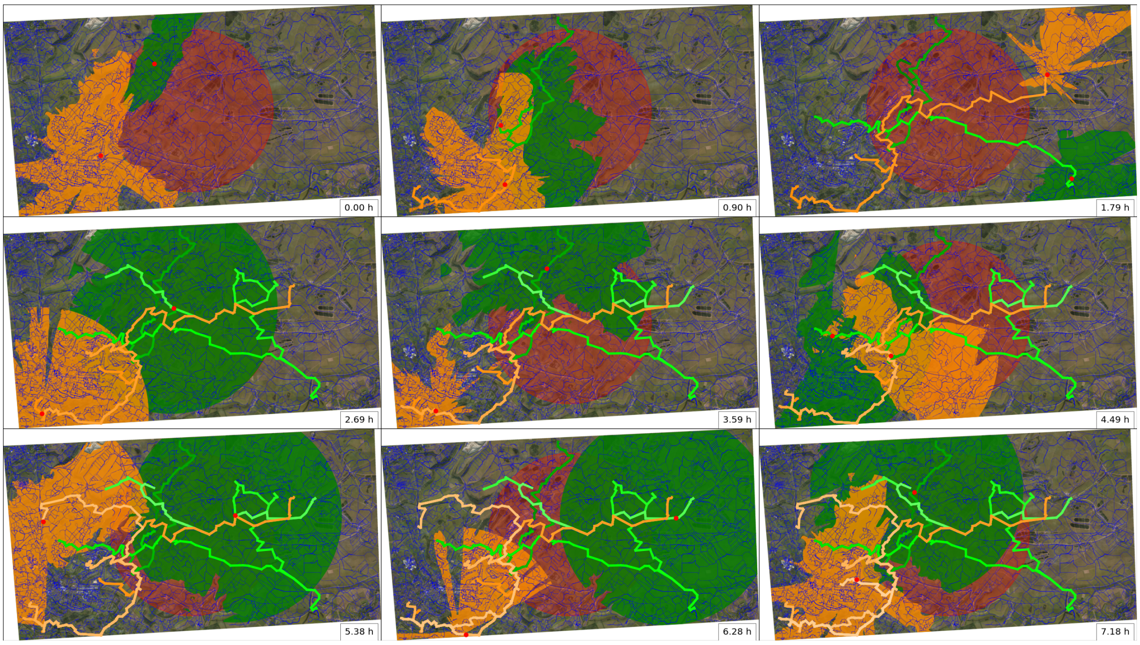


Figure 6.3: Chronological snapshots of the multi-route patrol (t=$\{0, 0.9, 1.8, \ldots, 7.18\}$ h). Underlying map: Salisbury Plains. The blue graph structure overlaid on the terrain is the traversability graph. Polylines are the planned paths (color-gradient by timestep); the red dots indicates the current vehicle positions. Shaded discs correspond to current radar footprints and the solid red circle marks the KOZ boundary.

The cumulative radar heatmap over the full 7.18 h mission is depicted in Fig. 6.4. Coverage shows intensity in the central–south-west quadrant, coinciding with low-lying urban terrain the south-east, whereas elevated terrain to the north-west and far east exhibit lower hit counts.

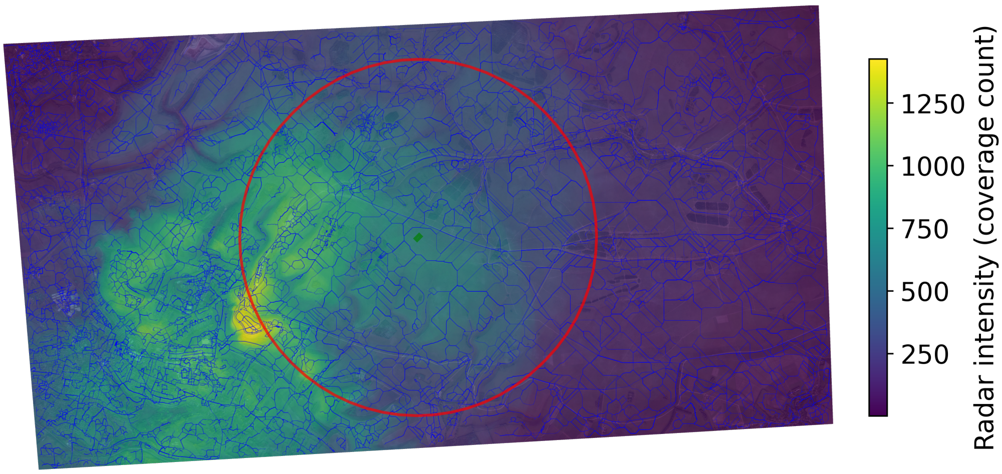


Figure 6.4: Heatmap of the density of coverage events in Salisbury Plains. Highest intensity occurs above the city in the south-west; reduced returns align with high-ground to the north-west and south.

6.3.2 Surveillance & Synchronisation Analysis

Figure 6.5 presents the instantaneous fractional radar coverage $f(t)$ for each of the three zones across the mission duration. The Defended Asset (DA), having a relatively small area, achieves consistently high coverage with a mean of $\mu_{DA} = 82\%$ despite noticeable fluctuations due to rapid revisits. The Keep-Out Zone (KOZ) maintains a stable and substantial coverage at $\mu_{KOZ} = 63\%$. The largest zone, REST, achieves the lowest instantaneous coverage, averaging $\mu_{REST} = 31\%$.

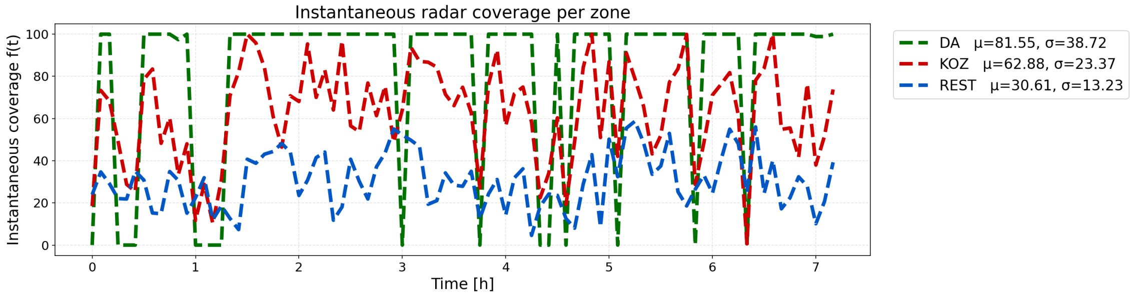


Figure 6.5: Time-series of instantaneous radar coverage fraction $f(t)$ for the Defended Asset (DA, green dashed), Keep-Out Zone (KOZ, red dashed) and REST area (blue dash-dot) over the 7.2 h mission. DA achieves a high mean coverage of $\mu_{DA} = 82\%$ despite rapid revisit fluctuations; KOZ maintains $\mu_{KOZ} = 63\%$; REST averages $\mu_{REST} = 31\%$.

The cumulative coverage results, shown in Figure 6.6, reinforce the overall surveillance time to cover the whole terrain. The DA achieves complete coverage immediately at mission start. The KOZ is completely surveyed within approximately 30 minutes. The REST area reaches full coverage within two hours.

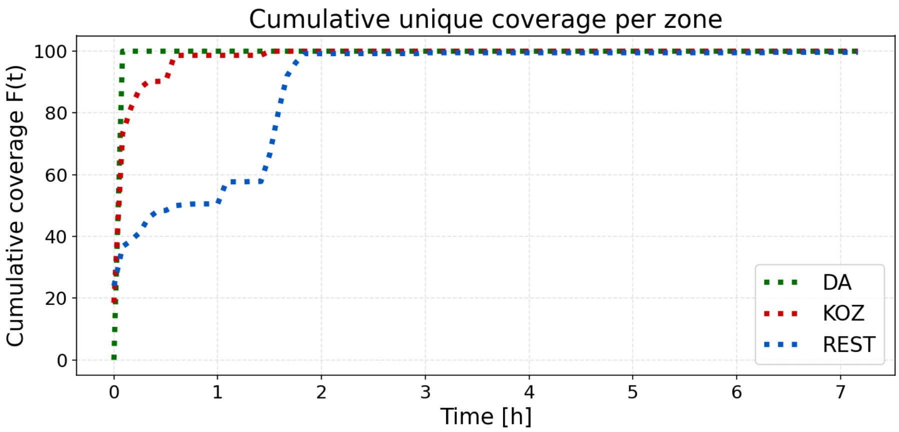


Figure 6.6: Cumulative unique radar coverage $F(t)$ per zone, showing the fraction of each area observed at least once by time t. DA (green) reaches full coverage at $t = 0$; KOZ (red) by $t \approx 0.5$ h; REST (blue) by $t \approx 2$ h.

Sector-based coverage metrics provide a understanding of coverage continuity and directional surveillance. Figure 6.7 depicts the sector-wise analysis for the DA zone.

The DA's limited area results in frequent revisits across all bearings with maximum coverage gaps do not exceed 20 minutes and typically around 5-8 minutes.

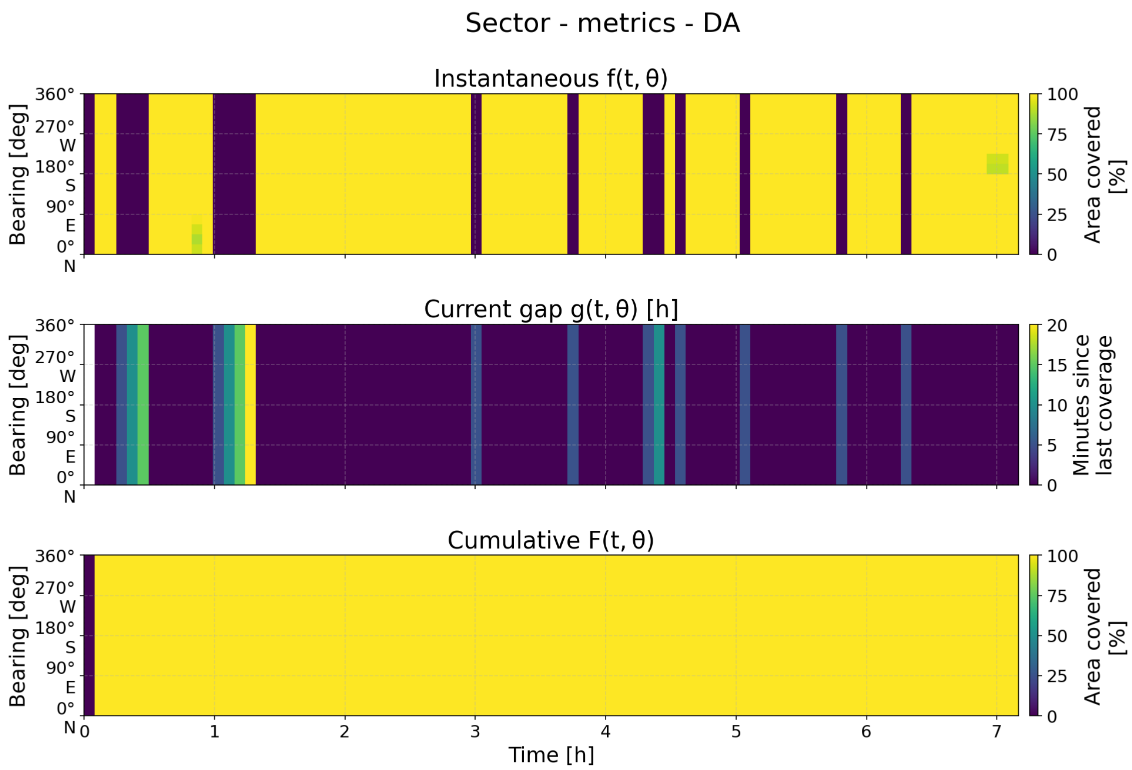


Figure 6.7: Sector-wise metrics for the DA zone. Top: instantaneous coverage $f(t, \theta)$ by bearing θ; middle: current revisit gap $g(t, \theta)$ in minutes since last coverage; bottom: cumulative coverage $F(t, \theta)$. Bearings are shown from 0° (N) to 360° clockwise. Short gaps (typically 5–8 min, never exceeding 20 min) reflect the DA's small area and frequent revisits.

Similarly, the KOZ sector-wise metrics shown in Figure 6.8 exhibit continuous coverage, with few directional outages. Brief gaps of approximately 15-20 minutes occur primarily toward the northern sectors around 1.5 hours into the mission. Cumulative coverage achieves completeness across all bearings by approximately 1.5 hours.

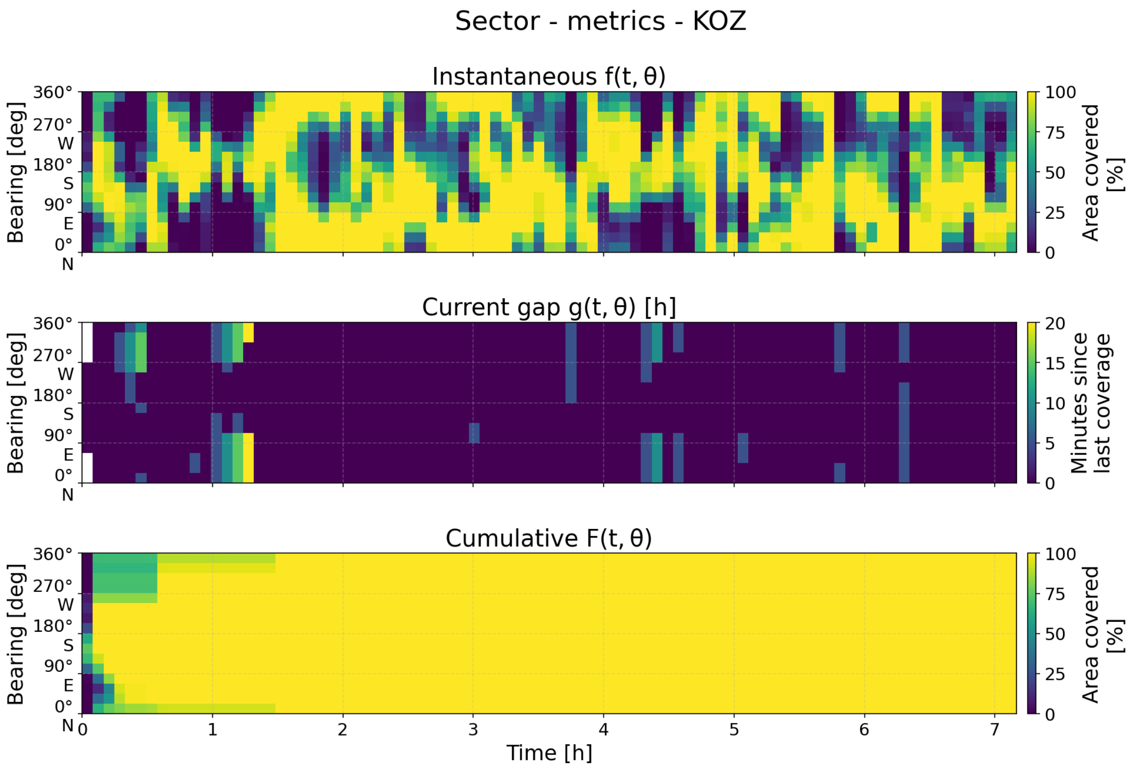


Figure 6.8: Sector-wise metrics for the KOZ. Top: instantaneous coverage $f(t, \theta)$; middle: current gap $g(t, \theta)$ in minutes; bottom: cumulative coverage $F(t, \theta)$. Coverage is largely continuous with isolated gaps up to 15–20 min in northern sectors around $t \approx 1.5$ h, achieving full angular coverage by that time.

The REST area (Figure 6.9), being the largest and most complex to survey, shows distinctive traveling-wave patterns from coordinated vehicle movement through the terrain over time. The largest observed gap occurs around 4.8 hours into the mission, spanning approximately 75 minutes in the northern sector. Despite its size, cumulative coverage is nearly complete, with only a small angular portion (around 10%) in the southern sectors left partially uncovered.

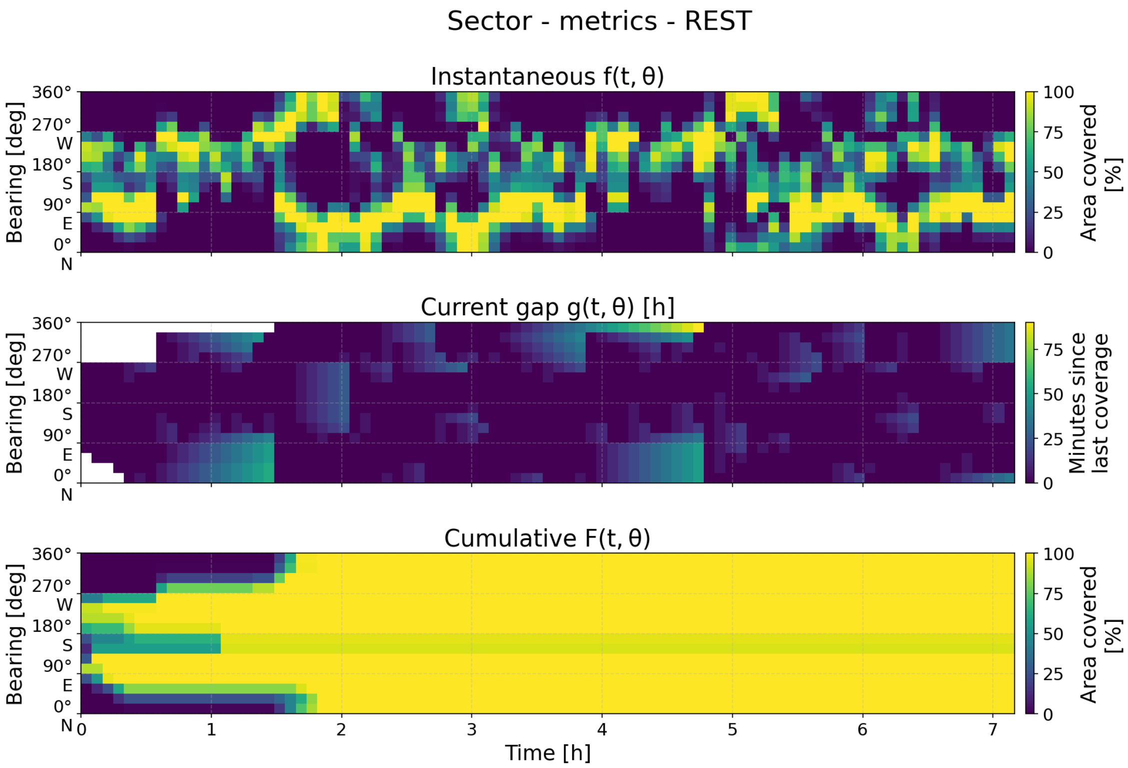


Figure 6.9: Sector-wise metrics for the REST area. Top: instantaneous coverage $f(t, \theta)$; middle: current gap $g(t, \theta)$ in minutes; bottom: cumulative coverage $F(t, \theta)$. Traveling-wave patterns reflect coordinated routing; the maximum gap (75 min) occurs near $t \approx 4.8$ h in the northern sector, while 10% of the southern sectors remain uncovered until mission end.

Results for the two additional training environments, **Los Angeles** and the **Vidsel Test Range**, are detailed in Appendix A. While exhibiting similar performance, the Los Angeles scenario experiences deeper coverage gaps due to mountainous terrain, prompting the model to strategically favor sea-ward patrols to maintain KOZ security.

6.4 Model Generalization

The trained policy was evaluated on two spatially shifted scenario variants: one on Salisbury Plains and one on the Vidsel Test Range. These validation cases assess the model's ability to generalize to unseen scenarios with different DA, KOZ, and REST configurations. The characteristics of the validation scenarios, including node counts and zone areas, are summarized in Table 6.4.

Table 6.4: Validation scenario statistics

Terrain	Nodes	Edges	Map area (km^2)	DA area (km^2)	KOZ area (km^2)	REST area (km^2)
Salisbury Plains	17 097	55 415	99.813	0.005	13.146	86.662
Vidsel Test Range	48 493	122 560	125.485	0.005	15.841	109.640

6.4.1 Salisbury Plains

In this validation scenario, the Defended Asset (DA) and Keep-Out Zone (KOZ) have been shifted to the southeast region of the Salisbury Plains environment. Notably, the solutions initial vehicle positions are located centrally within the terrain, far from the new KOZ position. Figure 6.10 demonstrates that the model adapts the multi-route strategy by redirecting both vehicles towards the southeast, explicitly prioritizing the new KOZ and surrounding area. Although the routes emphasize KOZ protection, they still manage to ensure adequate coverage of the remaining terrain.

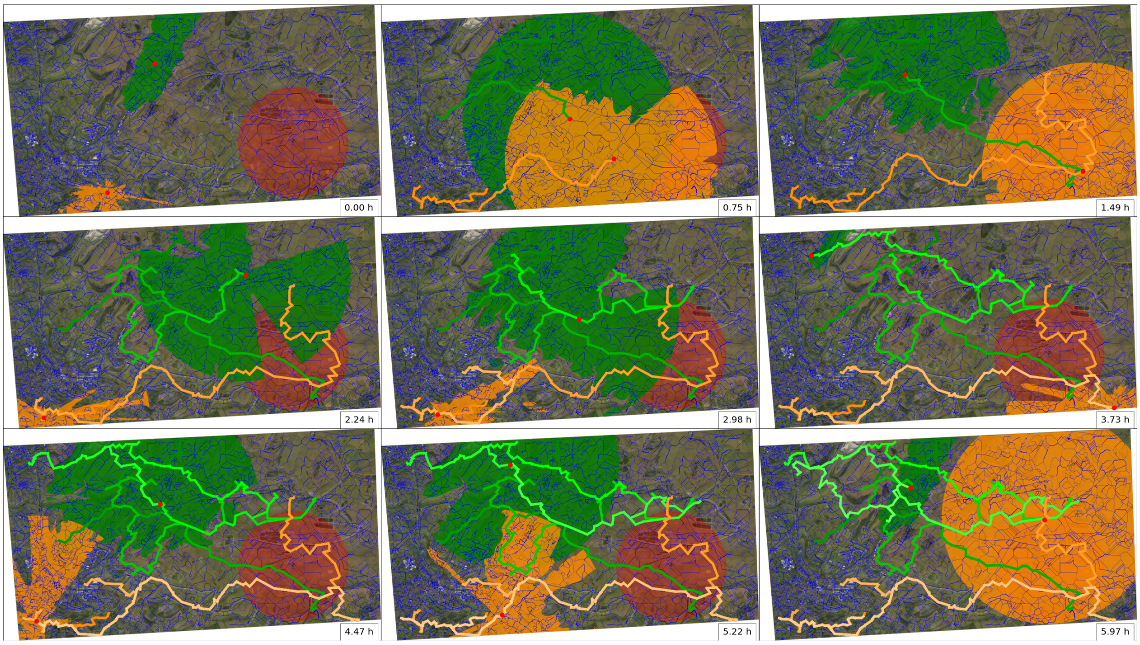


Figure 6.10: Chronological snapshots of the multi-route patrol (t=$\{0, 0.75, 1.49, \ldots, 5.97\}$ h). Underlying map: Salisbury Plains. The blue graph structure overlaid on the terrain is the traversability graph. Polylines are the planned paths (color-gradient by timestep); the red dots indicates the current vehicle positions. Shaded discs correspond to current radar footprints and the solid red circle marks the KOZ boundary.

The radar coverage heatmap (Figure 6.11) further highlights the adaptive shift in surveillance intensity towards the KOZ located in the southeast. Compared to the training scenario (Figure 6.4), the highest radar-hit intensity is reduced from approximately 1300 to about 800, indicative of a more uniformly distributed surveillance pattern.

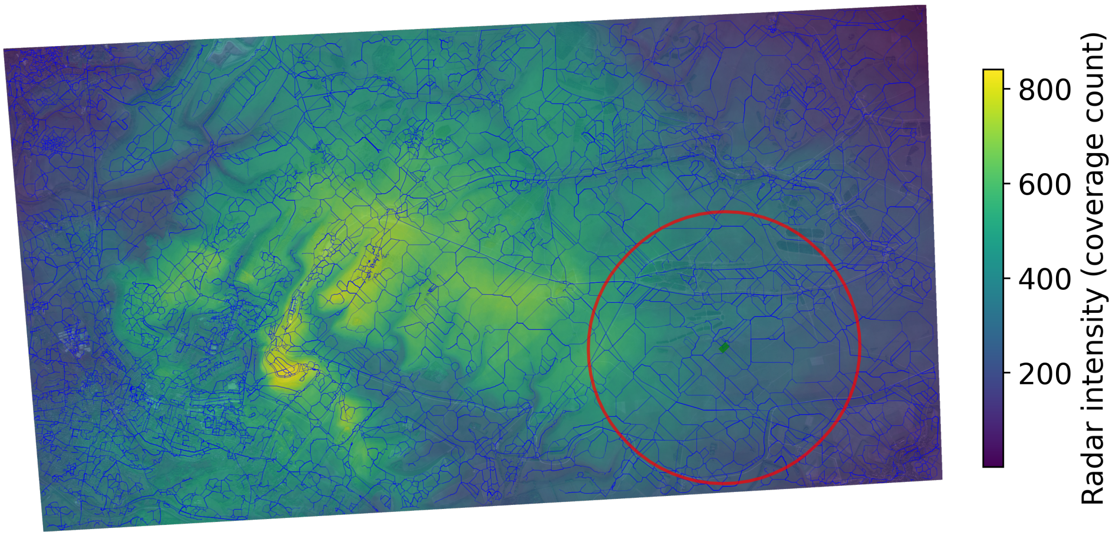


Figure 6.11: Heatmap of the density of coverage events in the validation (Salisbury Plains) scenario. Warmer colors indicate higher coverage counts per map cell over the mission duration. Surveillance effort is redistributed southeastward toward the KOZ and DA zones relative to the training scenario.

6.4.2 Surveillance & Synchronization Analysis

The instantaneous coverage (Figure 6.12) indicates that the REST area, now relatively larger due to the smaller KOZ, achieves a higher mean coverage of $\mu_{REST} = 38\%$. However, there are increased coverage gaps for the KOZ, reflecting the challenge posed by its placement and smaller relative size.

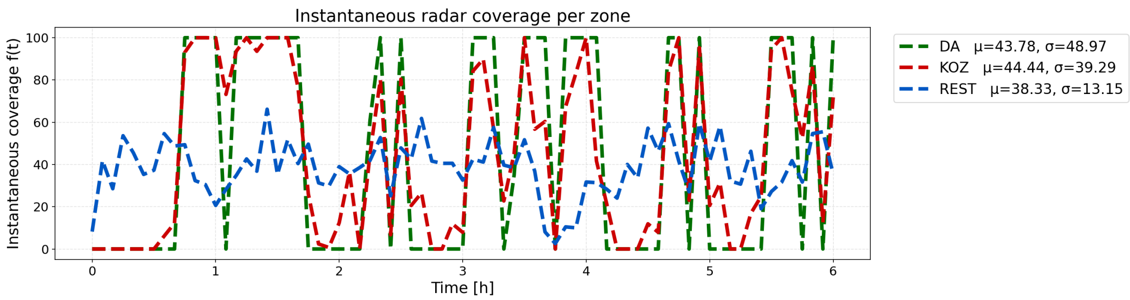


Figure 6.12: Instantaneous radar coverage fraction $f(t)$ in the generalization scenario for DA (green dashed), KOZ (red dashed), and REST (blue dash-dot). REST achieves a higher mean coverage $\mu_{REST} = 38\%$, while KOZ exhibits intermittent coverage gaps due to its smaller size and corner placement.

The cumulative coverage plot (Figure 6.13) confirms complete terrain coverage approximately one hour into the mission, demonstrating adaptation to the shifted scenario.

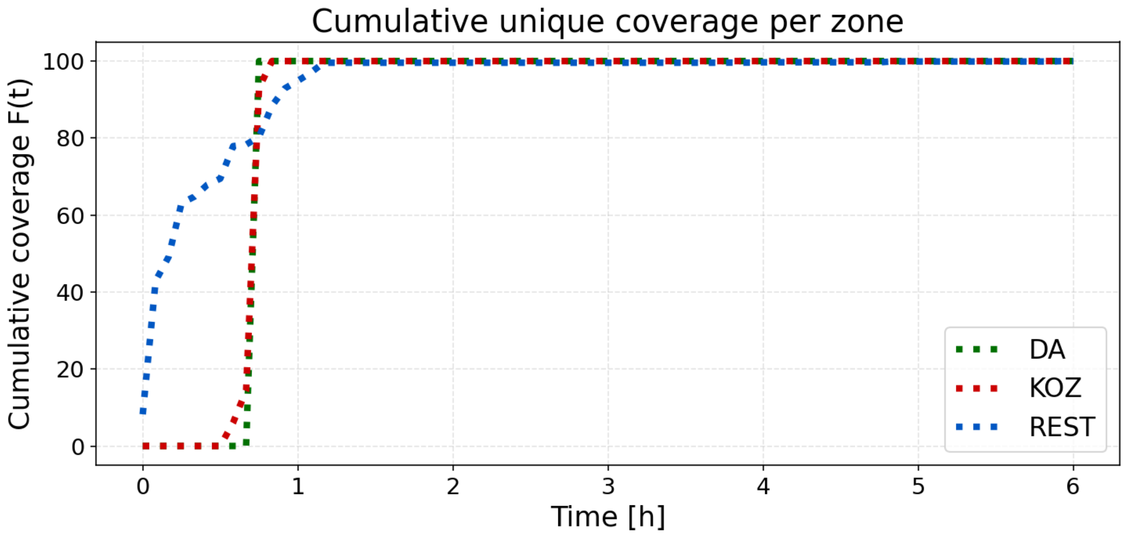


Figure 6.13: Cumulative unique radar coverage $F(t)$ for the generalization scenario. Full terrain coverage is attained by $t \approx 1$ h, demonstrating adaptation of the multi-route strategy to the shifted operational scenario.

The sector analysis of the DA (Figure 6.14) reveals initial coverage gaps due to distant route initialization, notably filled around 50 minutes into the mission. Subsequent gaps occur intermittently, reaching up to 30 minutes around 2, 3, 4.5, and 5.5 hours into the patrol.

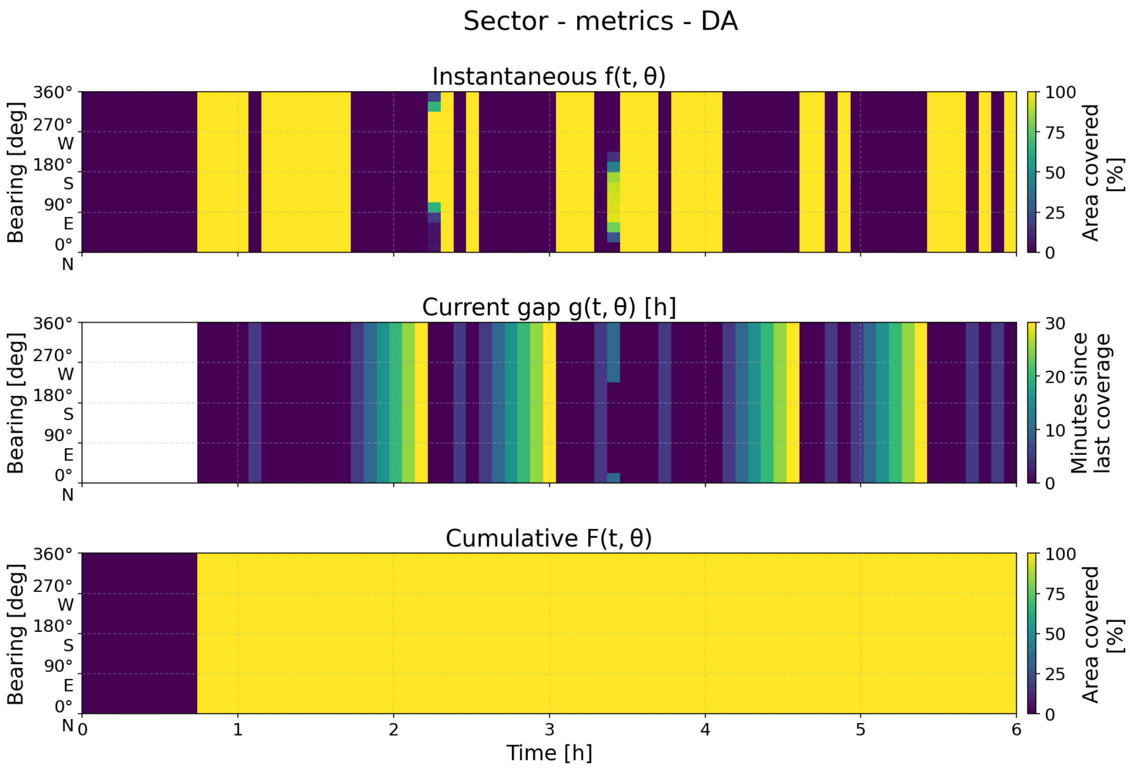


Figure 6.14: DA sector-wise metrics in the generalization scenario. Top: instantaneous coverage $f(t, \theta)$; middle: current revisit gap $g(t, \theta)$ in minutes; bottom: cumulative coverage $F(t, \theta)$. Initial gaps (filled by $t \approx 0.83$ h) and intermittent gaps up to ~ 30 min (around 2, 3, 4.5 and 5.5 h) reflect the increased route initialization distance.

KOZ sector metrics (Figure 6.15) also indicate similar temporal gaps primarily in northern and western sectors, matching the timing patterns of the DA gaps but occurring less extensively.

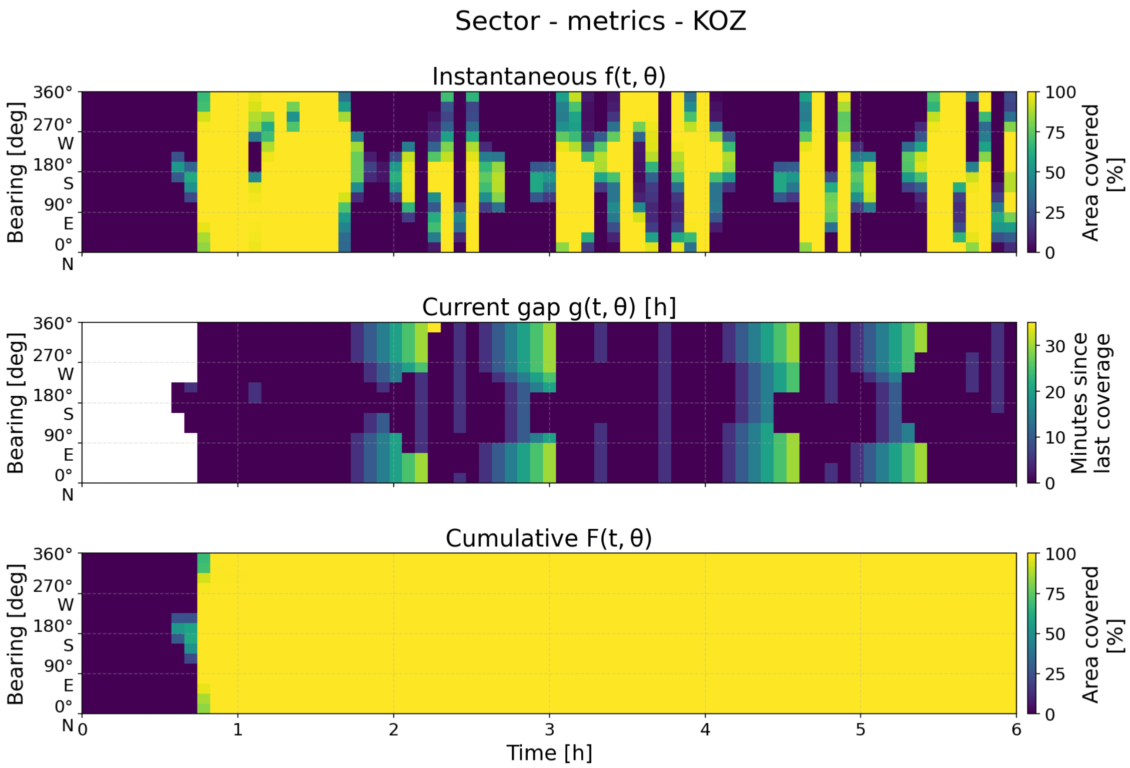


Figure 6.15: KOZ sector-wise metrics in the generalization scenario. Top: instantaneous coverage $f(t, \theta)$; middle: current gap $g(t, \theta)$ in minutes; bottom: cumulative coverage $F(t, \theta)$. Periodic northern and western outages mirror DA timing but are less pronounced, with full angular coverage restored shortly after each gap.

Finally, the REST area (Figure 6.16) maintains comprehensive and well-distributed coverage, with minimal gaps around one and four hours into the mission, each lasting roughly 20 minutes. The distinctive traveling spiral pattern in instantaneous coverage clearly illustrates the effective and coordinated spatial exploration strategy adopted by the vehicles.

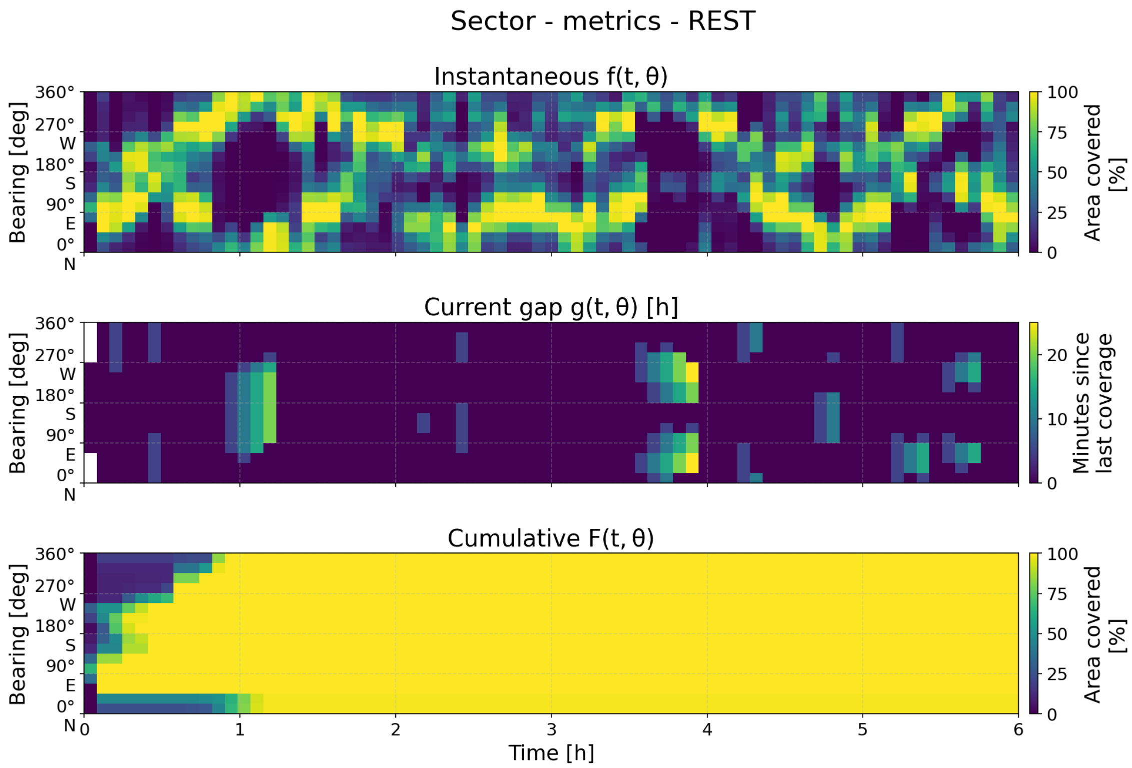


Figure 6.16: REST sector-wise metrics in the generalization scenario. Top: instantaneous coverage $f(t, \theta)$; middle: current gap $g(t, \theta)$ in minutes; bottom: cumulative coverage $F(t, \theta)$. Coverage is comprehensive, with brief 20 min gaps at $t \approx 1$ h and 4 h, and a clear spiral traversal pattern illustrating coordinated multi-route exploration.

The observed gaps in DA and KOZ coverage align logically with their corner placement and the initial route distance, as well as the REST zone's higher weighting (3:1 REST:KOZ) in the objective function. Nevertheless, the graph-based model demonstrates some generalization abilities, effectively realigning the routes toward previously unvisited regions of interest. Additional validation results form Vidsel can be found in Appendix A.

6.5 Interpreting Network Attention

Figure 6.17 visualizes the attention weights from each of the four attention heads Graph Attention Network (GAT) layers for the Salisbury Plains scenario. Each attention head highlights degrees of node relevance, expressed as normalized attention weights.

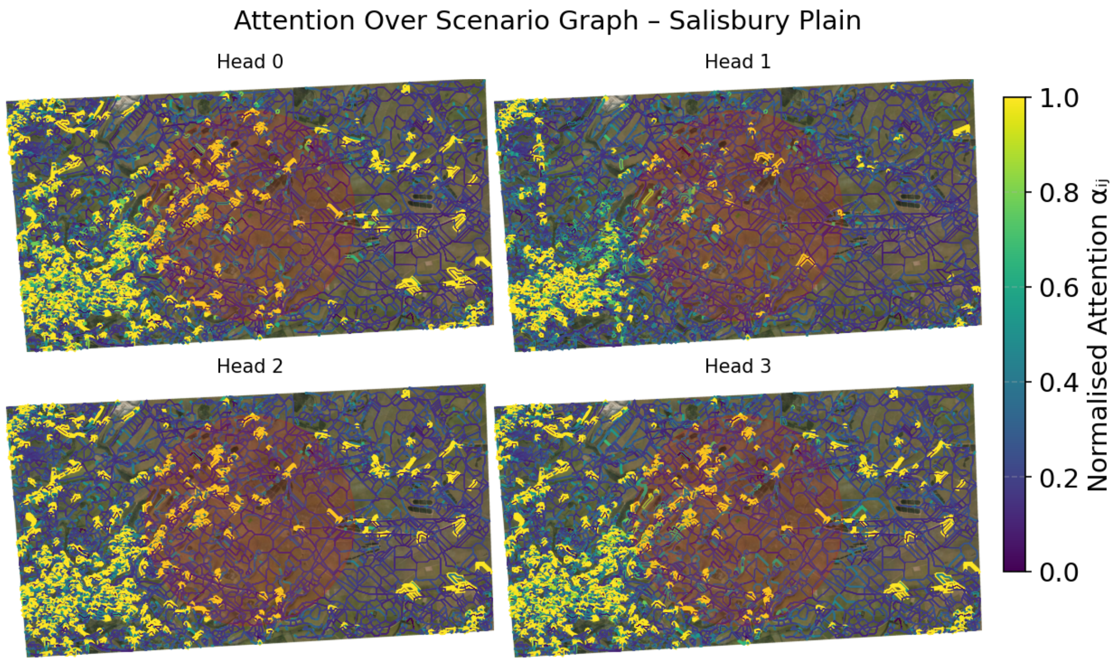


Figure 6.17: Normalized attention weights from the four heads in the GAT layer, highlighting attention on closely located, feature-similar urban nodes.

A pattern emerges where nodes in close proximity, particularly within the urbanized region (south-west region), display significantly higher attention scores. This concentration of attention in urban areas is consistent across all four heads, suggesting a pronounced feature similarity within these nodes. The high attention values imply that the model learns to prioritize node interactions that are geographically close and possibly share environmental or topographic features.

7

Discussion

7.1 Interpretation of Results

The study confirms that a *neuroevolutionary* approach for optimising a Graph Neural Network (GNN) with Separable CMA-ES can solve the non-differentiable, multi-route search-on-the-move (SOTM) radar-planning problem at proof-of-concept scale.

The outcome of the training process demonstrated the efficacy of the Separable Covariance Matrix Adaptation Evolution Strategy (SepCMA-ES) over traditional Genetic Algorithms (GAs) for optimising neural networks, indicating superior navigation of rugged, high-dimensional fitness landscapes [24, 14]. SepCMA-ES not only achieved higher fitness scores but also showed a greater ability to escape local optima. In Section 6.1 and Table 6.1, SepCMA-ES attained its highest performance with the more constrained *Compact* model, implying that a simpler architecture, when effectively optimised, may be sufficient or even preferable for the routing task. Furthermore, experiments show that inference times scale approximately linearly with the size of the scenario graph, enabling multi-route generation in seconds even on large terrains and supporting real-time re-planning in dynamic operational environments.

Analysis of mission performance across diverse terrains, such as Salisbury Plains and Vidsel (Table 6.3), demonstrated the system's capacity to generate multi-route coverage strategies. The objective function weights surveillance of the REST (Remaining Area) zone three times higher than the KOZ (Keep-Out Zone), steering coverage toward this broader area while still maintaining KOZ surveillance. The exploration component discretises time into 30-minute bins (Section 4.2.2.5) to encourage coverage in different regions within each time interval. In the Salisbury Plains training scenario (99.8 km^2), the KOZ maintained robust coverage (mean $\mu_{\text{KOZ}} \approx 62\,\%$) and achieved full coverage in roughly 30 minutes. The larger REST zone displayed generally short gaps, although one northern sector recorded a single gap of 75 minutes (Figure 6.9). Nevertheless, the vast majority of gaps in both KOZ and REST fell within the 30-minute revisit window, indicating that the exploration mechanism effectively maintains the desired coverage frequency across most of the operational area and angular sectors.

In the spatially shifted validation scenario, designed to test generalisation, the system remained adaptable. The REST zone exhibited some localised gaps, about 20 minutes in the southern and western regions—while KOZ gap distributions were similar to those in training. Full terrain coverage was typically achieved within one hour, demonstrating the planner's robustness under distribution shift.

7.2 Relation to Research Aims and Theoretical Framework

The primary aim was to create a multi-vehicle planner that (i) maximises area surveillance, (ii) minimises blind-time, and (iii) supports rapid planning. GNNs were chosen because their message-passing operations naturally encode spatial relations on traversability graphs and can, in principle, emulate classical routing algorithms [27, 20]. CMA-ES was selected for its proven ability to optimise non-differentiable objectives [24, 14], important as the objective includes discrete thresholding (node selection) and a greedy path-construction step. By restricting optimization to the network weights, this approach transforms a massive combinatorial search problem into a manageable continuous optimization task through representation learning.

The evolved policy learned an implicit patrol algorithm: nodes commanding wide angular visibility received high scores; edge scores reflected a learned trade-off between travel cost and coverage gain. The subsequent Dijkstra-style route builder then constructs these learned priorities. This two-level design, *learned heuristic* and *deterministic path finder*, aligns with the paradigm of neural algorithmic reasoning, where neural networks approximate classical algorithms but retain formal guarantees where needed.

7.3 Concept of Operation and Practical Deployment

To move beyond proof-of-concept and toward operational deployment, this section defines the intended use of the multi-route planning system for Search-on-the-Move (SOTM) radar operations, along with the practical steps required for real-world application. This includes terrain data preparation, operational planning, model training, route generation, and ultimately, real-time adaptability during live operations.

Initially, Digital Surface Model (DSM) data is converted into a traversability graph. This graph includes geographic coordinates, maximum allowed speeds, and distances between connected nodes, thus capturing the characteristics affecting terrain traversal. Each node in the graph has a unique identifier connected to a polygon that specifies radar coverage achievable at that location. Both the traversability graph and radar coverage polygons are maintained within an internal database. If the required traversability graph does not already exist for a chosen operational area, it can be generated from DSM data, ideally ahead of operational planning.

With the traversability graph prepared, the operational planning phase commences. During this phase, operational planners define critical operational zones directly, typically including a defended asset (DA) and an associated keep-out zone (KOZ), defined by a radius around the asset. After these zones are set, a scenario-specific graph, which incorporates all relevant operational constraints detailed in Section 4.2, is created and stored.

Following scenario definition, the multi-route planning starts, offering two distinct methods:
Strategy-Specific Model Training: A particular mission strategy is selected through an objective function, and a Graph Neural Network (GNN) is trained accordingly. Training can be conducted using a Genetic Algorithm (GA) suitable for standard or distributed hardware, or a Covariance Matrix Adaptation Evolution Strategy (CMA-ES), preferred for higher-performance computing resources. The resulting trained GNN model generates route solutions based on the number of vehicles specified by the operational planner.

Pre-Trained Model: In a practical and preferred scenario, operational planners utilise an existing library of pre-trained GNN models. Because each GNN has learned a generalised policy for route generation, it can produce solutions within seconds and can be re-invoked whenever the scenario graph changes (e.g. if obstacles appear or nodes become inaccessible). Planners simply update the graph by removing affected nodes and edges, and the same pre-trained model instantly recomputes viable routes without further training.

After route generation, a thorough analysis assesses the routes for vulnerabilities, such as terrain or routing issues, allowing planners to mitigate these during the planning stage. Such mitigations can include positioning stationary radar systems or implementing additional measures to strengthen the operation.

A significant advantage of the proposed method is its capability for real-time adaptability. Operational environments are subject to frequent and unpredictable changes, including previously unknown vegetational growth, damage to roads, or new obstacles and threats appearing suddenly. Because the route solutions are embedded within the trained neural network model, planners can immediately adapt the solutions by simply omitting affected nodes or regions from the scenario graph. This allows for rapid updates to operational routes without requiring additional training or lengthy recalculations.

7.4 Comparison with Earlier Work

Initial studies within this series focused on optimizing deployment schedules for single radar units, followed by an extension to static deployment optimization for multi-unit air defense companies [19, 18]. Both works utilized genetic algorithms and leveraged data provided by high-resolution geospatial data.

The cumulative work from these prior works has affirmed the benefits of combining geospatial data with machine learning, particularly evolutionary algorithms, for military planning in radar and air defense scenarios. These efforts developed methodologies for processing terrain data to assess traversability graphs.

The research presented in this thesis advances this established line of methods by tackling Search-On-The-Move (SOTM) multi-vehicle route planning. In comparison to the earlier emphasis on static deployments or pre-defined schedules for single or multiple units [19, 18], the neuroevolutionary GNN-based approach developed herein offers advantages. It is well-suited to handling the non-differentiable and often heuristic objective functions and enables rapid re-planning in response to evolving tactical situations or environmental dynamics. While the genetic algorithms employed in previous works were effective for their respective static or discrete scheduling problems, the current method is suited for continuous generation required for multiple mobile units maintaining coordinated surveillance while actively maneuvering.

The collective findings from this research series, demonstrate that these computational methods offers solutions to complex, real-world military planning challenges. The consistent success observed across radar surveillance, from optimizing static deployment locations [19, 18] to enabling dynamic multi-vehicle routing as achieved in this work, highlights the potential of machine learning.

7.5 Limitations

Several limitations constrained the scope and depth of this research. Firstly, the project encountered hardware limitations; the absence of a dedicated training cluster restricted the use of the full CMA-ES algorithm due to the memory demands of storing and manipulating the full covariance matrix on available GPU resources. CPU-based matrix operations, in contrast, scale poorly, thereby limiting overall compute capability and have limited the exploration of the search space, potentially leaving more optimal solutions undiscovered.

Secondly, time constraints impacted the optimization of the graph builder, which is responsible for loading and processing the full DSM dataset to generate the underlying traversability graphs. In our current implementation, the entire DSM is processed at once, which can quickly fill up available memory. Ideally, this would be replaced by a parallel approach that divides the DSM into smaller segments, processes each segment independently, and then stitches the resulting subgraphs together into a global traversability graph.

The time constraints also affected our data generation for training and validation. In total, five scenario graphs were constructed, three for training and two for validation. Ideally, a larger and more diverse dataset would be beneficial for more robust training and comprehensive assessment of the model's generalization capabilities in a real-world deployment; however, this limitation is primarily time limitation, given additional time, a substantially larger pool of scenarios could be generated with the existing tooling.

A notable constraint in the current model is the limited consideration of detailed terrain geography and topology. While DSM data is used for traversability and line-of-sight calculations, subtler tactical aspects of terrain, such as features that might expose a vehicle to isolation, create vulnerabilities, or offer strategic choke points, were not explicitly incorporated into the feature set due to challenges in defining such features and time limitations. This represents an area where the current feature set could be improved. Due to the aforementioned constraints, several additional analyses (described in 8.1) could not be performed.

8
Conclusion

This thesis successfully developed and validated a novel neuroevolutionary framework for multi-route planning in search-on-the-move (SOTM) radar operations, demonstrating its efficacy in tackling complex, non-differentiable military planning objectives using machine learning. The research culminated in the adoption of an approach coupling Graph Neural Networks (GNNs) with the Separable Covariance Matrix Adaptation Evolution Strategy (SepCMA-ES), a combination that proved superior to traditional Genetic Algorithms for this task. This methodological choice was pivotal for navigating the vast search space inherent in multi-vehicle coordination and for optimizing the non-differentiable objective function, which incorporated discrete node selection and greedy path-construction steps. A key advantage of this neuroevolutionary strategy was its ability to constrain the complex combinatorial optimization problem to the significantly smaller, continuous parameter space of the neural network. The system demonstrated the capacity to generate cooperative patrol routes that ensure synchronized surveillance across designated areas, consistently maintaining Keep-Out Zone (KOZ) blind-time below set thresholds in diverse simulated landscapes. Furthermore, the trained models exhibited generalization to unseen scenarios, while also enabling route generation in seconds and thereby supporting the critical capability for real-time re-planning in dynamic operational environments. The central objective of generating effective, cooperative patrols was achieved by leveraging high-resolution geospatial data transformed into a traversable graph representations.

8.1 Future Work

Research efforts still remain to fully develop the method and system, moving it closer to a robust, deployable solution.

8.1.1 Data Processing and Graph Generation

An area for improvement lies in the initial data-processing stage. The current graph builder, while functional as a proof of concept, processes the entire Digital Surface Model (DSM) dataset at once. This approach leads to significant memory constraints when dealing with larger terrains. A scalable builder could ideally divide the DSM into smaller, overlapping tiles that can be processed in parallel; a margin of overlap would ensure that nodes created on neighbouring tiles share identical (x, y) coordinates, allowing them to be merged reliably when stitching the sub-graphs into a global traversability graph.

8.1.2 Feature Engineering and Scenario Representation

The current set of node and edge features, as detailed in Section 3.2, provides a basis for our operational approach. Nonetheless, refining the environmental representation could yield more nuanced route planning. Future work should incorporate a broader range of descriptors, including detailed geographical and topological characteristics that highlight tactical aspects of terrain—e.g. exposure indices, natural funnels, choke points, or flat versus rugged areas. Temporal features, such as dynamically evolving threat levels in specific regions, could also be valuable.

Dynamic factors (emerging threats, weather changes, updated traversability) can be incorporated in two complementary ways. **(i) Feature-based adaptation:** treat these factors as time-varying inputs to the GNN, allowing the network itself to learn how to reroute under changing conditions. **(ii) Graph-level modification:** update the scenario graph in real time by temporarily removing or re-weighting nodes and edges affected by new conditions, and then re-evaluate it with the static, pre-trained model. The latter approach simplifies the GNN's task, while the former may offer finer-grained responsiveness. Further experimentation is required to determine which option, or what method strikes the best balance between operator control, computational cost, and solution quality.

8.1.3 Diversification of Objective Functions and Strategic Modeling

The current objective function modeled synchronized coverage with a focus on surveillance for incoming threats into the Keep-Out Zone (KOZ). However, real-world military operations encompass a wider array of tactical objectives. Developing a library of distinct objective functions, each encapsulating a different SOTM military strategy, would significantly enhance system versatility. Examples of such strategies include:

- **Mutual Support:** Prioritizing routes where radar units can cover each other's dead zones (e.g., the cone directly above a radar unit) or provide inter-vehicle support.
- **Stealth-Oriented Approaches:** Emphasizing routes that minimize radar emissions or utilize terrain masking to reduce detectability.
- **Dynamic Threat Avoidance/Engagement:** Routes that actively maneuver to avoid known threat areas or, conversely, to maintain surveillance on active threats safely.
- **Convoy Protection/Route Surveillance:** Planning routes to protect a moving asset or to ensure surveillance along a specific transportation corridor from point A to point B.

Building upon this, future efforts could focus on developing and training separate neural network models tailored to these military strategies and for varying numbers of radar-equipped vehicles. This would involve creating larger and more diverse datasets of scenarios, covering a wide range of terrains, and training of GNN models on these strategy-focused objectives. Such a development would allow an operator to select a desired strategy and the number of available units, with the system then deploying the corresponding pre-trained model for rapid solution generation.

8.1.4 Transition to a Deployed System

Finally, to transition this research into a practical tool for operational planners, several development steps are necessary. Once the core components, including data processing and diverse strategy models have matured, the should then shift towards the practical aspects of deployment. This includes the design and implementation of a user-friendly software architecture and interface that allows operators to easily define scenarios, select strategies, visualize planned routes, and analyze performance metrics. Furthermore, ensuring that the system can integrate seamlessly with broader command and control or existing mission planning systems.

Bibliography

[1] Gdal viewshed. `https://gdal.org/en/stable/programs/gdal_viewshed.html`, 2025. Accessed: 2025-04-11.

[2] Saab AB. Giraffe 1x, 2025. Accessed: 2025-04-02.

[3] Shaked Brody, Uri Alon, and Eran Yahav. How attentive are graph attention networks?, 2022.

[4] Ming Chen, Zhewei Wei, Bolin Ding, Yaliang Li, Ye Yuan, Xiaoyong Du, and Ji-Rong Wen. Scalable graph neural networks via bidirectional propagation, 2021.

[5] Charles Darwin. *On the Origin of Species by Means of Natural Selection, or the Preservation of Favoured Races in the Struggle for Life.* John Murray, London, 1859.

[6] Kalyanmoy Deb and Hans-Georg Beyer. Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary Computation*, 9(2):197–221, 2001. PMID: 11382356.

[7] Department of Defense, USA. MIL-STD-89020B: Multispectral Image Data of Landsat, 1996. Accessed: 2025-04-03.

[8] Edsger W Dijkstra. A note on two problems in connexion with graphs. *Numerische mathematik*, 1(1):269–271, 1959.

[9] Michael L. Fredman and Robert Endre Tarjan. Fibonacci heaps and their uses in improved network optimization algorithms. *J. ACM*, 34(3):596–615, July 1987.

[10] Sean Gillies, Casper van der Wel, Joris Van den Bossche, Mike W. Taves, Joshua Arnott, Brendan C. Ward, et al. Shapely, 2025. Manipulation and analysis of geometric objects in the Cartesian plane.

[11] Abhishek Gupta and B. Mishra. Neuroevolving monotonic pinns for particle breakage analysis. pages 993–996, 06 2024.

[12] Mordechai Haklay and Patrick Weber. Openstreetmap: User-generated street maps. *IEEE Pervasive Computing*, 7(4):12–18, 2008.

[13] Nikolaus Hansen. *The CMA Evolution Strategy: A Comparing Review*, volume 192, pages 75–102. 06 2007.

[14] Nikolaus Hansen. The cma evolution strategy: A tutorial, 2023.

[15] Robert M. Haralick, Stanley R. Sternberg, and Xinhua Zhuang. Image analysis using mathematical morphology. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, PAMI-9(4):532–550, 1987.

[16] John H. Holland. Genetic algorithms. *Scientific American*, 267(1):66–73, 1992.

[17] Kurt Hornik. Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251–257, 1991.

[18] Erik Johansson and Sofia Karlsson. Deployment of air defense. Master's thesis, Chalmers University of Technology, Gothenburg, Sweden, 2019.

[19] Kevin Larsson. Deployment planning for artillery hunting radar systems using high-resolution geospatial data. Master's thesis, Chalmers University of Technology, Gothenburg, Sweden, 2018.

[20] Tong Liu and Hadi Meidani. Graph neural networks for travel distance estimation and route recommendation under probabilistic hazards, 2025.

[21] Bernhard Mehlig. *Machine Learning with Neural Networks: An Introduction for Scientists and Engineers*. Cambridge University Press, October 2021.

[22] Giannis Nikolentzos, George Dasoulas, and Michalis Vazirgiannis. k-hop graph neural networks, 2020.

[23] OpenStreetMap Contributors. Openstreetmap copyright and license, 2024. Accessed: 15-Jan-2025.

[24] Raymond Ros and Nikolaus Hansen. A simple modification in cma-es achieving linear time and space complexity. In Günter Rudolph, Thomas Jansen, Nicola Beume, Simon Lucas, and Carlo Poloni, editors, *Parallel Problem Solving from Nature – PPSN X*, pages 296–305, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[25] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002.

[26] Felipe Petroski Such, Vashisht Madhavan, Edoardo Conti, Joel Lehman, Kenneth O. Stanley, and Jeff Clune. Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning, 2018.

[27] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks, 2018.

[28] Jianjun Wang, Gary Robinson, and Kevin White. Generating viewsheds without using sightlines. 66, 01 2000.

[29] Jian Cheng Wong, Abhishek Gupta, and Yew-Soon Ong. Can transfer neuroevolution tractably solve your differential equations? *IEEE Computational Intelligence Magazine*, 16(2):14–30, 2021.

[30] Jian Cheng Wong, Abhishek Gupta, Chin Chun Ooi, Pao-Hsiung Chiu, Jiao Liu, and Yew-Soon Ong. Evolutionary optimization of physics-informed neural networks: Survey and prospects, 2025.

[31] Jie Zhou, Ganqu Cui, Shengding Hu, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, Lifeng Wang, Changcheng Li, and Maosong Sun. Graph neural networks: A review of methods and applications. *AI Open*, 1:57–81, 2020.

A

Appendix 1

A.1 Vidsel Training Scenario

The Vidsel training scenario spans approximately 11 hours and shows the cooperation between the vehicles in the routes, indicated by their coverage around the KOZ and the broader REST area. As illustrated in Figure A.1.

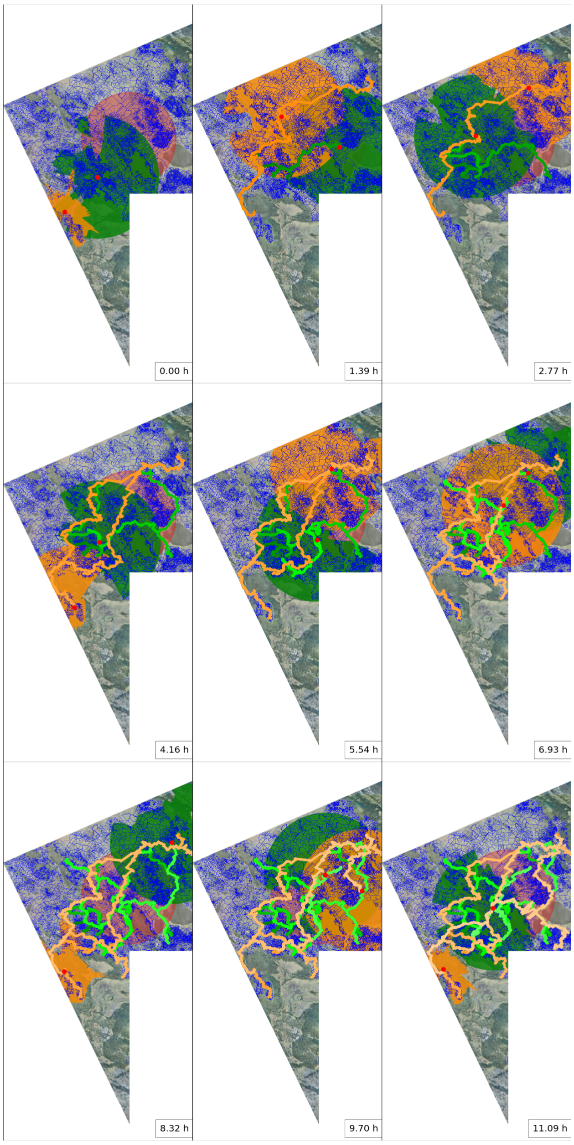


Figure A.1: Chronological snapshots of the multi-route patrol (t=$\{0, 0.1.39, 2.77, \ldots, 11.09\}$ h). Underlying map: Vidsel. The blue graph structure overlaid on the terrain is the traversability graph. Polylines are the planned paths (color-gradient by timestep); the red dot indicates the current vehicle positions. Shaded discs correspond to current radar footprints and the solid red circle marks the KOZ boundary.

The cumulative radar coverage heatmap (Figure A.2) reveals the highest intensity within the central KOZ, gradually diminishing outward. Noticeably, the lower part of the terrain receives limited coverage due to traversal constraints, shown by the lack traversal paths in the graph (blue).

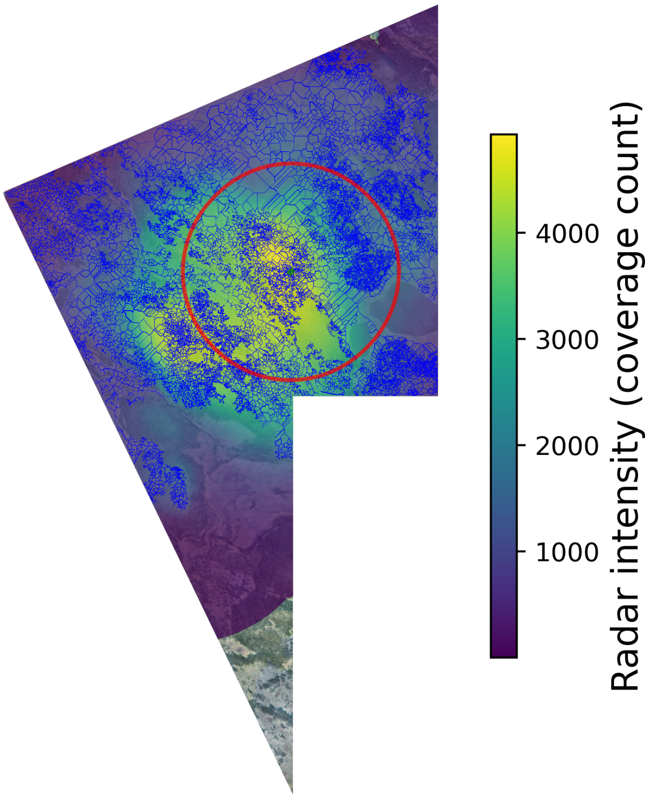


Figure A.2: Cumulative radar coverage heatmap over the Vidsel Test Range, showing the total number of revisits (colorbar) with the traversable terrain graph overlaid in blue. Highest revisit density occurs within the central KOZ, while the southern sector exhibits limited coverage due to traversal constraints.

Instantaneous radar coverage (Figure A.3) shows the KOZ having a mean coverage of 76% with minimal variance of 16%, the REST area also maintains stable coverage with a mean of 32% and low variance of 7%.

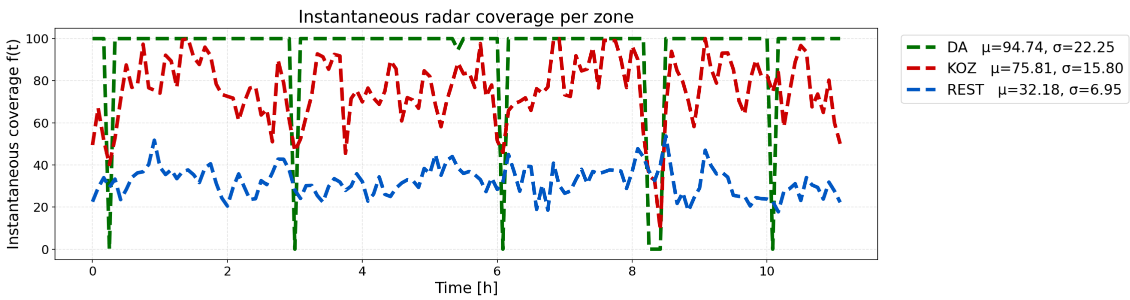


Figure A.3: Instantaneous radar coverage fraction $f(t)$ for the Vidsel mission: DA (green dashed, $\mu = 95\%, \sigma = 22\%$) KOZ (red dashed, $\mu = 76\%, \sigma = 16\%$) and REST (blue dash-dot, $\mu = 32\%, \sigma = 7\%$), shows stable surveillance; DA coverage (green dashed) remains consistently high.

The cumulative radar coverage (Figure A.4) further validates the complete coverage of the KOZ within approximately one hour. REST coverage is substantially completed around two hours into the mission, subsequently increasing linearly to around 90%, limited primarily by vehicle traversal capabilities.

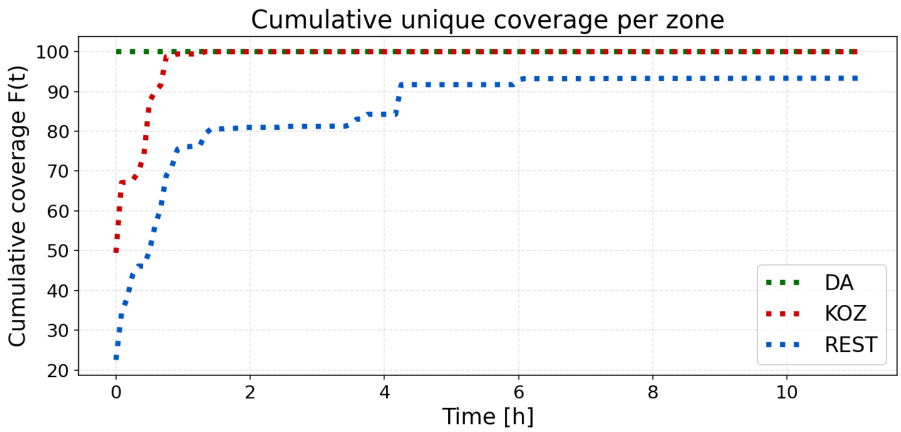


Figure A.4: Cumulative unique radar coverage $F(t)$ for Vidsel: KOZ (red) reaches full coverage by $t \approx 1$ h; REST (blue) attains 90 % by $t \approx 2$ h and grows linearly thereafter, limited by vehicle traversal capabilities.

Sector-based metrics highlight the geographical surveillance. Figure A.5 for the DA shows overall complete coverage, with a notable 15-minute coverage gap occurring approximately 8 hours into the mission. Apart from small gaps in the interval, the DA experiences continuous coverage.

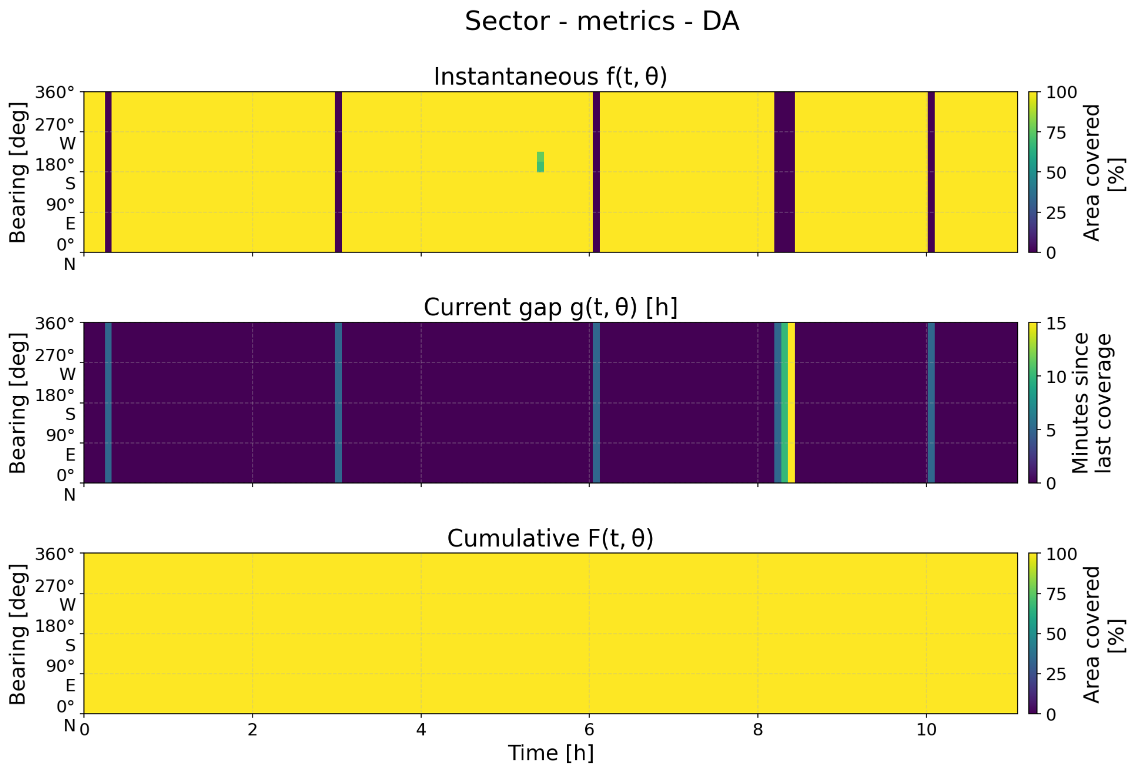


Figure A.5: DA sector metrics for Vidsel: top—instantaneous coverage $f(t, \theta)$; middle—current gap $g(t, \theta)$ in minutes since last revisit; bottom—cumulative coverage $F(t, \theta)$. Only a single notable 15 min gap occurs around $t \approx 8$ h, with otherwise continuous coverage across all bearings ($0° =$ N, clockwise).

The KOZ similarly shows robust surveillance (Figure A.6), with only a single significant gap in the western and southern regions at the same 8-hour mark, again lasting about 15 minutes.

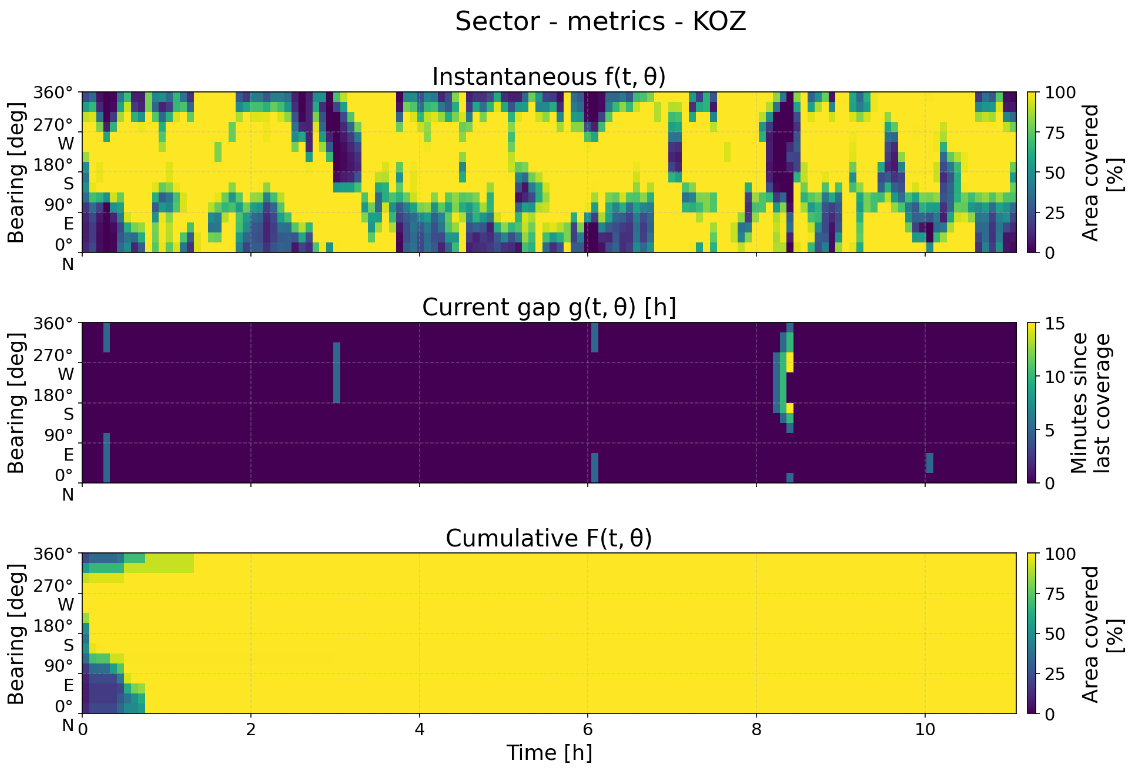


Figure A.6: KOZ sector metrics for Vidsel: instantaneous coverage $f(t, \theta)$, gap duration $g(t, \theta)$, and cumulative coverage $F(t, \theta)$. A solitary 15 min outage appears in western/southern sectors near $t \approx 8$ h, with full directional coverage maintained elsewhere.

Lastly, the REST area's sector coverage (Figure A.7). Noticeably the coverage gaps occur at approximately 2.2, 2.5, 5, and 6.3 hours into the mission, each spanning 40 to 60 minutes primarily in the north to northeastern sectors, except at 2.5 hours, where a western gap also emerges. Nonetheless, the route maintain around 75% cumulative coverage consistently in the western sector.

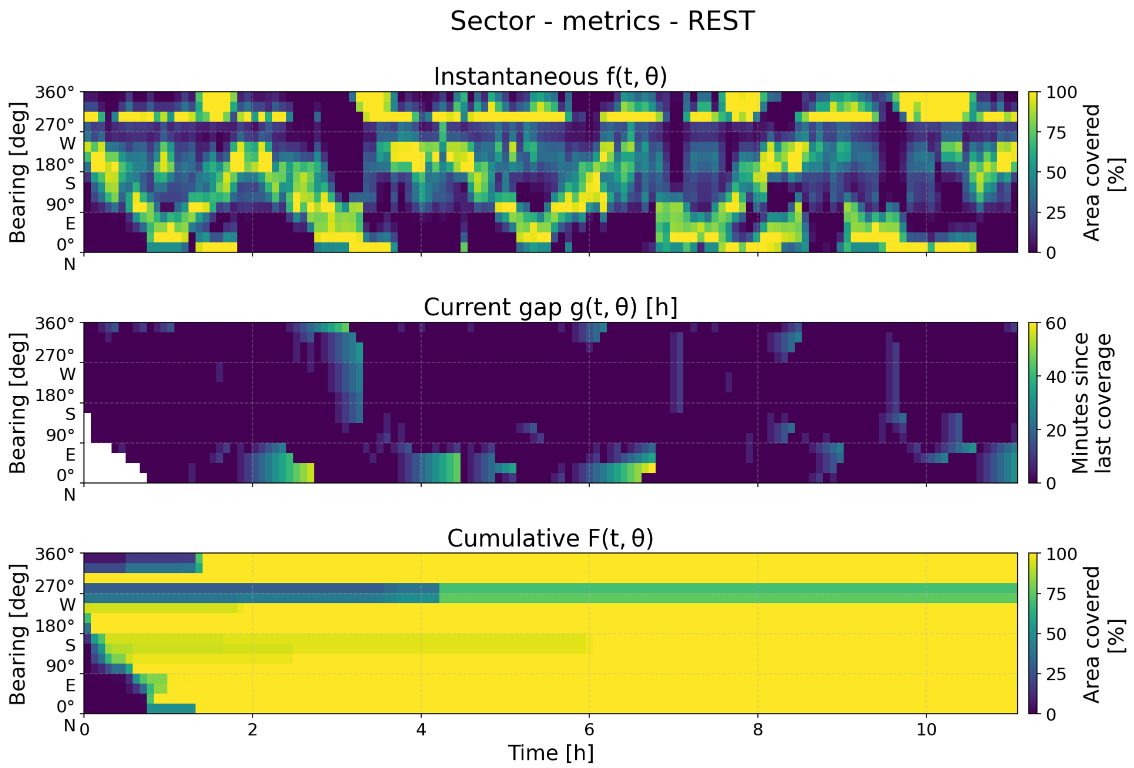


Figure A.7: REST sector metrics for Vidsel: $f(t, \theta)$, $g(t, \theta)$, and $F(t, \theta)$. Noticeable gaps at $t \approx 2.2$, 2.5, 5.0, and 6.3 h span 40–60 min in N–NE bearings (and at 2.5 h also in W); western sectors sustain 75 % cumulative coverage throughout.

A.2 Los Angeles Training Scenario

In the Los Angeles training scenario, the solution adopted a distinctly different strategy by positioning one unit primarily around the Keep-Out Zone (KOZ) for immediate surveillance, while the other unit prioritized sea-based coverage, exploiting the available terrain. Figure A.8 illustrates this strategic division, highlighting how challenging terrain conditions, primarily the mountainous regions covering most of the area, significantly limited route traversability and thus complicated optimal surveillance.

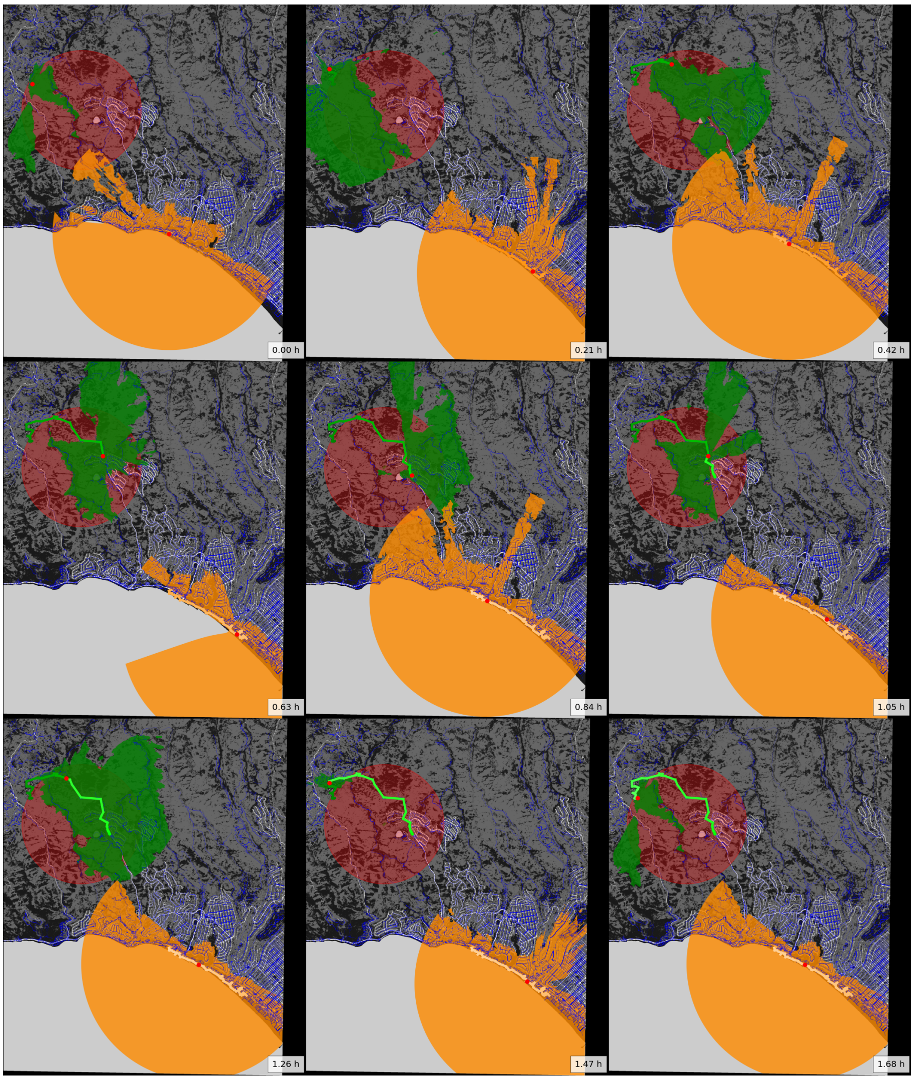


Figure A.8: Chronological snapshots of the multi-route patrol (t={0, 0.21, 0.42, . . . , 1.68} h). Underlying map: Los Angeles. The blue graph structure overlaid on the terrain is the traversability graph. Polylines are the planned paths (color-gradient by timestep); the red dot indicates the current vehicle positions. Shaded discs correspond to current radar footprints and the solid red circle marks the KOZ boundary.

Radar heatmap analysis (Figure A.9) further shows the challenge, showing numerous coverage gaps, particularly pronounced in the mountainous northeastern regions and the southwestern areas. The highest intensity of radar coverage occurs in the southeastern sea-bound region and directly surrounding the KOZ, aligning with the strategic priorities of the adopted surveillance strategy.

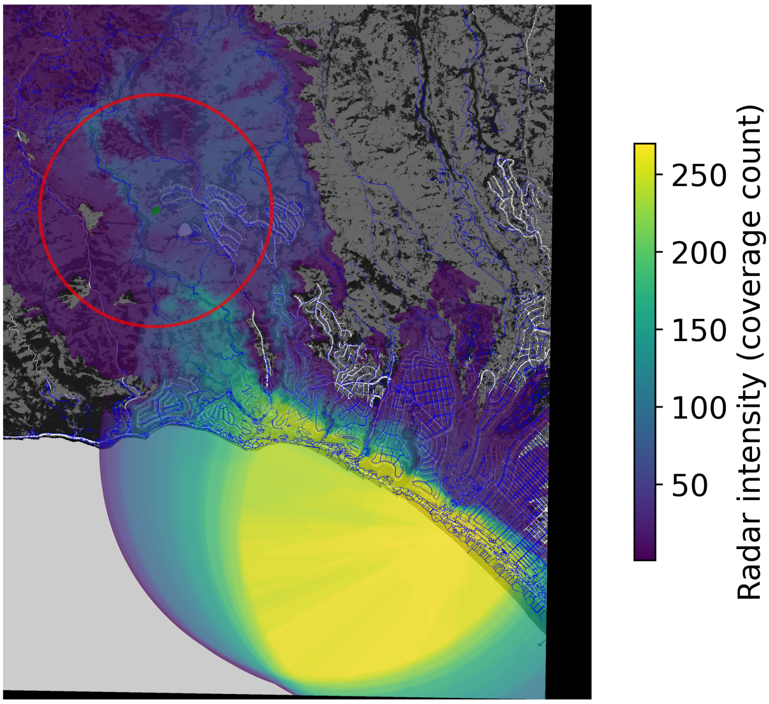


Figure A.9: Spatial heatmap of radar coverage intensity over the Los Angeles training area. Colorbar shows coverage count per grid cell; highest revisit density occurs in the southeastern sea-bound region and around the KOZ (red circle), reflecting prioritized maritime patrols, while pronounced gaps appear in the mountainous northeastern sectors and the southwestern interior due to terrain constraints.

Instantaneous radar coverage metrics (Figure A.10) reflect these spatial challenges, with notably low REST area coverage (mean coverage of 12%) and modest KOZ coverage (mean of 29%). Additionally, the DA exhibits high coverage variance (49%) despite an acceptable mean coverage of 38%, indicating significant surveillance difficulties in maintaining continuous protection due to challenging terrain constraints.

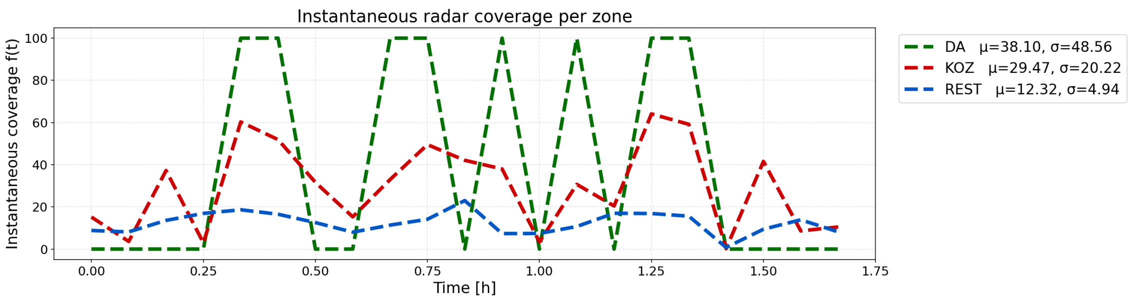


Figure A.10: Time-series of instantaneous radar coverage fraction $f(t)$ for the Los Angeles scenario: Defended Asset (DA, green), Keep-Out Zone (KOZ, red), and REST area (blue). DA mean $\mu_{DA} = 38\%$ with high variance $\sigma_{DA} = 49\%$; KOZ mean $\mu_{KOZ} = 29\%$; REST mean $\mu_{REST} = 12\%$. Fluctuations highlight surveillance challenges imposed by rugged terrain.

The cumulative coverage shown in figure A.11 illustrates that the DA and KOZ are rapidly gets covered within the first 15 minutes of the mission. However, REST coverage quickly stagnates around 40%, emphasizing the significant traversal limitations imposed by terrain.

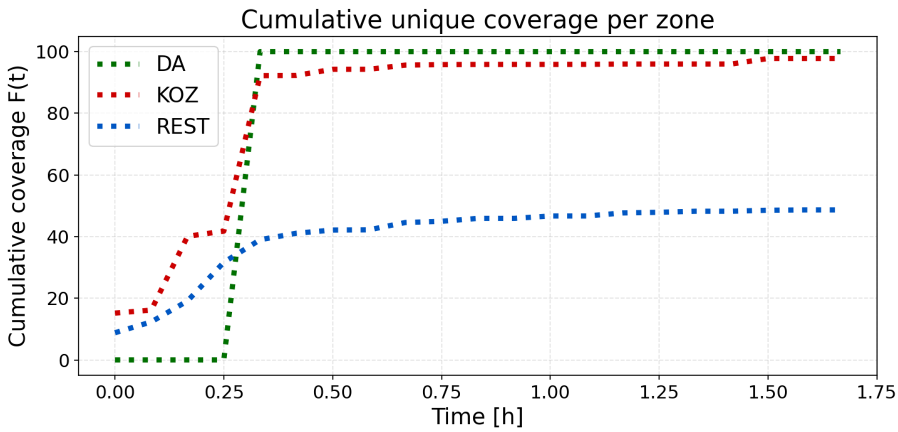


Figure A.11: Cumulative unique coverage $F(t)$ per zone in the Los Angeles scenario. DA and KOZ reach full coverage within the first 15 minutes of the mission; REST coverage plateaus at approximately 40% thereafter, indicating persistent areas unvisited due to mountainous and access limitations.

Sector-based analysis of the DA (Figure A.12) identifies a critical coverage gap of approximately 20 minutes toward the mission's end, signifying a substantial lapse in continuous protection.

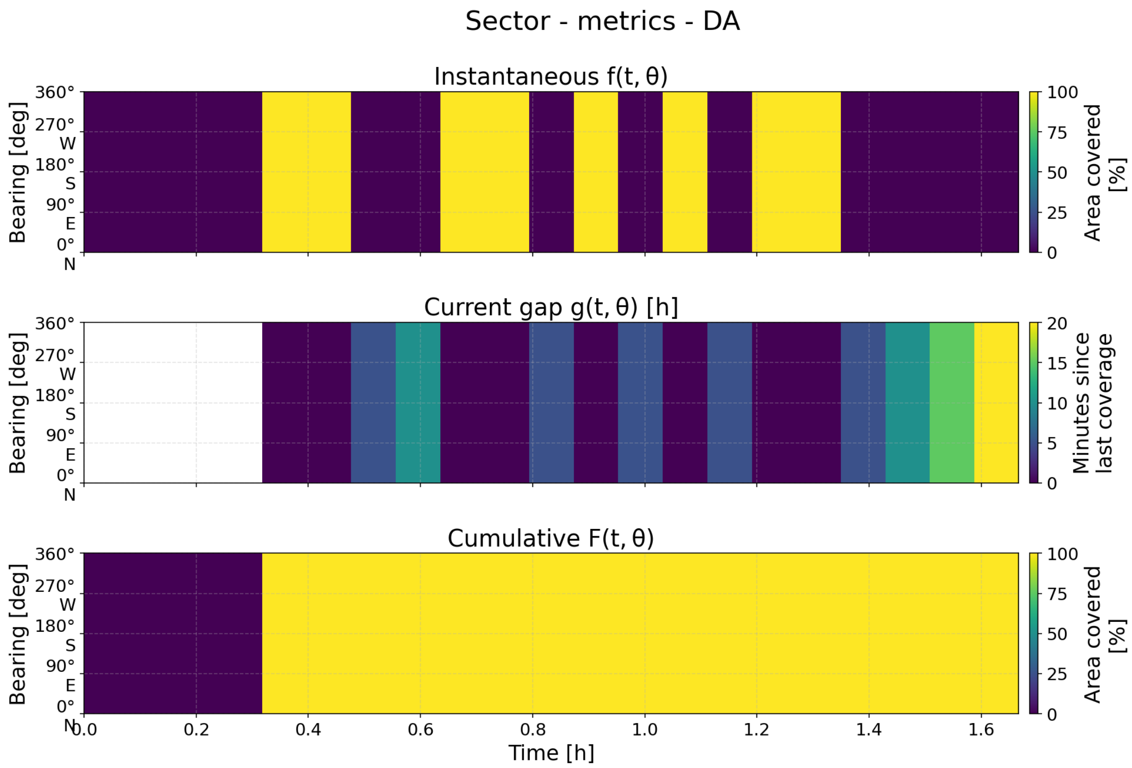


Figure A.12: Sector-wise metrics for the DA in the Los Angeles scenario. Top: instantaneous coverage $f(t, \theta)$ by bearing θ; middle: current revisit gap $g(t, \theta)$ in minutes since last coverage; bottom: cumulative coverage $F(t, \theta)$. A critical ~ 20 min gap occurs near mission end in the northeastern bearings, reflecting lapse in continuous DA protection.

Similarly, KOZ sector metrics (Figure A.13) reflect a significant gap of approximately 20 minutes towards the northern region near mission-end, with cumulative coverage notably lower (around 75%) in the western sectors.

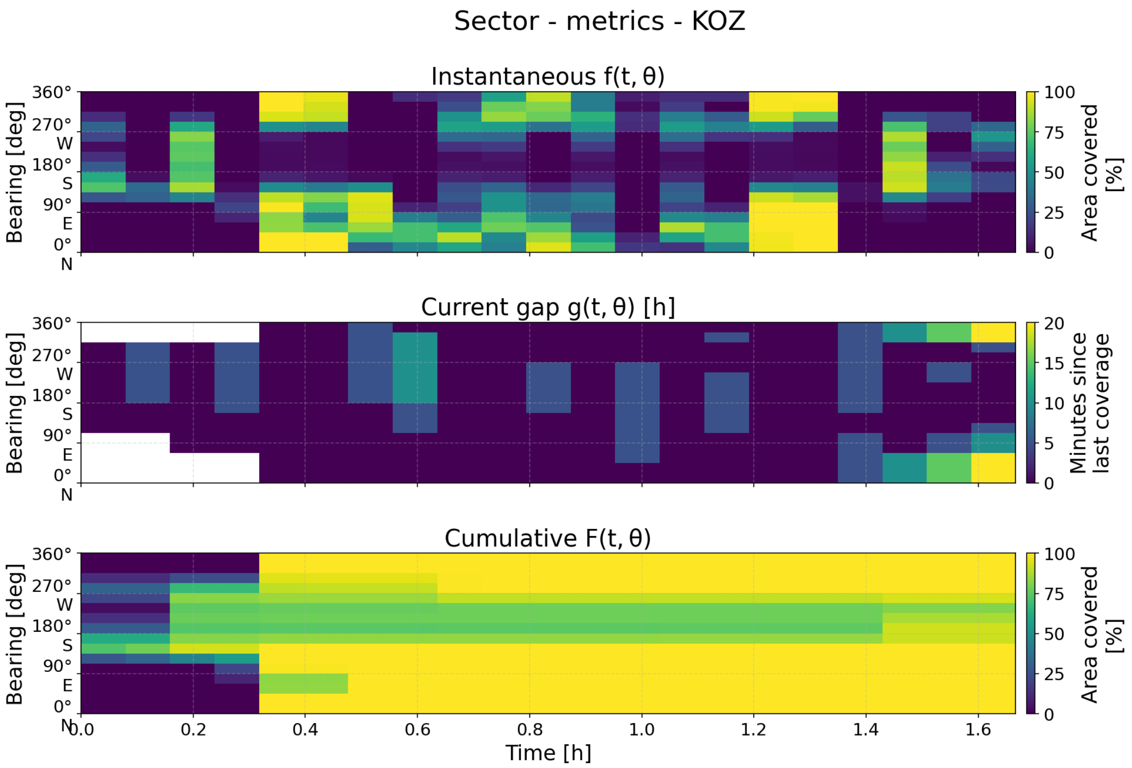


Figure A.13: Sector-wise metrics for the KOZ in the Los Angeles scenario. Top: instantaneous coverage $f(t, \theta)$; middle: gap duration $g(t, \theta)$ in minutes; bottom: cumulative coverage $F(t, \theta)$. A $\sim$20 min gap emerges near mission end in northern sectors, while western bearings only achieve $\sim$75% cumulative coverage by mission close.

The REST area coverage (Figure A.14) remains limited, notably deficient in the mountainous northeastern terrain, which receives virtually no coverage due to severe traversal constraints. This underscores the substantial difficulty faced by the model in effectively surveying such challenging terrains.

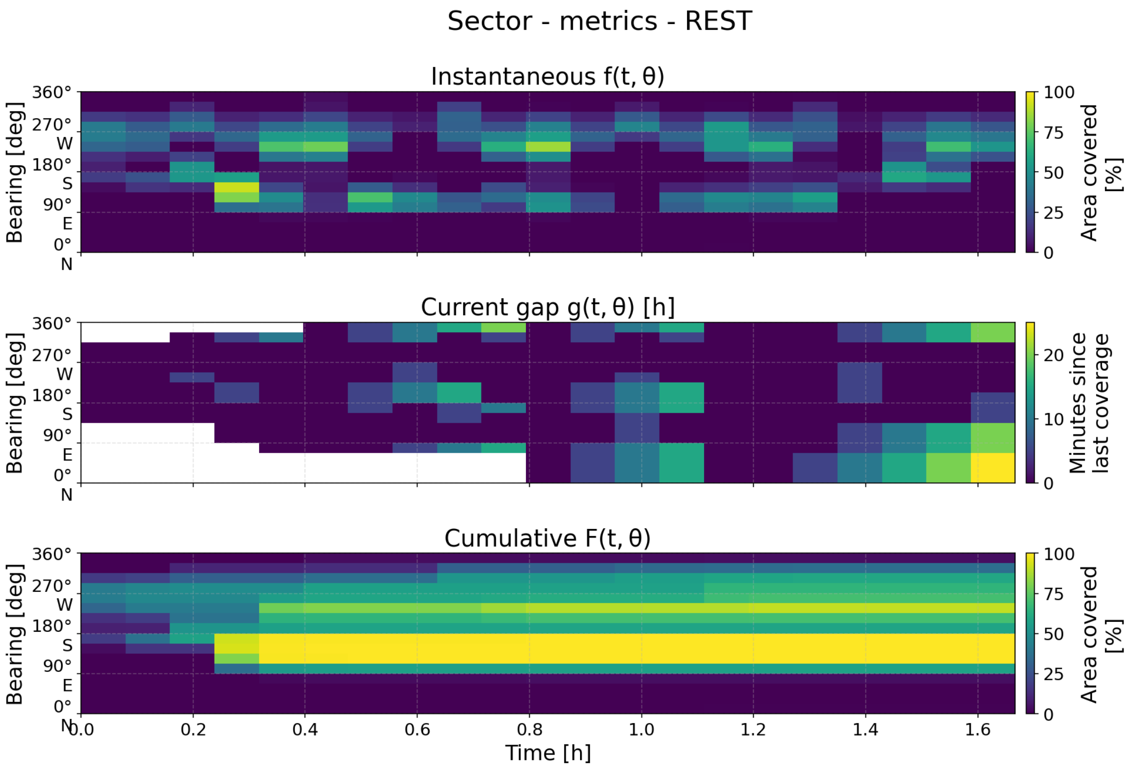


Figure A.14: Sector-wise metrics for the REST area in the Los Angeles scenario. Top: instantaneous coverage $f(t,\theta)$; middle: current gap $g(t,\theta)$ in minutes; bottom: cumulative coverage $F(t,\theta)$. Coverage is almost completely absent in the mountainous northeastern sectors, and many bearings remain under-served due to severe traversal constraints.

A.3 Vidsel Validation Scenario

In this validation scenario, the Keep-Out Zone (KOZ) has been repositioned to the northwest region and reduced in size compared to the training scenario. As Figure A.15 illustrates, the multi-route model effectively adapts by shifting its routes explicitly toward the new KOZ location, underscoring the model's ability to generalize effectively to previously unseen scenarios.

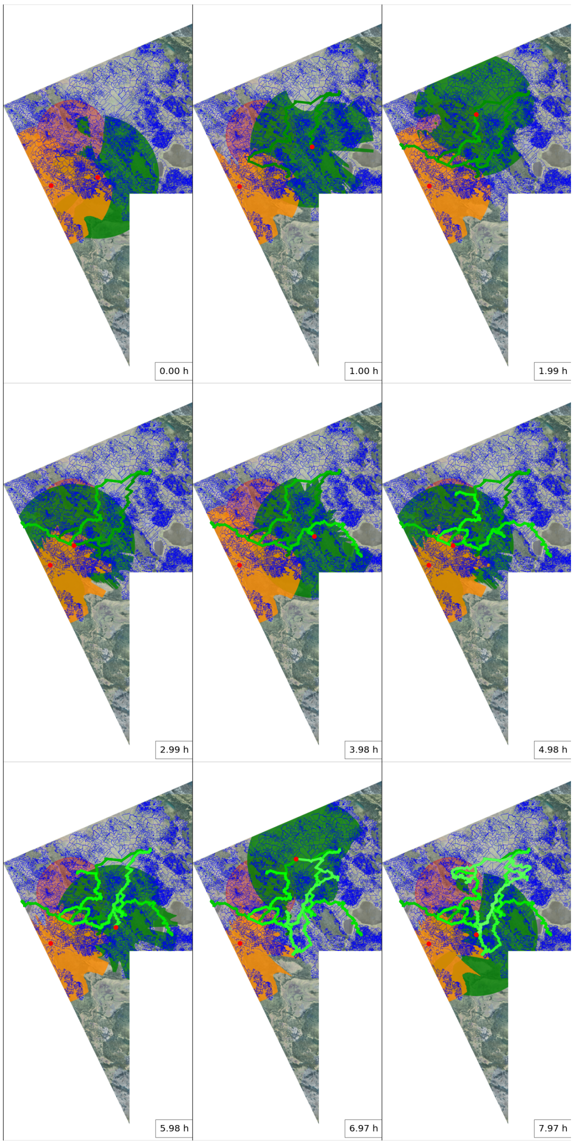


Figure A.15: Chronological snapshots of the multi-route patrol (t={0, 1.0, 1.99, . . . , 7.97} h). Underlying map: Vidsel. The blue graph structure overlaid on the terrain is the traversability graph. Polylines are the planned paths (color-gradient by timestep); the red dot indicates the current vehicle positions. Shaded discs correspond to current radar footprints and the solid red circle marks the KOZ boundary.

The radar coverage heatmap (Figure A.16) confirms a slight northwest shift in surveillance intensity relative to the training scenario (Figure A.2), with notable central coverage. There is also an observable reduction in radar coverage in the northeastern REST region, aligning with the shifted strategic priorities.

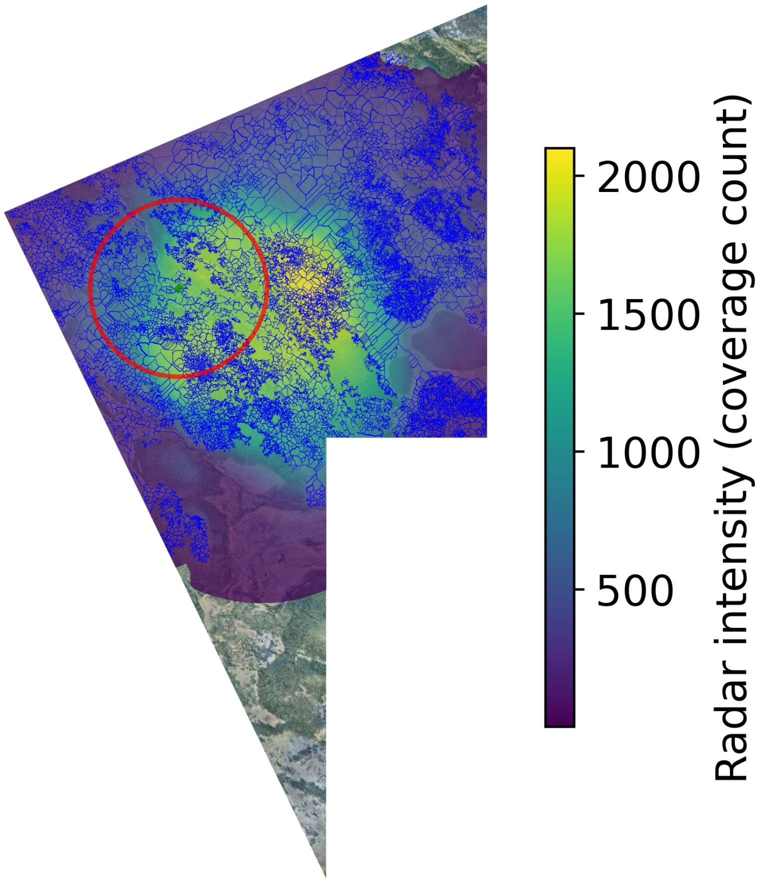


Figure A.16: Heatmap of cumulative radar revisit counts over the Vidsel Test Range validation area. Underlying terrain and road network in grey/blue; colorbar shows number of coverage events per grid cell. Surveillance intensity exhibits a northwest shift relative to the training scenario, with strong central coverage and reduced revisit density in the northeastern REST region.

Instantaneous coverage metrics (Figure A.17) show stable coverage of the KOZ with a mean of 67% and moderate variance ($\sigma = 27\%$). REST coverage is equally stable, presenting a mean of 35% and a low variance of 8%, emphasizing consistent surveillance.

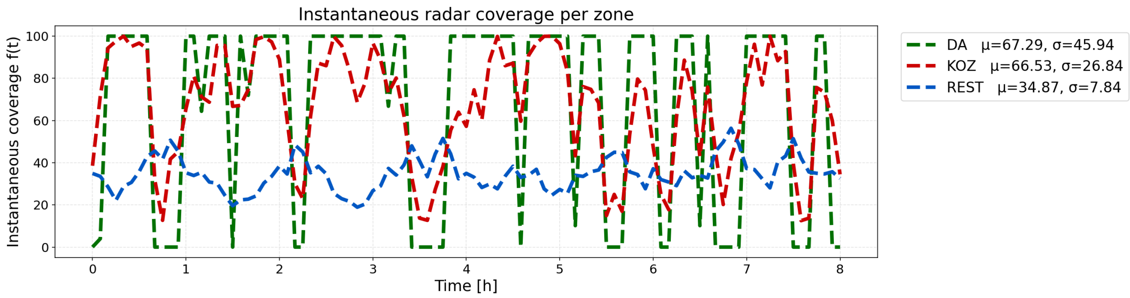


Figure A.17: Instantaneous radar coverage fraction $f(t)$ for the KOZ (red dashed) and REST (blue dash-dot) regions during the validation mission. KOZ achieves a mean coverage of $\mu_{KOZ} = 0.67$ ($\sigma = 27\%$), while REST records $\mu_{REST} = 35\%$ ($\sigma = 8\%$), indicating stable and consistent surveillance.

Cumulative radar coverage (Figure A.18) demonstrates rapid initial coverage of the KOZ and Defended Asset (DA), suggesting the vehicles were strategically initialized close to these critical regions. REST coverage reaches approximately 80% within the first hour.

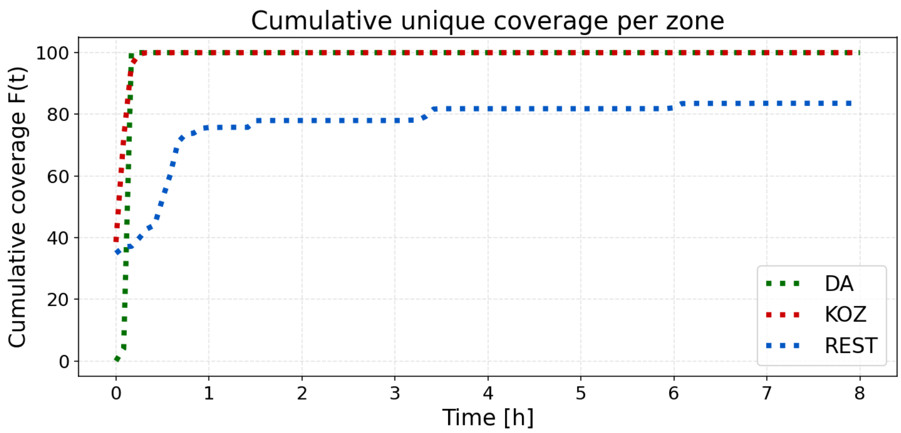


Figure A.18: Cumulative unique coverage $F(t)$ for the DA (green), KOZ (red), and REST (blue) regions in the validation scenario. DA and KOZ reach full coverage at mission start, reflecting initial vehicle placement within KOZ and DA. REST attains approximately 80% coverage within the first hour.

Sector analysis of the DA (Figure A.19) reveals several coverage gaps aligning closely with the mission's 20-minute revisit objective. Notable gaps occur approximately 1, 3.8, and 7 hours into the mission.

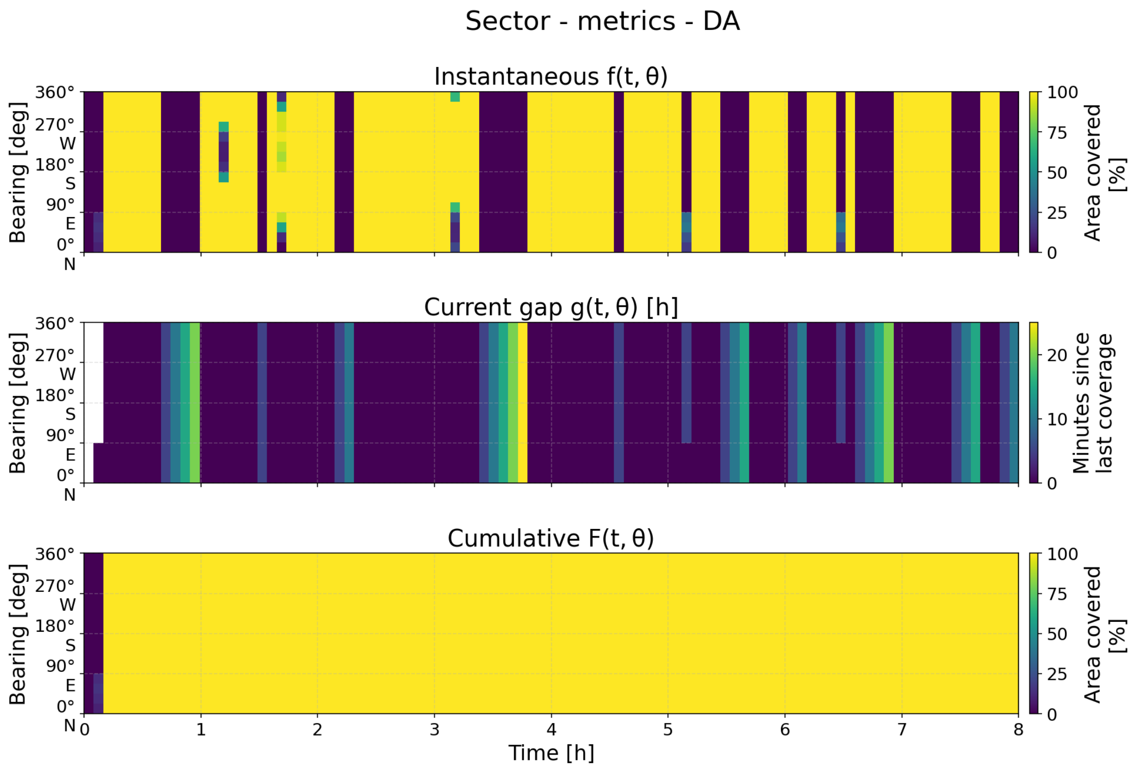


Figure A.19: Sector-wise metrics for the DA zone in the Vidsel validation. Top: instantaneous coverage $f(t, \theta)$ by bearing θ (0°=N, clockwise); middle: current gap $g(t, \theta)$ in minutes since last revisit; bottom: cumulative coverage $F(t, \theta)$. Observed coverage gaps at $t \approx 1$, 3.8, and 7 h.

KOZ metrics (Figure A.20) display consistent instantaneous coverage, with only one significant 20-minute gap occurring around 3.5 hours into the mission in the southeast sector, further demonstrating robust and adaptive surveillance.

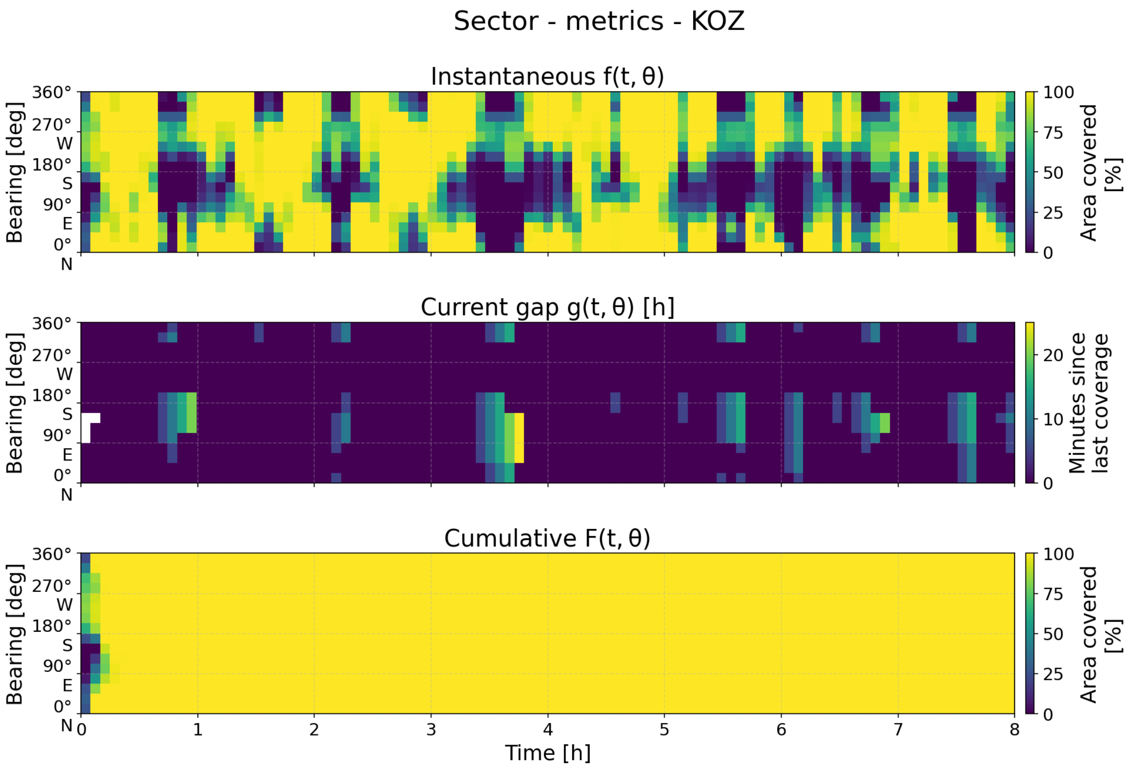


Figure A.20: Sector-wise metrics for the KOZ zone in validation. Top: $f(t,\theta)$; middle: $g(t,\theta)$ (minutes); bottom: $F(t,\theta)$. Coverage is nearly continuous, with only one 20-minute gap occurring at $t \approx 3.5$ h in the southeast bearing.

The REST area's coverage (Figure A.21) closely mirrors the training scenario, with a minor 20-minute coverage gaps around 1, 2.5, and 5.8 hours into the mission, predominantly in the northwestern sectors, along with smaller gaps in southern sectors.

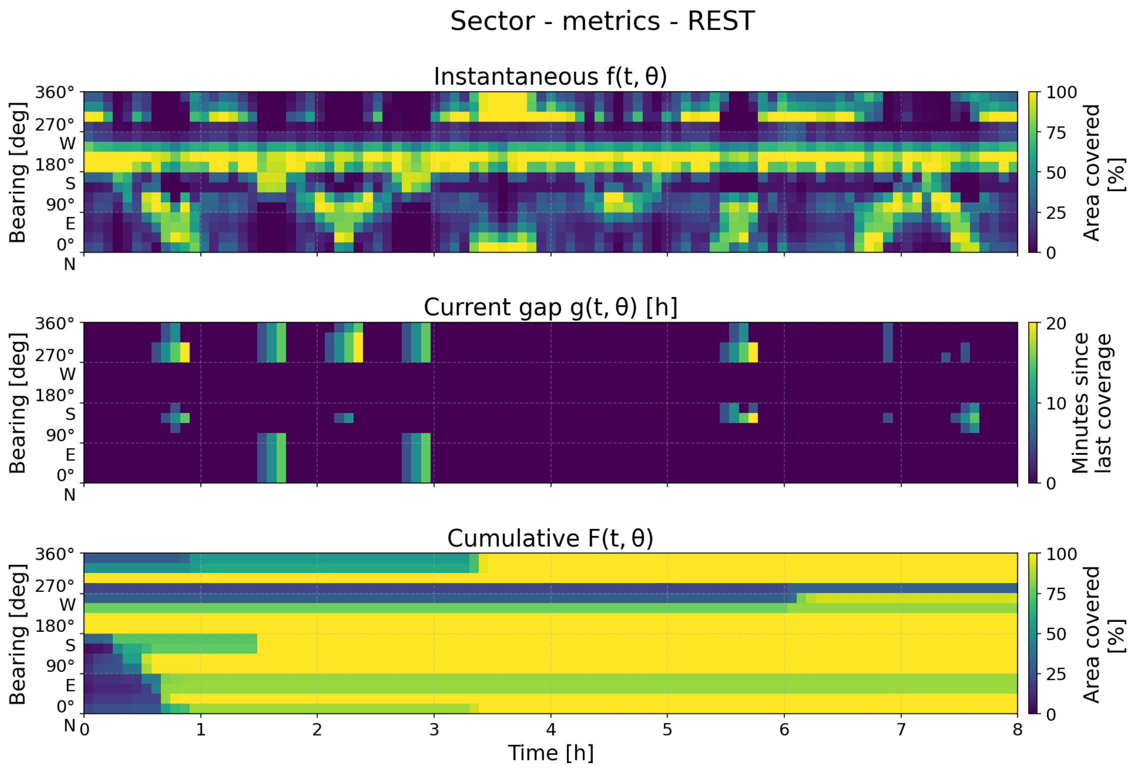


Figure A.21: Sector-wise metrics for the REST zone in validation. Top: $f(t, \theta)$; middle: $g(t, \theta)$ (minutes); bottom: $F(t, \theta)$. Periodic 20-minute gaps appear at $t \approx 1$, 2.5, and 5.8 h in northwestern sectors, with minor southern-sector interruptions.

Overall, despite limited training scenarios, the model effectively generalizes somewhat to new configurations, closely adhering to the predefined objective function criteria. This demonstrates the model's and methods capacity to extend learned solutions effectively into untrained, novel scenarios.

CHALMERS
UNIVERSITY OF TECHNOLOGY