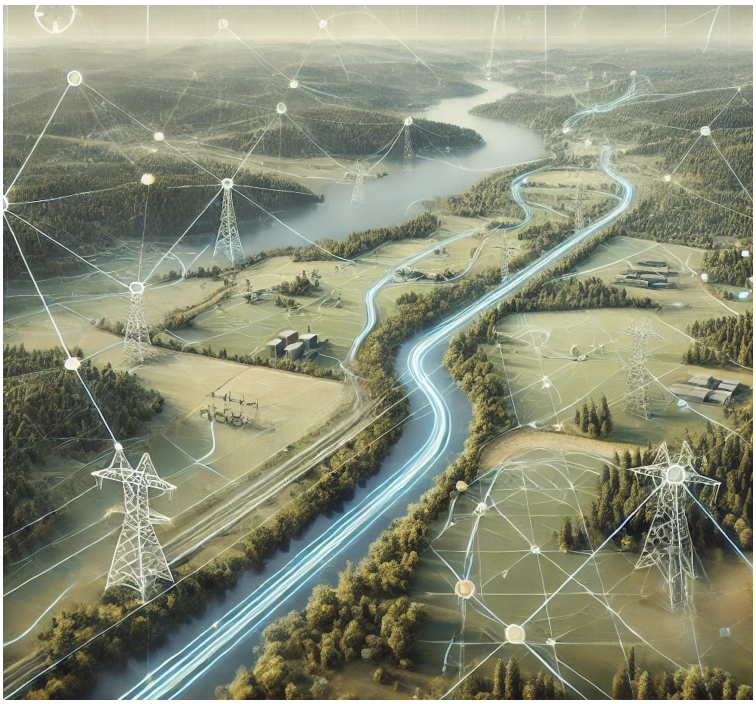


Optimizing Power Cable Routing using AI

Applying Min-Cost-Path Models on GIS Data For Projecting an Optimal Cable Route

Master's thesis in Master Program Complex Adaptive Systems

JOHAN LARSSON
OLIVER DE ROSA

DEPARTMENT OF PHYSICS

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2025
www.chalmers.se

Master's thesis 2025

Optimizing Power Cable Routing using AI

Applying Min-Cost-Path Models on GIS Data For Projecting an Optimal Cable Route

JOHAN LARSSON
OLIVER DE ROSA

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Physics
Chalmers University of Technology
Gothenburg, Sweden 2025

Optimizing Power Cable Routing using AI
Applying Min-Cost-Path Models on GIS Data For Projecting an Optimal Cable Route
JOHAN LARSSON
OLIVER DE ROSA

Project Owner: Martin Skilbred, Head of WSP Power Systems Sweden
Project Supervisor: Josef Abriren, Employee WSP Power Systems Sweden
Examiner: Mats Granath, Director of Master Program Complex Adaptive Systems and Senior Lecturer at Department of Physics

Master's Thesis 2025
Department of Physics
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: An AI generated figure of a landscape with different electrical transmission infrastructures and terrain features. Generated by OpenAI's model ChatGPT 4o.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2025

Optimizing Power Cable Routing using AI
Applying Min-Cost-Path Models on GIS Data For Projecting an Optimal Cable Route
JOHAN LARSSON
OLIVER DE ROSA
Department of Physics
Chalmers University of Technology

Abstract

Planning underground transmission cable routes is traditionally a labor-intensive and expert-driven task involving multiple, often conflicting, objectives. To meet increasing energy demands, scalable and intelligent methods are needed to enhance efficiency while accounting for diverse spatial constraints. This study investigated how AI-driven optimization algorithms can improve underground cable routing by integrating them with Geographic Information Systems (GIS) data. Key geospatial constraints, such as soil type, land use, and vegetation, were identified and systematically encoded into a unified cost surface using the Analytic Hierarchy Process (AHP). Three algorithmic approaches were implemented and evaluated: Dijkstra's algorithm (including single- and bi-objective variants), the A* algorithm with weighted heuristics, and a bi-objective Ant Colony Optimization (ACO) algorithm. These were assessed across synthetic and real-world environments, including the planned Gotland link in Sweden. While each model exhibited distinct strengths and limitations, all produced competitive and adaptable routes compared to manual planning, effectively balancing traversal cost and distance, and identifying Pareto-optimal solutions that highlighted strategically important areas. This suggests that the integration of AI and GIS has strong potential to automate and improve cable routing processes. At the same time, the study underscores the importance of tuning both model architecture and environmental representation to maximize real-world applicability, revealing that model performance is not only algorithm-dependent but also highly sensitive to the spatial structure and scaling of the input data.

Keywords: underground cable routing, GIS, AI, (bi-objective) optimization, AHP, Dijkstra algorithm, A* algorithm, Ant Colony Optimization, Pareto solutions, infrastructure planning

Acknowledgements

We want to express our gratitude to our supervisors at WSP, Martin Skillbred and Josef Abriren, and the rest of the team, both in Stockholm and Gothenburg, for giving us the opportunity to explore the exciting field of cable routing. We are especially thankful for the support, insights, and hospitality shown by everyone at WSP throughout the project. Access to the knowledge and experience within such a large organization have been invaluable.

We also extend our thanks to our examiner and academic supervisor at Chalmers, Mats Granath, for his valuable feedback, guidance, and encouragement during this thesis work.

Lastly, we wish to thank our families, partners and friends for their unwavering support throughout this spring, particularly during the more intense phases of the project. Their patience and encouragement have meant a great deal to us - and without it, completing this work would have been far more difficult.

Johan Larsson and Oliver De Rosa, Gothenburg, June 2025

Use of AI Tools During This Thesis

With the recent advancement of large language models, their capabilities in generating both text and code have significantly improved. Throughout this thesis work, models such as ChatGPT, Copilot, and Gemini have been used to support idea generation, revise academic writing, and assist in developing code pipelines. Their outputs have always been reviewed, as these models are still prone to hallucinations and factual inaccuracies. The use of such tools have served as a complement rather than a replacement for rigorous evaluation, domain understanding, and manual verification.

List of Acronyms

Below is the list of acronyms used throughout this thesis listed in alphabetical order:

A*	A* Search Algorithm
AC	Alternating Current
ACO	Ant Colony Optimization
AE	Autoencoder
AI	Artificial Intelligence
AHP	Analytic Hierarchy Process
BSP	Biobjective shortest path
CI	Consistency Index
CNN	Convolutional Neural Network
CR	Consistency Ratio
C-D3QN	Convergent Dueling Double Deep Q-Network
DDQN	Double Deep Q-Network
DNN	Deep Neural Network
GA	Genetic Algorithm
GDB	Geodatabase
GeoAI	Geospatial artificial intelligence
GIS	Geographic Information System
LCP	Least-Cost Path
LP	Linear Programming
MCDA	Multi-Criteria Decision Analysis
MILP	Mixed Integer Linear Programming
MSP	Multiobjective Shortest Path
NVV	Naturvårdsverket
PCA	Principal Component Analysis
PER	Prioritized Experience Replay

PSO	Particle Swarm Optimization
RI	Random Index
SGU	Sveriges Geologiska Undersökning
SkS	Skogsstyrelsen
SMHI	Sveriges Meteorologiska och Hydrologiska Institut
STP	Steiner Tree Problem
SVK	Svenska Kraftnät
TSP	Traveling Salesman Problem
VNS	Variable Neighborhood Search
WMS	Web Map Service

Contents

List of Figures

List of Tables

1

Introduction

Transmission line planning and cable routing is today a very iterative task [1], where much of the work is manual - the main reason to why this master thesis is conducted. It requires a lot of time and material costs, as it relies mostly on experienced professionals through fields surveys, map data and repetitive experiences [2]. Hence, the outcomes are largely dependent on the expertise and experience of personnel. As energy demand continues to grow, expanding the existing transmission network becomes essential, underscoring the increasing importance of effective transmission network planning and optimization [3]. A valuable tool towards this is Geographic Information Systems, or GIS, which is widely used as it is capable of gathering and analyzing regional data, such as topography, climate conditions, and land use classifications, providing valuable support for tasks like site selection and route optimization. The cable routing team at WSP currently uses GIS as one of their main workflow tools. However, it lacks inherent intelligence and depends on human input for leadership and decision-making. Hence, in recent years, Artificial Intelligence (AI) Geographic Information Systems, *AI-GIS*, has become an important research topic [4]. AI-GIS consist of three components:

(1) **Geospatial artificial intelligence (GeoAI)**, which refers to the integration of AI techniques, such as machine learning and deep learning, into geospatial data processing and analysis. It focuses on solving geospatial problems using AI methods.

(2) **AI for GIS**, which involves applying artificial intelligence to improve the functionality, efficiency, and user experience of GIS software and tools.

(3) **GIS for AI**, focusing on leveraging GIS tools and capabilities to support artificial intelligence models, particularly in visualizing, analyzing, and interpreting AI-generated outputs.

Optimizing the cable routing task, through the use of AI models, is therefore categorized as an GeoAI project, and hence part of a global AI geospatial analytics market projected to reach $172 million by 2026 [5]. This highlights the interests, investments and, consequently, the significance for those active in the field, such as consultants like WSP, to implement AI-GIS in their workflow and stay updated

with advancements within the area. It does not only reduce lead times, costs and recourse usage, making it both a business- and environmental-beneficial workflow, it also reduces the risk of human error, thereby enhancing overall efficiency and reliability [6].

For WSP, one of the world's leading engineering and professional consultant firms active within the area, a step towards this is taken by organizing and supervising this master thesis, which aims to automate the cable routing work process by integrating AI models to the GIS data processing in order to find an optimal path given multiple constraints, mainly minimizing distance and construction cost.

1.1 Project Description

Here we describe the project purpose, objectives and limitations as well as the stakeholders of the project.

1.1.1 Project Purpose

Originally initiated to develop a model for generating optimal transmission cable routes, this project - conducted in collaboration with the Power Systems department at WSP - has evolved into a broader exploration of a specific optimization method and its practical application within infrastructure planning. Rather than focusing solely on producing an optimal path, the project now emphasizes defining the methodological framework, illustrating its application in a real-world context, and critically assessing its limitations and suitability. The study includes a review of existing techniques in transmission line routing and optimization, followed by the implementation of a selected approach to demonstrate its potential. The resulting model serves as a decision-support tool rather than a final design solution. Any proposed route generated by the system is intended as a baseline for further expert analysis and revision, ensuring that project-specific requirements, stakeholder priorities, and practical constraints are fully addressed.

1.1.2 Project Formulation

The problem is formulated as a Multiobjective Shortest Path (MSP) problem, where key objectives include minimizing path length, accounting for multiple terrain factors, and adhering to bend radius limitations, along with other task-specific constraints. The project focuses on the routing of underground transmission cable, more specifically across a part of the east coast mainland and the island of Gotland in Sweden. Terrain factors are represented in a cost matrix or a total weighted surface, which models the environment through which the agents will navigate. Hence, the problem can be reduced to a biobjective shortest path (BSP) problem, with a trade-off between traversal cost and path length. Therefore, an optimal path refers to a Pareto-optimal solution; one where no objective can be improved without worsening the other. The environments are raster-based, where the neighbors of each

cell are considered through the Moore neighborhood, as seen in Figure 1.1. The route is evaluated between a fixed starting point and a fixed endpoint.

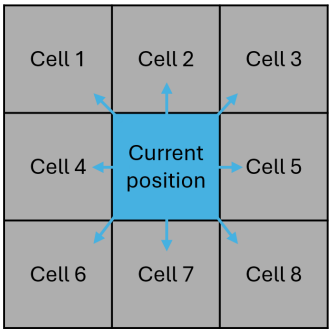


Figure 1.1: Illustration of the Moore neighborhood in a grid-based environment. The agent (blue cell) is surrounded by eight adjacent cells (horizontal, vertical, and diagonal neighbors) representing all possible movement directions from the current position. This structure is commonly used in pathfinding and cellular automata to define local connectivity.

1.1.3 Research Questions

The following are the research questions of this project:

- How can AI-driven optimization methods improve the efficiency of underground cable routing compared to traditional approaches?
- Which key factors/constraints can be used in route optimization?
- Which machine learning or optimization algorithms are best suited for optimizing cable routing while considering multiple constraints?
- How can one quantify the influence of model architecture versus environment setup on the final routing outcome, and to what extent does each component contribute to overall performance?

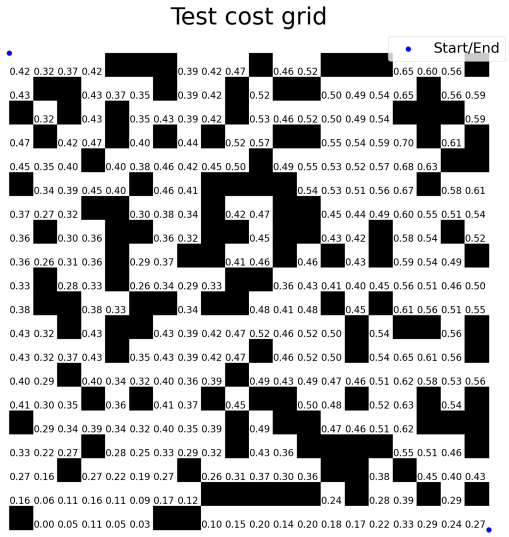
1.1.4 Testing Grounds

1.1.4.1 Initial Testing Ground

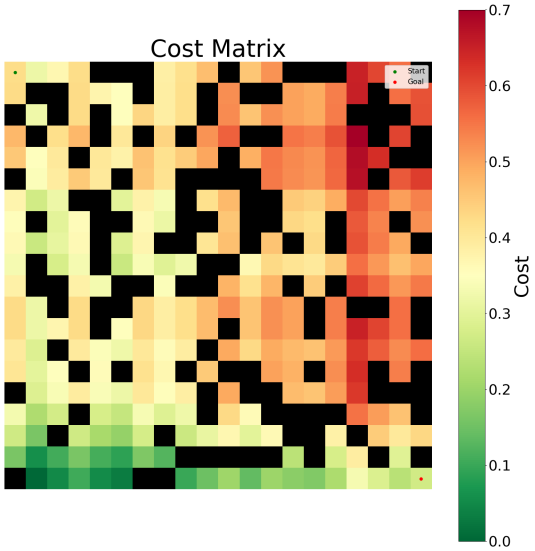
The initial model testing environment consists of a 20×20 grid, representing a simplified scenario for evaluating the algorithms' performance. The grid contains non-traversable (black) and traversable (white) cells (i, j), each randomly assigned a cost value $c_{i,j}$ in a range consistent of that of the real world testing grounds, closely mimicking the cost distributions found in real-world applications. More specifically, $c_{i,j} \in [0, 0.7]$. To generate a more realistic cost landscape, Gaussian smoothing is applied to the grid. This operation ensures a smooth transition of cost values across the grid, enabling analysis of whether the models effectively follow low-cost regions

in the environment. It also reduces abrupt cost changes - an uncommon feature in real-world routing scenarios, where such transitions are typically represented as discrete obstacles - thereby enhancing both the realism and practical applicability of the results.

This test environment serves as an initial evaluation setup to assess the model's ability to navigate complex terrains while considering cost constraints. The start and goal positions are fixed at the top-left and bottom-right corners, respectively, ensuring a consistent pathfinding scenario.



(a) Initial testing ground cost matrix, with black regions denoting obstacles and the numbers representing individual cell cost values.

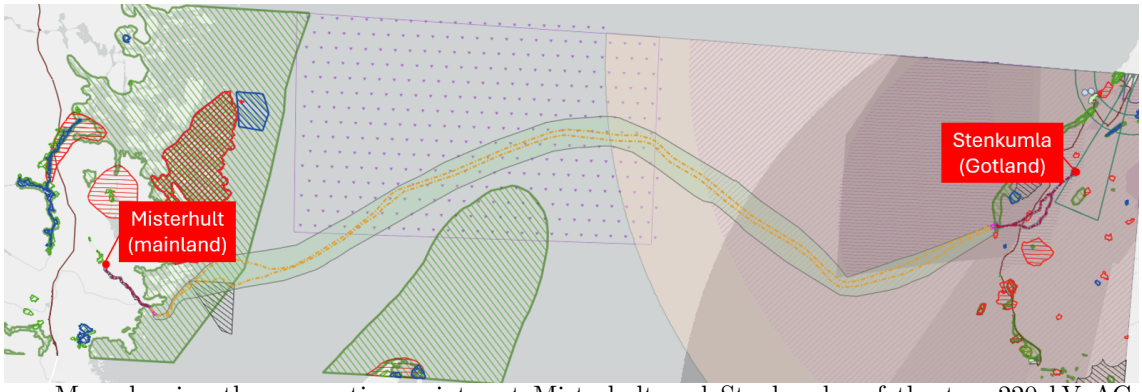


(b) Colormap visualization of the initial testing ground cost matrix.

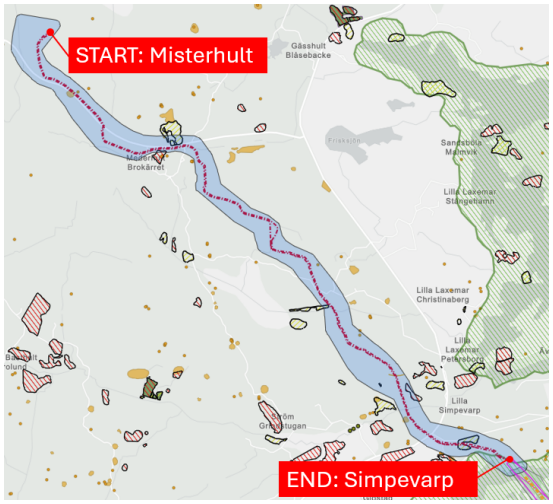
Figure 1.2: Initial testing ground grid of size 20×20 cells, with costs in the range $[0, 0.7]$, generated using Gaussian smoothing to create a more realistic routing cost landscape.

1.1.4.2 Real World Application Testing Ground

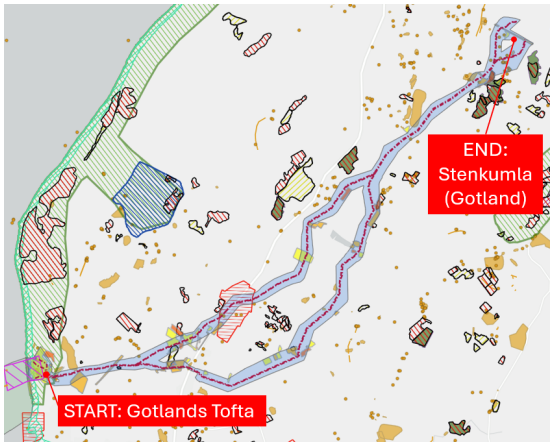
The real application testing environment for the project is the Gotland link (Gotlandsförbindelsen) [7]. This is a critical infrastructure project undertaken by Svenska Kraftnät (SVK) to enhance the electricity supply to the island of Gotland. The project involves the construction of two 220 kV alternating current (AC) cables connecting the Swedish mainland to Gotland. The two cables will link the mainland at Misterhult in Oskarshamn Municipality to Stenkumla, situated just south of Visby on Gotland, as seen in Figure 1.3(a). The model's performance is evaluated on both the mainland and Gotland route. This is between Misterhult and Simpevarp on the mainland, and Gotlands Tofta and Stenkumla on Gotland. Both routes are seen in Figure 1.3(b) Like the initial model testing environment, these areas are represented as rasters. Subgrids of these environments will also be used for analyzing model performance.



(a) Map showing the connection points, at Misterhult and Stenkumla, of the two 220 kV AC cables planned to connect Gotland to the mainland, a project called Gotland link, or *Gotlandsförbindelsen*.



The testing ground on the mainland, between Misterhult and Simpevarp. The blue area indicates the planned cable corridor, whereas the red-dotted line is the more exact planned path of the two 220 kV cables.



The testing ground on Gotland, between Gotlands Tofta and Stenkumla. The blue area indicates the planned cable corridor, whereas the red-dotted line is the more exact planned path of the two 220 kV cables.

(b)

Figure 1.3: Visualization of the Gotland link (*Gotlandsförbindelsen*) project: (a) the connection points at Misterhult and Stenkumla, and (b) the testing ground on the mainland and Gotland. Maps taken from [8].

1.1.5 Project Objectives

The thesis project is structured into four stages, each refining the models, cost matrix, and data layers to improve routing optimization. The stages are outlined in Table 1.1. Before reading the staged work-plan, note that the layer-weighting technique (Analytic Hierarchy Process, AHP) and the optimization models mentioned

below are motivated theoretically in Section 2 and implemented in detail in Section 3. The purpose of this section is to provide a concise overview of the project's structure.

Table 1.1: Project objectives structured in four stages, from model development to real-world application. The theory and method of the different approaches and models are described in Section 2 and 3, respectively.

Stage	Description
Stage 1	**Initial Model Development and Environment Data Processing** The first stage focuses on developing and validating the Dijkstra, A* and ACO algorithms, while also processing and initializing the real-world GIS data for the mainland testing ground. • Develop Dijkstra, A* and ACO models and test on initial testing ground. • Implement improvements to the models and rerun on random grids. • Analyze the mainland testing ground data and rasterize. • Analyze AHP weights for the raster layers. • Construct the cost matrix using the raster layers and AHP weights. • Analyze results to identify potential refinements.
Stage 2	**Run Models on Subgrids and Add More Data** Building upon insights from Stage 1, more data layers are added for the cost matrix and subgrids of the mainland environment are extracted for further evaluation. • Improve the models and refine the cost matrix using insights from Stage 1. • Integrate additional data layers into the cost matrix. • Extract subgrids of the mainland environment. • Analyze the performance of the improved models, comparing models on the mainland subgrids.
Stage 3	**Scaling to Multiple Testing Grounds (Mainland and Gotland)** Further refinements are made based on Stage 2, with expanded testing across both the mainland and Gotland testing grounds. • Further refine the models and cost matrix based on findings from Stage 2. • Expand testing by applying the method the Gotland testing ground. • Evaluate the performance of the models across the the different runs and locations.
Stage 4	**Final Evaluation** The final stage focuses on applying the refined models and conducting a comprehensive evaluation. • Implement final refinements based on insights from previous results. • Validate the models by testing them on selected current and/or past projects, assessing their performance and overall effectiveness. • Answer research questions.

1.1.6 Limitations

To ensure the project remains feasible and the resulting model is user-friendly, the following limitations are defined:

(1) The analysis focuses on a specific geographic region provided by WSP, ensuring manageable data sizes.

(2) Only publicly available GIS data and WSP-provided datasets are used to avoid licensing complexities, hence limiting both the number of testing grounds available for evaluation and their included parameters.

(3) Not all input parameters are considered simultaneously in the beginning of the project, but added one by one given time constraints, feasibility, convertibility to appropriate data format and feedback from the WSP team.

(4) The cost matrix construction through the AHP model (see Section 3.1) is based on previously used values from the literature, instead of local interviews with experts within WSP, since this is very time consuming and deemed outside the scope of this project.

1.1.7 Stakeholders

The project has three primary stakeholders: the authors, Chalmers University of Technology, and WSP.

1.1.7.1 Master Thesis Authors

The authors, Johan Larsson and Oliver De Rosa, are master's students enrolled in the Master Programe Complex Adaptive Systems at Chalmers University of Technology. This thesis is the culmination of their academic journey, applying advanced knowledge and skills to address challenges in electrical power systems under the supervision of WSP's Power Systems team.

1.1.7.2 Chalmers University of Technology

Chalmers University of Technology, located in Gothenburg, Sweden, is renowned for its emphasis on science, technology, and innovation. Mats Granath, director of Master Program Complex Adaptive Systems and senior lecturer at Department of Physics, oversees this thesis, providing academic guidance and ensuring the research aligns with the university's standards of excellence.

1.1.7.3 WSP

WSP is a global leader in engineering and professional services, with over 73,900 employees, delivering innovative solutions in the energy, transportation, infrastructure, and environmental sectors. The Power Systems team at WSP specializes in solving power supply challenges for their clients. As an industrial partner, WSP provides essential expertise and insights, ensuring the thesis research addresses practical, real-world challenges in electrical power systems.

1.1.8 Ethical Considerations

The use of AI in transmission cable routing offers clear advantages in efficiency and precision, but also raises important ethical concerns. A key issue is transparency; AI models, especially those based on reinforcement learning or evolutionary algorithms, can be difficult to interpret, making it unclear how specific routing decisions are made [9]. To maintain both public and customer trust for actors like WSP, and

facilitate regulatory processes, it is essential to ensure that model assumptions and outcomes are well-documented and explainable.

Accountability is another concern. As AI increasingly influences decisions in infrastructure planning, responsibility for outcomes - particularly in the case of errors, land-use conflicts, or environmental impact - must remain clearly defined [10].

Moreover, AI-based routing inherently involves value-based trade-offs, such as balancing cost, environmental impact, and social constraints. These trade-offs must be made explicit and include stakeholder perspectives to ensure legitimacy and fairness [11].

Finally, while AI can help avoid ecologically sensitive areas, its use should not justify technically optimal but ethically questionable developments. Responsible AI deployment must align with sustainability goals and uphold environmental standards.

1.1.9 Sustainability Considerations

Sustainable infrastructure is essential for supporting society's transition to a low-carbon, resilient future. As energy systems expand to accommodate renewable sources and electrification, the planning of transmission cables becomes increasingly important; not only for technical efficiency but also for minimizing environmental and societal impact. By enabling automated, cost-sensitive, and constraint-aware routing, AI reduces the need for manual trial-and-error methods and facilitates smarter use of space and resources. This aligns with many of UN's *Sustainable Development Goals* (SDGs), e.g. SDG 7 (Affordable and Clean Energy) by improving the deployment of transmission networks that integrate renewables, and with SDG 9 (Industry, Innovation and Infrastructure), through the advancement of intelligent, scalable design methods [12, 13].

Furthermore, by incorporating environmental and land-use constraints into the routing model, such as avoiding forested areas, steep terrain, and water bodies, the project supports SDG 13 (Climate Action) and SDG 15 (Life on Land). AI-driven routing can reduce construction impact, fragmentation of ecosystems, and unnecessary excavation, thereby contributing to more sustainable land management [14]. It is important, however, to consider the high energy consumption associated with training large AI models - an increasingly relevant concern as AI pipelines scale.

In summary, using AI for cable routing is not only a technical innovation but also a tool for addressing key sustainability goals, by enabling faster, smarter, and more environmentally responsible infrastructure development.

2

Literature Review

This review explores established and emerging approaches to cable and transmission line routing, drawing upon existing literature and similar projects. The goal is to identify key methodologies, understand their strengths and limitations, and thereby establish a foundation for the methods employed in this thesis. We begin by outlining traditional routing practices and the challenges that motivate the adoption of computational methods, followed by an examination of GIS and AI integration, current techniques including cost surface modeling and optimization algorithms, and a review of specific studies in the field. Finally, particular attention will then turn to the various optimization algorithms and the AHP, leading to the selection of an appropriate method for this project.

2.0.1 Background to Cable/Transmission Line Routing

As previously mentioned, cable routing optimization is traditionally a labor-intensive and iterative process, heavily reliant on expert knowledge and manual adjustments. These conventional approaches typically integrate data from field surveys, maps, and past project experience to determine feasible routes [15].

A central tool in facilitating this process is GIS, which is widely used within transmission line routing [15, 16]. GIS platforms are employed to layer and visualize various spatial data types, encompassing terrain characteristics, existing infrastructure, land ownership, environmental protection zones, and geological data. This enables planners to perform initial pathfinding, identify significant obstacles, and assess the suitability of different corridors. The process often involves manually drawing potential route segments within the GIS environment and then iteratively refining them based on newly acquired information or re-evaluation of constraints.

While these conventional approaches have demonstrated their utility in successfully completing many infrastructure projects, it is important to recognize, as previously suggested, that they are accompanied by limitations. The reliance on manual interventions and the iterative cycle of design, review, and modification inherently render these approaches time-consuming. Each iteration, from data collection to GIS-based analysis and expert review, can extend project timelines considerably. Consequently, this extended effort translates into higher project costs, encompassing labor, resource allocation, and potential delays. Furthermore, despite the best

efforts and expertise of the personnel involved, the manual nature of the process makes it susceptible to human error in data interpretation, judgment, or transcription [1, 16]. The increasing complexity of modern infrastructure projects, coupled with the growing demand for energy, highlights the need for scalable, sustainable, and intelligent planning methods for power grids and related infrastructure.

To address these limitations, the integration of AI with GIS has gained significant attention. GIS provides a structured platform for managing and analyzing spatial data, while AI contributes advanced optimization and decision-support capabilities [17]. AI-GIS applications have been explored in areas such as soil erosion prediction, urban land-use planning, and logistics site selection, using methods like neural networks, genetic algorithms, and swarm intelligence. However, as [17] notes, a standardized and cross-domain methodology for AI-GIS integration is still lacking, underscoring the need for further development.

While AI-GIS applications are diverse, the most found common approach in infrastructure routing involves transforming varied geospatial data, including terrain, land cover, environmental restrictions, and costs, into composite cost or weight surfaces. This is frequently achieved using Multi-Criteria Decision Analysis (MCDA) techniques, among which the Analytic Hierarchy Process (AHP, see Section 3.1) is notably prevalent for structuring criteria and assigning weights [16, 18, 19, 20, 2, 21]. These cost surfaces convert multidimensional spatial information into a unified scalar value per grid cell, enabling quantitative comparison of potential routes.

A primary benefit of this cost surface model is its compatibility with established Least-Cost Path (LCP) algorithms, like Dijkstra's algorithm, which efficiently find the optimal path based on minimizing cumulative cost [18, 22]. However, this represents optimization for a single objective (the composite cost). Real-world routing is inherently a more intricate, multi-objective challenge, requiring simultaneous consideration of factors like total length, specific costs, environmental impacts, and technical constraints [23, 2]. Simple LCP algorithms on a single cost surface struggle to inherently balance these competing objectives to find Pareto-optimal solutions.

Consequently, much research focuses on applying more sophisticated optimization algorithms to navigate these cost surfaces or directly handle multiple objectives. This includes heuristic search algorithms like A*, metaheuristics like Ant Colony Optimization (ACO) and Genetic Algorithms (GA), mathematical programming techniques like Linear Programming (LP), and machine learning approaches such as Reinforcement Learning (RL). Despite the algorithmic sophistication, the application often remains centered on pathfinding within the predefined cost-grid framework and additional constraints are implemented within the models themselves. This algorithmic focus could be argued to overlook the holistic modeling of complex planning environments as the defined static and weighted surface layer may oversimplify the underlying spatial and regulatory realities [24, 17].

This prevailing focus on algorithmic optimization within defined cost grids, often utilizing methods like ACO [23, 16, 15, 24, 19, 17, 23], forms the immediate context

for the present work. The utility of these techniques for efficient pathfinding is evident, the identified limitations, particularly the potential disconnect from the broader complexities, motivate further investigation. Therefore, this thesis positions itself within this specific landscape. It aims to understand and build upon the established foundation of algorithmic optimization on cost surfaces, exploring how such techniques perform and can potentially be adapted when applied within a GIS-based framework designed to incorporate a wider spectrum of practical routing considerations.

To contextualize and compare current advancements in the field, Table 2.1 below presents a selection of studies where most explore the integration of AI techniques with GIS based modeling for transmission line routing. Each entry outlines the study's inferred research focus, methodological approach, modeling environment, and key findings. These works reflect a wide range of techniques, from ACO to deep reinforcement learning, and demonstrate the growing interest in combining intelligent algorithms with spatial planning tools to improve routing efficiency, realism, and adaptability. In particular, AHP appears repeatedly in multiple studies, underscoring its relevance and versatility in MCDA. This method is further discussed below.

Table 2.1: A selections of previous studies adhering to the field to transmission line path routing.

No.	Title (year)	Inferred Research Focus	Methods Used	Test Environment	Key Findings
[17]	Transmission Line Planning Based on AI in Smart Cities (2022)	How can AI algorithms and GIS be integrated? Can neural networks provide objective cost modeling? How does AI compare to Dijkstra/manual routing?	BP Neural Network, Ant Colony Optimization (ACO), Dijkstra	50m x 50m raster cells, classified terrain grid	AI with ArcGIS and BP-NN improves routing quality; real-world deployment still limited but can aid in path planning.
[16]	Transmission line route planning based on AHP-ACO algorithm (2022)	How can AHP quantify constraints? Can AHP+ACO improve realism in planning? How effective in complex terrain?	Analytic Hierarchy Process (AHP), ACO in Python and ArcGIS	Raster representation, layers: elevation, slope, roads, reserves etc...	Efficient, scalable planning with improved realism and decision support.
[15]	Research on transmission line path planning model based on TFN-AHP and ACO (2024)	How can TFN-AHP improve weighting? Can enhanced ACO improve routing? Efficiency vs. traditional ACO?	TFN-AHP, Improved ACO, ArcGIS + MATLAB	Raster representation, binary masks for avoidance	Improved ACO is faster/-more efficient. + TFN-AHP helps reduce ambiguity and reduce dimensions.
[18]	Least-cost path analysis and multi-criteria assessment for routing electricity transmission lines (2016)	Can GIS + AHP offer a better routing method? AHP vs. monetary cost models? Advantages of AHP?	AHP, Monetary cost weighting (using database), Dijkstra in QGIS	Raster cost surface, One by AHP and one economic layer through monetary values	AHP effective but lacks financial nuance; monetary weights more realistic.
[19]	Optimization of high voltage transmission line path based on AI (2022)	Can AI automate HV routing? Can it match manual methods?	AHP, Cellular Automata (CA), ACO	Raster grid, cell types for tower placement feasibility	AI models effective for route finding; realism limited by physical constraints. Has guiding significance for manual route selection.
[20]	Transmission Line Routing Using GIS Tools of Spatial Sciences (2020)	Can GIS-based methods replace manual planning? How effective is AHP + A*? Role of expert input?	AHP, A* algorithm (modified Dijkstra), Python	Raster layers created from Sentinel-2 satellite images, slope data, and OpenStreetMap inputs.	Scalable, objective routing using A* + AHP; limited by raster resolution.

No.	Title (year)	Inferred Research Focus	Methods Used	Test Environment	Key Findings
[2]	Automatic optimization model of transmission line based on GIS and genetic algorithm (2023)	Can GA + GIS find most reasonable path? Performance under multi-factor conditions? GA vs. Dijkstra?	Genetic Algorithm (GA), AHP, ArcGIS and (C#)	Raster grid, Routing conducted under two conditions: traffic-only vs. multi-factor environment including terrain, land use, disasters, protected zones, etc.	GA model outperforms Dijkstra; handles avoidance zones better.
[25]	The Research of Path Planning for Transmission Network Based on Improved Ant Colony Algorithms (2019)	Can ACO convergence be improved? Simplified parameter tuning? Effectiveness for static/multi-stage plans?	Improved ACO (IACO), static + multi-stage models	Simulated on benchmark systems: Garver 6-node, 18-node, and IEEE 24-node networks. Evaluation includes investment cost and overload penalties.	By using constant pheromone limits to decouple parameters, the proposed Improved ACO (IACO) achieves much faster computation speeds than standard ACO for transmission network planning problems.
[26]	A Comparation Study of Transmission Line Routing Based on A* and RRT Algorithms (2024)	How do A* and RRT compare? Obstacle handling? Improvement over manual routing?	A* (grid-based), RRT (random sampling using nodes of random tree)	Simulated grid, raster for A*, continuous or discrete space for RRT	RRT better for dynamic, obstacle-rich scenarios; A* more rigid but requires recalculations in changing environments.
[21]	A study on TNB transmission line route sustainability and suitability using GIS-AHP (2012)	Can GIS-AHP define suitable TL routes? Criteria for siting TL towers and routing?	AHP to weight spatial criteria, GIS Suitability Model	Raster model with natural, social, technical layers	GIS-AHP integration provides a structured and transparent framework for transmission line planning.
[3]	Smart line planning method for power transmission based on D3QN-PER algorithm (2024)	Can deep reinforcement learning satisfy TL routing constraints? How to reflect construction constraints? Value of prioritized experience replay?	D3QN, Improved AHP (IAHP), prioritized experience replay (PER) and ACO	Segmented satellite images, raster model with forests, rivers and houses, 8-direction grid world	D3QN-PER gives shorter paths, fewer turns, better regulation adherence.
[22]	Revolutionizing energy infrastructure: Automated route planning for underground transmission lines in Phnom Penh (2024)	Which pathfinding method suits UTL? Role of traffic, restrictions? Best balance of cost and accuracy?	Dijkstra, A*, Bellman-Ford, Floyd-Warshall, BFS, MST, Improved Dijkstra, Highway Hierarchy	Urban graph, traffic volume and constraints	A*, HH most efficient; strong adaptability to real-time data.
[24]	A Transmission Line Planning Method Based on Variable Size Grid and Improved Ant Colony Optimization Algorithm (2023)	Can variable grids improve ACO? Adaptation to terrain and costs? Effect of parallel/smooth search?	Improved ACO, FAHP, Parallel Ants, VSG	Raster GIS with adaptive segmentation (quad-tree)	Improved ACO and VSG improves fit, speed, and realism.
[23]	Optimization on shortest path finding for underground cable transmission lines routing using GIS (2014)	Can GIS + SACO automate UGC routing? SACO vs. Network Analyst? Custom weights vs. distance-only?	Simple ACO (SACO), Network of nodes	1m resolution network, Chennai India (urban study)	Custom SACO reduces cost/power loss, outperforms built-in tools.

2.0.2 Different Optimization Algorithms

As seen in the table, a variety of optimization algorithms have been applied to determine the most efficient routing paths for transmission lines, following the transformation of geospatial data into cost or weight layers. The selection of a particular algorithm often depends on the characteristics and constraints of the specific prob-

lem, as different models offer distinct advantages in addressing parameters such as terrain variation, obstacle handling, and multi-criteria trade-offs. Despite these differences, the primary objective in most studies remains the same: to identify the LCP between the start and end points, where cost reflects a combination of physical, environmental, and regulatory considerations.

2.0.2.1 Dijkstra's Algorithm

Dijkstra's algorithm is a classic graph-based method for computing the LCP from a source node to every other node by iteratively expanding the shortest-known routes. It uses a priority queue to explore nodes in order of increasing cost, updating distances until the optimal path is determined. Recognized as one of the most reliable shortest-path algorithms [27], it is widely employed in LCP applications for underground transmission line routing - selecting routes that avoid high-slope, forested, and landslide-prone areas while favoring low-cost terrain near roads [28]. Although the standard algorithm does not natively support multi-objective optimization, modifications exist to incorporate this feature [29, 30, 31]. In cases where heuristic guidance is crucial, the A* algorithm is often preferred [22]; alternatively, problems can be decomposed into subproblems - such as in linear programming models with relaxed constraints - that are solved incrementally using Dijkstra's approach [32].

2.0.2.2 A* Algorithm

The A* algorithm is a heuristic-based shortest-path algorithm, that extends Dijkstra's algorithm with a heuristic search parameter. It uses a cost function $f(n) = g(n) + h(n)$, where $g(n)$ is the cost from the start to the current node n, and $h(n)$ is a heuristic estimate of the cost from n to the goal. For Dijkstra, $h(n) = 0$. In the context of underground transmission line routing, A* has demonstrated versatility and precision by integrating constraints into the heuristic cost $g(n)$ like route length, traffic, and road restrictions [22]. Studies show it outperforms other algorithms such as Dijkstra's and Minimum Spanning Tree in urban settings [33, 34], while remaining simple to both understand and implement.

2.0.2.3 Linear Programming

Linear Programming (LP) is a mathematical optimization method used to find the best outcome (e.g., maximum profit or minimum cost) in a model whose requirements are represented by linear relationships. Mixed-Integer and Mixed-Binary Linear Programming (MILP/MBLP) extend LP by allowing some decision variables to take integer or binary values, enabling the modeling of more complex, discrete decisions. These methods have been widely applied to optimize cable layouts, particularly in wind farm projects [35, 36], but also in cable harness layouts in car compartments [32]. They aim to minimize costs and power losses while adhering to constraints such as cable capacity and terrain obstacles.

2.0.2.4 Ant Colony Optimization

Among the heuristic and metaheuristic techniques explored, Ant Colony Optimization (ACO) is one of the most frequently applied methods [23, 16, 15, 24, 19, 17, 23]. ACO mimics the natural behavior of ants, where artificial ants iteratively explore paths, guided by pheromone trails that reinforce optimal routes. This algorithm excels in flexibility, adaptability, and managing large solution spaces by integrating multiple parameters. Although hyperparameter tuning can be challenging, ACO effectively searches for paths that meet various requirements of power transmission projects, ultimately identifying the optimal route.

Recent studies have begun to explore ways to improve the adaptability and spatial accuracy of ACO type algorithms. For example, [24] introduce the use of variable-sized grid structures, which offer finer resolution in areas of high complexity and coarser resolution elsewhere, thereby improving computational efficiency without compromising route precision. In a related advancement, [17] propose an integrated approach in which a neural network is used to estimate the cost values of individual raster cells based on multiple geographical and environmental features within the planning area. The model is trained using standardized cost data sourced from the State Grid Corporation of China, specifically for 110 kV transmission lines under varying terrain and meteorological conditions.

2.0.2.5 Genetic Algorithm

Stochastic methods such as Genetic Algorithms (GAs) have proven effective for cable routing problems, leveraging natural selection-inspired mechanisms like mutation, crossover, and selection to optimize complex, multi-dimensional spaces. A GA-based approach for cable harness routing demonstrated better performance than some heuristic methods by minimizing routing costs and efficiently handling constraints like bundle length and wire count [37]. Similarly, combining GAs with a probabilistic roadmap, a sampling-based algorithm that creates a graph of feasible configurations to plan paths in high-dimensional and constrained environments, has enabled multi-branch harness designs, improving fitness values and computation times through adaptive mutation and crossover rates [38].

2.0.2.6 Particle Swarm Optimization

Particle Swarm Optimization (PSO), inspired by the social behavior of swarms, has also been applied to cable and pipe routing tasks. It is often used to optimize the Steiner Tree Problem (STP), i.e. minimizing the total length connecting a given set of points (terminals) by allowing additional intermediate points (Steiner points) to be added in order to reduce overall costs (see Figure 2.1). PSO has been effectively employed to approximate solutions to the STP by combining global search with domain-specific refinements. For instance, an improved Pattern Search PSO integrates local optimization to refine paths, achieving smoother layouts and reducing bends and costs in cable harness design [39]. Similarly, Multi-Objective PSO has been used for 3D cable layout optimization, balancing objectives like weight minimization and manufacturability through Pareto-optimal solutions [40]. Another ap-

plication combines PSO with Steiner minimal tree theory to optimize multi-terminal pipe routing, achieving efficient, collision-free layouts under complex constraints [41].

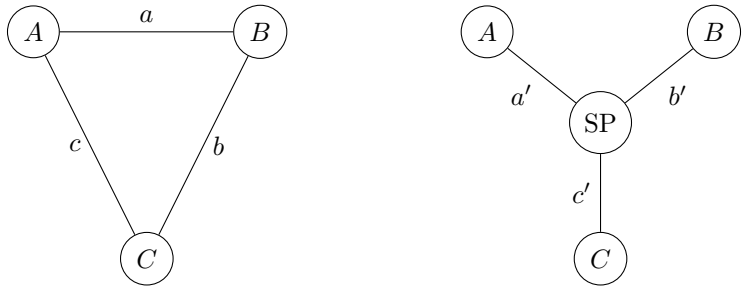


Figure 2.1: Comparison of a three-node network (left) versus the addition of a Steiner point (right), where $a' + b' + c' < a + b + c$.

2.0.2.7 Hybrid Models

Hybrid approaches that combine exact optimization methods with heuristics have also been explored to overcome the limitations of individual techniques. For example, combining MILP with heuristic methods like Variable Neighborhood Search (VNS) enables the generation of feasible solutions for large-scale problems while maintaining high solution quality [36].

2.0.2.8 Deep Q-learning

Reinforcement learning has recently been introduced as a promising technique for optimizing cable routing. One of its key strengths lies in its model-free learning capability, which allows it to dynamically adapt to its environment. This adaptability means the possibility of incorporating a variety of parameters for fine-tuning the system.

A Dueling Double Deep Q-Network with Prioritized Experience Replay (D3QN-PER) has been applied to power transmission planning, leveraging GIS to preprocess map data into grid-based constraints. This method has proven not only to generate smoother and more cost-effective paths, but to converge faster than a regular D3QN and with higher rewards compared to ACO [3]. The reward mechanisms in these models factor in various constraints, such as crossing costs, corner penalties, and adherence to environmental regulations, ensuring that solutions are practical and comply with planning requirements.

2.0.3 Data Analysis

2.0.3.1 Analytic Hierarchy Process

To improve route planning with AI, it is crucial not only to select an appropriate model, but also to translate the qualitative attributes of the data into quantifiable metrics for numerical optimization. This challenging task, due to its subjective nature, is addressed using the Analytic Hierarchy Process, an MCDA method that

organizes complex decisions into a hierarchical framework of criteria [42]. Considering the complexity of planning for an underground transmission line, AHP appears to be a popular and well-suited evaluation method when quantifying complex structural problems [15] and is often used in the found literature [16, 15, 18, 19, 20, 2, 21, 24]. In our case, this includes cost considerations as well as constraints like environmental impact and distance requirements. When combined with GIS raster data, AHP can quantify and integrate information across different raster layers.

2.1 Suitable Method for This Project

This project aims to explore and experiment with multiple optimization algorithms, with the selection and application of these methods being guided by their performance and results. A critical first step in this process is data preprocessing, which plays a foundational role in structuring the GIS data for optimization. The primary objective of this phase is to transform raw data into a structured format that can be effectively utilized by optimization algorithms. This involves quantifying obstacles and constraints, which is essential for accurately defining environment guiding the optimization process. For this step, the AHP is selected. The choice of AHP is, after discussions, supported by both the project team and external teams at WSP, who identified its potential relevance in this context.

For pathfinding and optimization, this project explores and compares the following algorithms from the literature, identified relevant for this project.

- **Standard Dijkstra's Algorithm:** Used as a benchmark to evaluate single-objective optimization results (based on cumulative cost and distance, respectively) against the more complex methods.
- **A* Algorithm:** Chosen for its efficiency in finding optimal paths on cost surfaces and its ability to incorporate heuristics, offering a strong baseline for path quality [22, 33].
- **Modified Bi-objective Dijkstra's Algorithm:** Selected specifically to address the multi-objective nature of the problem and identify Pareto-optimal solutions representing trade-offs between competing criteria (e.g., cost vs. length) [30, 31].
- **Ant Colony Optimization:** Implemented to explore its metaheuristic capabilities in searching complex solution spaces and finding paths considering multiple objectives biobjectively. Its adaptability and success in similar routing projects support its inclusion [23, 43, 15, 17, 44, 16, 24, 25].

A Convergent Dueling Double Deep Q-Network (C-D3QN) was explored during the early stages of the project due to its capacity to handle high-dimensional environments and learn optimal routing policies without the need for explicit programming. By integrating deep neural networks with reinforcement learning, this approach has previously shown strong performance in navigating complex search spaces while

balancing path efficiency and environmental constraints [3, 45]. However, due to significant differences between the environments used in this project and those in prior studies, as well as inferior performance compared to the selected baseline models, C-D3QN was not pursued further. Details of the work conducted can be found in Appendix A.2.

3

Theory

Here, the necessary theory for the projects different models and approaches is elaborated.

3.1 Analytic Hierarchy Process

The Analytic Hierarchy Process (AHP), a MCDA method for hierarchically organizing decisions, is in our case the base of the mechanism for the routing inside the grid given the different parameter layers. This approach establishes the foundation for developing the cost matrix. The criteria of the AHP process can be outlined as follows [21]:

(1) **Define the Problem and Goal:** Clearly establish the problem to be addressed and identify the desired outcome or solution.

(2) **Develop the Hierarchical Structure:** Construct a hierarchy that, as seen in Figure 3.1:
   - Begins with the overall goal at the top level.
   - Includes sub-criteria and criteria at intermediate levels.
   - Ends with possible alternatives or options at the lowest level.

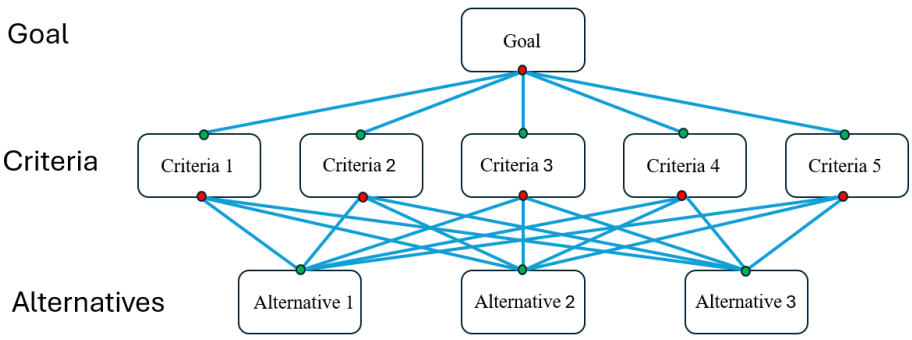


Figure 3.1: Analytic Hierarchy Process Model

(3) **Construct the Pairwise Comparison Matrix:** For each criterion, create a matrix that reflects the relative importance or contribution of each element with respect to the criterion above it. For example, with respect to criteria 1, how much more do you prefer alternative 1 to alternative 2?. These comparisons are based on the decision-maker's judgments about the relative importance of one element over another.

An example can be found in Table 3.1, which aims to help decide which phone to purchase among three options. While there are likely many criteria to consider for this decision, one important factor is the quality of the phone's camera. This table presents a comparison matrix for the cameras, where the rows indicate how much better one phone's camera is compared to another. For instance, the camera of Phone X is rated as twice as good as that of Phone Y, and six times better than that of Phone Z.

Table 3.1: Example of an AHP comparison matrix for comparing camera quality of different phones X, Y, Z ($n = 3$).

	Phone X	Phone Y	Phone Z
Phone X	1	2	6
Phone Y	$\frac{1}{2}$	1	3
Phone Z	$\frac{1}{6}$	$\frac{1}{3}$	1

(4) **Quantify the Judgments:** With n elements being compared, all are pairwise compared, resulting in $\frac{n(n-1)}{2}$ comparisons. The priority of the elements are set using the Saaty scale as seen in Table 3.2 below [46].

Table 3.2: The Saaty Scale for intensity of importance in the AHP.

Intensity of Importance	**Definition**	**Explanation**
1	Equal importance	Two factors contribute equally to the objective.
3	Moderate importance	Experience and judgment slightly favor one factor over the other.
5	Essential or strong importance	Experience and judgment strongly favor one factor over the other.
7	Very strong importance	Experience and judgment strongly favor one factor over the other. Its importance is demonstrated in practice.
9	Extreme importance	The evidence favoring one factor over the other is of the highest possible validity.
2, 4, 6, 8	Intermediate values	Used when compromise between two intensities is needed.

(5) **Compute the Eigenvalues and Perform Consistency Check:**

(a) Calculate the principal eigenvalue $\lambda_{\max}$ for the comparison matrix.

(b) Assess the consistency of the matrix using the Consistency Index (CI) and Consistency Ratio (CR):

$$\mathrm{CI} = \frac{\lambda_{\max} - n}{n-1}, \quad \mathrm{CR} = \frac{\mathrm{CI}}{\mathrm{RI}}$$

RI is the Random Index which depends on the dimension n of the comparison matrix [47]. See Table 3.3 below. If $\mathrm{CR} > 0.1$, judgments should be revised [28].

Table 3.3: Random Index RI for $n = 1, 2, \ldots, 15$ as used in previous projects.

n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
RI	0.00	0.00	0.58	0.90	1.12	1.24	1.32	1.41	1.45	1.49	1.51	1.48	1.56	1.57	1.58

(6) **Repeat Steps for Each Level:** Perform pairwise comparisons, eigenvalue calculations, and consistency checks for all levels in the hierarchy.

(7) **Synthesize the Judgments:** Compute the eigenvectors (normalized priorities) of each pairwise comparison matrix. These eigenvectors represent the weights of each element relative to the overall goal. Aggregate the priorities from the lowest level of the hierarchy to the top. Solve:

$$A\,\mathbf{w} \;=\; \lambda_{\max}\,\mathbf{w},$$

A is the pairwise comparison matrix and $\lambda_{\max}$ its largest eigenvalue. Normalize $\mathbf{w}$ so that $\sum_i w_i = 1$. The resulting *priority vector* $\mathbf{w}$ then gives the ratio-scale weights used to combine the spatial layers.

(8) **Validate Consistency of the Hierarchy:** Re-evaluate the overall consistency across the hierarchy. Again, if the consistency exceeds 0.1, refine the evaluation process.

With the AHP, one can assess the relative weights of multiple criteria in a more heuristic manner. The supervisors at WSP are not familiar with this method, but agreed, after insight and discussion, on the general pattern of the discovered weights and encouraged to continue with this procedure. This method is found to be the most appropriate evaluation to numerically quantify the factors for the subsequent optimization step.

3.2 Models

In this section, the selected models are explored, providing a comprehensive explanation of their structure, mathematical foundations, technical aspects, and how they are applied within the given environment.

3.2.1 Dijkstra's Algorithm

Dijkstra's algorithm is a fundamental method for finding the shortest path in a weighted graph $G = (V, E)$, where each edge $(u, v) \in E$ from vertex u to v has a nonnegative weight $w(u, v) \in \mathbb{R}^+$. The algorithm's objective is to determine the minimum-cost route from a designated source vertex $s \in V$ to all other vertices in the graph. In its classic form, Dijkstra's algorithm maintains a set S of vertices for which the minimal distance from s is definitively known, along with a priority queue Q that holds the remaining vertices keyed by their current best-known distance values. Starting with $d[s] = 0$ and $d[v] = \infty \ \forall v \neq s$, the algorithm repeatedly extracts the vertex u in Q with the smallest $d[u]$, then relaxes each edge (u, v) by checking whether:

$$d[v] > d[u] + w(u, v).$$

If a shorter path is found, $d[v]$ is updated accordingly:

$$d[v] = \min(d[v], d[u] + w(u, v))$$

The priority queue is adjusted to reflect this improvement. The procedure continues until Q is empty, ensuring that each vertex in V is assigned its optimal distance from s [48]. For a more detailed and visualized explanation, the authors recommend the example provided in [49].

In many real-world scenarios, however, a single objective does not capture all the factors affecting route selection. A multiobjective extension to Dijkstra's algorithm addresses this limitation by allowing each edge (u, v) to have N cost components

$\mathbf{d} = (d_1, d_2, \ldots, d_N)$. A biobjective example is illustrated in Figure 3.2, where edges connect a source s and a target t with dual-cost vectors. In this setting, the algorithm no longer tracks a single "best" distance for each vertex; instead, it associates each vertex v with a set of *labels* L_v, each representing a non-dominated path from s to v under the objectives [30, 31]. Given v has K labels, $l_v^k \in L_v$, $k \leq K$ stores the cumulative cost vector of the traversed vertices j from s to v for path k:

$$\mathbf{D}^{l_v^k} = \sum_{j=s}^{v-1} \mathbf{d}_{(j,\, j+1)}$$

Whenever a path is extended from u to a neighbor v along (u, v), a new label l_v^{k+1} is formed, for which $\mathbf{D}^{l_v^{k+1}} = \mathbf{D}^{l_u} + \mathbf{d}_{(u,\, v)}$. In particular, the new label is said to dominate the existing label if:

$$\mathbf{D}^{l_v^{k+1}} \leq \mathbf{D}^{l_v^k} \quad \text{and} \quad d_i^{l_v^{k+1}} < d_i^{l_v^k} \quad \text{for at least one } i \in \{1, \ldots, N\}$$

Any label $k = \{0, 1, \ldots, K\}$ that is dominated by the new label is removed from v's label set, and if the new label is dominated by any existing label, it is discarded. This process ensures that only Pareto-optimal routes remain. By combining this dominance check with a priority queue, typically ordered lexicographically by $(d_1, d_2, \ldots, d_N)$, the algorithm preserves the greedy spirit of Dijkstra's approach while effectively exploring the multi-dimensional outcome space to build a complete Pareto front of efficient solutions.

The Dijkstra algorithm, while optimal for single-objective shortest path problems, becomes computationally expensive in large-scale environments, especially when extended to multiobjective scenarios where each vertex may maintain multiple labels representing different trade-offs. *Pruning* is a key technique for reducing this computational burden by discarding partial paths that cannot lead to a Pareto-optimal solution. In our approach, a partial path is pruned if the sum of its accumulated cost and an optimistic lower-bound estimate for the remaining path is dominated by an existing complete solution in the online efficient set. Additional strategies, such as pruning based on the nadir point and label dominance, further limit the number of candidate labels maintained at each node, as explained in [30, 31]. Together, these methods concentrate the search on the most promising regions of the solution space, significantly decreasing the number of computations required without sacrificing optimality.

In the following, we refer to the multiobjective Dijkstra implementation as "Dijkstra MSP", while "Dijkstra Classic" denotes the classical Dijkstra's algorithm.

3.2.2 A* Algorithm

A* is a generalization of Dijkstra's algorithm that finds the shortest path by combining the actual cost from the start with an estimate of the remaining cost to the goal. In other words, A* minimizes the function

$$f(n) = g(n) + h(n),$$

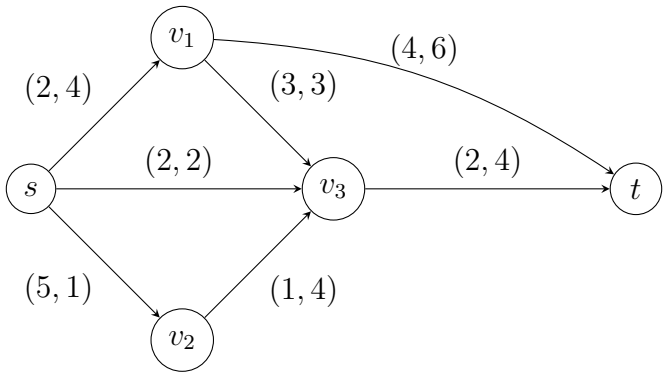


Figure 3.2: A graph with edges labeled by two costs (d_1, d_2), illustrating a biobjective scenario.

where $g(n)$ is the cumulative cost from the start node to the current node n and $h(n)$ is the heuristic estimate from n to the goal. Notably, if $h(n) = 0$ for every node, A* reduces to the classical Dijkstra algorithm.

The performance of A* hinges on the formulation of the heuristic function [50], which in this implementation is designed to incorporate both terrain cost information and geometric distance. The heuristic, found through trial-and-error, is computed as follows:

(1) The Euclidean distance d_{nE} between the current node n and the goal is calculated, and the corresponding traversal cost $C_{d_{nE}}$ is determined.

(2) Using the shortest path $\mathbf{P_S}$ and cheapest path $\mathbf{P_C}$ generated by the Dijkstra Classic algorithm, the position $(x, y)_S \in \mathbf{P_S}$ and $(x, y)_C \in \mathbf{P_C}$ are found that are at the same Euclidian distance from the goal as node n; $d_{(x,y)_S E} \approx d_{(x,y)_C E} \approx d_{nE}$.

(3) The remaining distance from $(x, y)_S$ to the goal of the shorest path $\mathbf{P_S}$ is calculated; $d_{\mathbf{P_S}((x,y)_S,E)}$. The corresponding is done for $(x, y)_C$ regarding $\mathbf{P_C}$, generating $C_{\mathbf{P_C}((x,y)_C,E)}$.

(4) These quantities are combined into a weighted sum:

$$h(n) = w_c \frac{C_{d_{nE}}}{C_{\mathbf{P_C}((x,y)_C,E)}} + w_p \frac{d_{nE}}{d_{\mathbf{P_S}((x,y)_S,E)}}$$

Here, w_c and w_p are weights satisfying $w_c + w_p = 1$.

This heuristic implies that when either

$$\frac{C_{d_{nE}}}{C_{\mathbf{P_C}((x,y)_C,E)}} \quad \text{or} \quad \frac{d_{nE}}{d_{\mathbf{P_S}((x,y)_S,E)}}$$

is large, the corresponding node is heavily penalized, thereby favoring expansions along paths that remain close to both the shortest and the cheapest routes. In theory,

the heuristic biases the search towards paths that simultaneously minimize travel distance and terrain cost, which enhances routing performance. By integrating this heuristic into the A* framework, the algorithm combines the exhaustive optimality of Dijkstra's method with the efficiency of greedy best-first search.

3.2.3 Ant Colony Optimization

Ant Colony Optimization (ACO) mimics the natural behavior of ants, where artificial ants iteratively explore paths, guided by pheromone trails that reinforce optimal routes. The algorithm treats cells as discrete spatial units with their corresponding costs. Like $g(n)$ does for A*, the heuristic value η_{ij} for ACO plays a crucial role in this process, as it represents the desirability of moving from cell i to cell j.

In traditional ACO applications, η_{ij} is often defined as the inverse of distance, guiding ants toward shorter paths. However, in this case, η_{ij} is adapted to reflect cost-based factors, ensuring that paths with lower construction costs and fewer constraints are prioritized. This modification allows the heuristic value to be more suited for transmission line routing.

3.2.3.1 Basic ant colony

The movement of an ant from cell i to cell j in the grid is governed by a probability $p_{ij}(t)$, which depends on the pheromone level $\tau_{ij}(t)$ and a problem-specific heuristic factor η_{ij}. In the basic ACO algorithm, it is inversely proportional to the distance from node i to node j, usually in order to solve the Traveling Salesman Problem (TSP) [51]. In this application, η_{ij} represents the cost of transitioning between cells, incorporating factors such as terrain, construction feasibility, and environmental impact. The transition probability is defined as [51]:

$$p_{ij}(t) = \begin{cases} \frac{\tau_{ij}^{\alpha}(t) \cdot \eta_{ij}^{\beta}}{\sum_{k \in \mathcal{N}_i} \tau_{ik}^{\alpha}(t) \cdot \eta_{ik}^{\beta}}, & \text{if } j \in \mathcal{N}_i, \\ 0, & \text{otherwise,} \end{cases}$$

The variables are as follows:

- $\tau_{ij}(t)$ is the pheromone level on the transition between cells i and j at time t.
- η_{ij} is the cost associated with moving from cell i to cell j.
- α and β control the influence of pheromone levels and costs, respectively.
- $\mathcal{N}_i$ is the set of neighboring cells of i.

After all ants have completed their paths, the pheromone levels are updated to reflect the quality of the solutions. The pheromone update rule combines evaporation and deposition:

$$\tau_{ij}(t+1) = (1-\rho) \cdot \tau_{ij}(t) + \Delta\tau_{ij},$$

Here, $\rho \in (0, 1]$ is the pheromone evaporation rate, which prevents excessive accumulation of pheromone and promotes exploration, while $\Delta\tau_{ij}$ is the total pheromone deposited by all ants during the iteration:

$$\Delta\tau_{ij} = \sum_{k=1}^{N} \Delta\tau_{ij}^{k},$$

$\Delta\tau_{ij}^{k} = \frac{Q}{L_k}$ if ant k uses the transition (i, j) in its solution, and 0 otherwise. Here, Q is a pheromone intensity constant and L_k is the total cost of the path taken by ant k.

3.2.3.2 Improved ant colony

Although basic ACO provides an intuitive and effective framework for path planning, its scalability suffers on larger, multi-objective problems; as the grid size and number of objectives increase, computation time grows and solution quality can decline due to the added complexity [43]. To address these challenges, several improvements from [43, 52] are implemented. These implementations are added to improve convergence, global & multi-objective search ability, and computational efficiency. They are described below:

(1) **Heuristic parameters:** As the original problem in [52] considers two objectives, two heuristic parameters are introduced, and therefore two pheromone parameters. This study adaptes this formulation to a raster-based grid format, rather than an arc-node mapping. Despite this change in structure, the core principle remains the same.

The first heuristic parameter, denoted by η_{ij}^{1k}, is derived from the cost matrix values and defined as:

$$\eta_{ij}^{1k} = \min\left(1, \left(\frac{c_{\max}^{k} - c_{ij}^{k}}{c_{\max}^{k} - c_{\min}^{k}} + \varepsilon\right)\right)$$

Here, $c_{\max}^{k}$ and $c_{\min}^{k}$ represent the maximum and minimum cost values across the cost matrix for objective k, respectively. The small positive constant ε ensures that the heuristic is non-zero, while the outer $\min(\cdot)$ function ensures it does not exceed 1.

In [52], multiple objectives are represented a cost vector (e.g., cost and time) and the distance is represented as η^2, whereas for this thesis, only a single cost objective, the cost matrix, is considered.

The second heuristic parameter represents the inverse of the minimum number of steps (cells) required to move from node i to the goal node t. In [52], this is defined using node-to-node connections. In a raster-based adaption, where

diagonal movement is permitted, this translates to the weighting illustrated in Figure 3.3, making the second heuristic designed to motivate the algorithm to prioritize shorter distances.

4	3	2	1	**t**
$3+1\sqrt{2}$	$2+1\sqrt{2}$	$1+1\sqrt{2}$	$0+1\sqrt{2}$	1
$2+2\sqrt{2}$	$1+2\sqrt{2}$	$0+2\sqrt{2}$	$1+1\sqrt{2}$	2
$1+3\sqrt{2}$	$0+3\sqrt{2}$	$1+2\sqrt{2}$	$2+1\sqrt{2}$	3
i	$1+3\sqrt{2}$	$2+2\sqrt{2}$	$3+1\sqrt{2}$	4

(a) Weighted matrix: minimum number of diagonal and straight steps from each node to sink node t, expressed as straight $+ \min(\cdot)\sqrt{2}$.

$\frac{1}{4}$	$\frac{1}{3}$	$\frac{1}{2}$	$\frac{1}{1}$	**t**
$\frac{1}{3+1\sqrt{2}}$	$\frac{1}{2+1\sqrt{2}}$	$\frac{1}{1+1\sqrt{2}}$	$\frac{1}{0+1\sqrt{2}}$	$\frac{1}{1}$
$\frac{1}{2+2\sqrt{2}}$	$\frac{1}{1+2\sqrt{2}}$	$\frac{1}{0+2\sqrt{2}}$	$\frac{1}{1+1\sqrt{2}}$	$\frac{1}{2}$
$\frac{1}{1+3\sqrt{2}}$	$\frac{1}{0+3\sqrt{2}}$	$\frac{1}{1+2\sqrt{2}}$	$\frac{1}{2+1\sqrt{2}}$	$\frac{1}{3}$
i	$\frac{1}{1+3\sqrt{2}}$	$\frac{1}{2+2\sqrt{2}}$	$\frac{1}{3+1\sqrt{2}}$	$\frac{1}{4}$

(b) Second heuristic matrix: inverse of weighted cost from each node to sink node t, expressed as $\frac{1}{\text{straight}+\min(\cdot)\sqrt{2}}$.

Figure 3.3: Illustration of (a) the weighted matrix and (b) the second heuristic matrix.

The two heuristic parameters are combined to create the transition rule:

$$p_{ij} = \begin{cases} 1, & \text{if } q \leq q_0 \text{ and } j = \arg\max_{u \in \omega(i)} A_{iu} \\ \dfrac{A_{ij}}{\sum_{u \in \omega(i)} A_{iu}}, & \text{if } q > q_0 \text{ and } j \in \omega(i) \\ 0, & \text{otherwise} \end{cases}$$

Here, $A_{iu} = (\tau_{iu}^{1\alpha} \cdot \eta_{iu}^{11\beta})^{\lambda} \cdot (\tau_{iu}^{2\alpha} \cdot \eta_{iu}^{12\beta})^{1-\lambda} \cdot \eta_{u}^{2\delta}$, q is a random number uniformly distributed in $(0, 1)$ and q_0 is the base rate of the transition ratio, controlling the balance between deterministic and random selection.

The value of q_0 significantly impacts the algorithm. A larger q_0 favors deterministic decisions, accelerating convergence but reducing exploration, while smaller q_0 increases randomness, enhancing global search ability but potentially slowing convergence.

$\omega(i)$ is a set of outgoing nodes from cell i that have not been previously visited, and λ is determined by the following rules:

$$\lambda = \begin{cases} 0, & \text{if } h \leq a \\ \dfrac{h-a}{b-a}, & \text{if } a < h < b \\ 1, & \text{if } h \geq b \end{cases}$$

Here, a and b are two experimental parameters dependent on the problem size, while h is the ant index number in the colony. For this study, the removal of time objective means that we effectively skip the δ parameter as it is unnecessary and set $\eta^{12} = \eta^2$. Essentially, the distance η^2 becomes the second objective η^{12}, and the cost vector instead contains the traversal cost based on terrain and the distance. As a result, A_{iu} becomes:

$$A_{iu} = (\tau_{iu}^{1\alpha} \cdot \eta_{iu}^{11\beta})^{\lambda} \cdot (\tau_{iu}^{2\alpha} \cdot \eta_{iu}^{12\beta})^{1-\lambda}$$

With this, λ is retained, allowing the algorithm to dynamically adjust the balance between the cost-based heuristic η^{11} and the distance-based heuristic η^{12} for each ant. This preserves the spirit of multi-objective exploration, even though only one 'cost' objective is used in this simplified setting. By tuning λ across the range of ants in the colony (from 0 to 1), a diverse set of solutions can still be encouraged, ranging from those that emphasize minimal cost to those that emphasize shortest distance to the goal.

In effect, the cost matrix continues to guide exploitation of low-cost paths, while the distance-to-goal heuristic serves as a guiding mechanism for promoting path feasibility and goal-directed search. This blending ensures that ants are not only rewarded for finding inexpensive paths, but also for making steady progress toward the goal node.

Furthermore, removing δ simplifies the formulation, while still preserving the essential influence of the second heuristic. In this way, the modified transition rule aligns closely with the original formulation in [52], and adapts well to the raster grid. If further objective is wished to be added, one can simply return to the previous formula.

(2) **Uneven Initial Pheromone Distribution:** The initial pheromone levels $\tau_{ij}(0)$ are unevenly distributed based on the relative distance between the current node, the next node, and the connection between the start and end nodes. This is represented as:

$$\tau_{ij}(0) = a_0 \frac{d_{SE}}{d_{Si} + d_{ij} + d_{jE}}, \quad j \in C,$$

- d_{SE} is the Euclidean distance between the start and end nodes.
- d_{Si} is the Euclidean distance between the start node and current node i.
- d_{ij} is the Euclidean distance between the current node i and the next node j, which equals 1 in a unit-spacing grid.
- d_{jE} is the Euclidean distance from the next node j to the end node.

- C is the set of possible next nodes.
- a_0 is a constant, typically set to 1.

This formulation ensures that paths closer to the straight-line connection between the start and end nodes receive higher initial pheromone values, reducing randomness of the initial search, avoiding early blindness, and accelerating convergence speed [43].

(3) **Improved Pheromone Update Mechanism:**

To prevent premature convergence and stagnation, the pheromone values associated with both objectives are constrained within a predefined range $\tau_{\min}$ and $\tau_{\max}$:

$$\tau_{ij}(t) = \begin{cases} \tau_{\max}, & \text{if } \tau_{ij}(t) > \tau_{\max}, \\ \tau_{\min}, & \text{if } \tau_{ij}(t) < \tau_{\min}, \\ \tau_{ij}(t), & \text{otherwise.} \end{cases}$$

This clamping mechanism ensures that pheromone levels neither saturate nor vanish. By enforcing these bounds, the algorithm maintains an appropriate level of search diversity, facilitating a balance between intensification (exploitation of promising paths) and diversification (exploration of new regions of the search space) [43].

At the end of each iteration, the algorithm performs a global pheromone update to reinforce the paths included in the current Pareto-optimal set. For each non-dominated solution, pheromone is deposited along its corresponding path in both pheromone matrices associated with the two objectives.

The pheromone deposit is inversely proportional to the total cost of the solution:

$$\Delta\tau = \frac{G}{2(c_1 + c_2)}$$

Here, c_1 and c_2 are the values of the two objectives, and G is a normalization constant defined as the sum of the grid dimensions, i.e., $G =$ `grid_size_x` + `grid_size_y`. In the original formulation, the deposit is defined as the number of nodes divided by the objective cost. The adaption of total cost across both objectives promotes the reinforcement of higher-quality (lower-cost) solutions, increasing their selection probability in subsequent iterations.

The pheromone value for each cell visited along the solution path is then updated using the following equation:

$$\tau_{ij}(t+1) = \min(\tau_{\max}, \tau_{ij}(t) + \Delta\tau) \cdot \rho$$

This formulation reinforces good paths while also applying evaporation to control pheromone accumulation, preventing any single path from dominating

too early and ensuring continued adaptability of the search process.

(4) **Deadlock Resolution:** In complex environments, ants may encounter a deadlock state where they cannot proceed further, leading to incomplete paths. This is exemplified in Figure 3.4. To balance computational complexity and solution diversity, a compromise method is proposed; ants in deadlock states are terminated, and the number of ants entering the same deadlock point (i, j) is counted as $n_{\text{Lost}}(i, j)$. The last two steps of the incomplete path $\mathbf{P}$ are then penalized based on:

$$\tau_{ij}(t+1) = \begin{cases} \left(1 - \phi^{\lceil n_{\text{Lost}}(i,j)/3 \rceil}\right) \cdot \tau_{ij}(t), & \text{if } \mathbf{P}(i,j) \in \text{Last two steps}, \\ \tau_{ij}(t), & \text{otherwise.} \end{cases}$$

Here, $\phi \in (0, 1)$ is the penalty factor and $\lceil \cdot \rceil$ denotes the ceiling function.

The more ants lost at a given location, the greater the pheromone reduction, lowering the likelihood that subsequent ants repeat the same mistake. Rather than allowing ants to move backward - which increases the risk of cyclic behavior, raises computational complexity, and undermines algorithmic simplicity - this approach promotes robust forward exploration while preserving efficiency, ensuring that paths leading to deadlocks are progressively avoided.

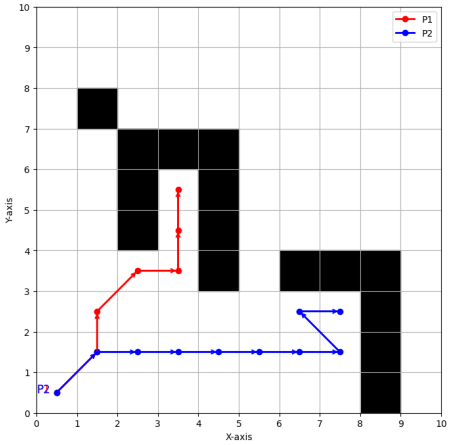


Figure 3.4: Status of a deadlock. Both ant P1 and P2 can't move in any direction.

These enhancements collectively address the limitations of basic ACO by improving convergence speed, avoiding local optima, and maintaining diversity in the solution space. The proposed modifications make the algorithm more suitable for large-scale and complex environments [43].

4

Methods

This section outlines the structure and methodology of the project, including the data preprocessing used to construct the environment and the subsequent application of the models to that environment.

4.1 Data Preprocessing

To enable a model to navigate an environment based on a set of parameters, all relevant data layers was first collected, processed, and integrated into a cost matrix for traversal. A structured approach involved gathering both *feature class data* and *raster data*. To optimize processing efficiency, the data was first clipped to a defined work area, reducing unnecessary computational overhead. Next, feature class data, including point, line, or polygon attributes, were converted into corresponding raster formats to ensure uniform shape, alignment, and spatial consistency with existing raster datasets. This conversion process, known as *rasterization*, enabled the application of AHP-based cost deductions, ultimately generating a grid-based traversal cost matrix for the model. The following sections provide a detailed explanation of this approach.

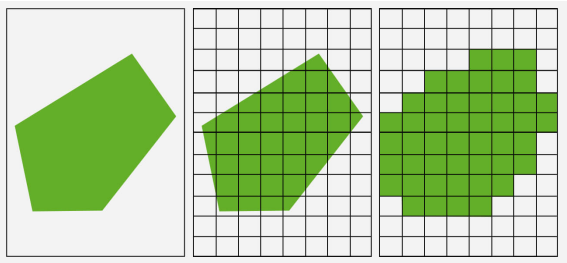


Figure 4.1: Visualization of polygon rasterization, where each grid cell is assigned a value of 1 if intersected by the polygon boundary. Adapted from [53].

4.1.1 Qualitative Parameters

Below are the parameters, listed by the team at WSP, common in a cable routing project and, hence, in different degrees needed to be considered by the model when generating a possible path.

(1) **Shortest Possible Distance**: Minimizing the total length of the cable route.

(2) **Impassable Areas**: Avoiding areas not possible to traverse, such as buildings, water bodies, landslide areas and restricted areas.

(3) **Minimizing Environmental Impact**: Minimizing clearance of forests, cultural heritage sites, natural habitats, and biotope-protected areas.

(4) **Distance Requirements**: Adhering to distance requirements from various objects as specified in SVK documents.

(5) **Minimizing Crossings**: Reducing the number of crossings with other infrastructure such as ditches, roads, streams, and power lines.

(6) **Excavation Preference**: Favoring soil excavation over rock excavation to reduce costs and environmental impact.

(7) **Bend Radius Limitations**: Ensuring that the cable follows permissible maximum bend radii.

4.1.2 GIS Parameter Layers

Table 4.1 below includes the WSP provided data used for the Gotland link testing ground. Since the cable routing today is manually conducted, water and forest areas, among others, are primarily identified visually using Web Map Service (WMS) maps. These WMS maps do not include any attribute tables, and can therefore not be rasterized. Instead, additional data is needed in order to enable an AI model to effectively identify and incorporate these features into the optimization process. Therefore, Table 4.1 was expanded with data layers from external sources. These were:

- Height raster data layer (cell size 50×50 meters) from Lantmäteriet [54]
- Water surface and line feature class data layers from the Swedish Meteorological and Hydrological Institute (SMHI) [55]
- Vegetation intensity index raster data layer (cell size 10×10 meters), landslide risk area feature class data layer, and multiple feature class data layers for protected areas (nature values, key habitats etc.) from Skogsstyrelsen (SkS) [56]
- Soil elements feature class data layer from Sveriges Geologiska Undersökning (SGU) [57]
- Multiple feature class data layers of land use and protected areas from Naturvårdsverket (NVV) [58]

As seen in the table, all the data was collectively mapped to all qualitative parameters given in Section 4.1.1, except (1) Shortest Possible Distance and (7) Bend

Radius Limitations. The shortest possible distance is an inherent quality of all models used, and thus, it is considered when running the models. The minimum bend radius R follows $R = k \times D$, where k is the bending factor, dependent of the cable type, and D is the outer cable diameter. For a 220 kV multi-core cable with screen, such as the one used in the Gotland link, $k \approx 15$ and $0.060 < D < 0.120$ meters according to the WSP team. This gives $R_{\max} \approx 15 \times 0.120 = 1.8$ meters, which is not exceeded for a raster with cell size 10×10 meters. Hence, further consideration of bend radius constraints was unnecessary. In summary, with the inclusion of these data layers, all qualitative parameters were effectively addressed.

Figure 4.2 shows some of these data layers visualized in ArcGIS Pro for the mainland and Gotland testing grounds, alongside the current planned cable route path by SVK which was manually created in ArcGIS Pro.

The presence of underground infrastructure, such as utility pipelines and fiber-optic cables, was identified as a relevant parameter in discussions with the team at WSP. To incorporate this information, the Swedish national web service *Ledningskollen* was initially considered [59]. This platform facilitates the identification of underground utilities and supports coordination with infrastructure owners to reduce the risk of damage during excavation. However, obtaining comprehensive data for the entire mainland and Gotland areas would have required more than five separate requests, each limited to a 10 km^2 area and subject to a maximum of two simultaneous inquiries. Given that the waiting time for each request could be up to two weeks, the decision was made not to pursue this option. If used, the data would have been integrated into the environment in the same manner as other spatial layers.

Regarding the landslide risk areas data layer, `main_RasoskreddKonsekvens-omradebranterYta`, which was used as an obstacle in this project, various landslide classification models were explored as a side project. However, after further investigation, it was determined that this aspect fell outside the project's scope and would be too time-consuming to integrate effectively. Nevertheless, the work conducted in this area is documented in Appendix A.3.

Table 4.1: Table of the used data for the Gotland link testing ground, provided both by WSP and various external sources. Each data layer has a geometry type, and is of either binary (B) or non-binary (NB) data type, and used as an obstacle (O) or non-obstacle (NO) in the environment. They are also mapped to the qualitative parameters in Section 4.1.1. The abbreviations of the sources are as follows: SkS = Skogsstyrelsen; SGU = Sveriges geologiska undersökning; NVV = Naturvårdsverket; LM = Lantmäteriet; SMHI = Swedish Meteorological and Hydrological Institute.

Data layer	**Geometry Type**	**Data Type (B/NB)**	**Data Usage (O/NO)**	**Description**	**Mapping**	**Source**
`work_area`	Polygon	B	NO	Delimited project work area.	N/A	Created by authors
`Approx_cable_location`	Polyline	B	NO	Estimated cable route.	N/A	Created by authors
`start_and_goal_fastlandet`	Point	B	NO	Starting and goal locations.	N/A	Created by authors
`63_5_2024.tif`	Raster	NB	NO	Terrain elevation model.	3, 6	LM
`nmdproduktivitet_ogeneraliserad_v1_1.tif`	Raster	NB	NO	Productivity model for forestry.	3	NVV
`NMD2023_Markanv_Bete_v1_0.tif`	Raster	NB	NO	Grazing land classification.	3, 6	NVV
`NMD2023_Markanv_Anlagda_omr_v1_0.tif`	Raster	NB	NO	Artificial land use areas.	2, 3, 6	NVV
`NMD2023bas_v0_1.tif`	Raster	NB	NO	General land cover classification.	2, 3, 6	NVV
`NNK_YTA`	Polygon	B	NO	Area-based nature protection sites.	3	NVV
`NNK_PKT`	Point	B	NO	Point-based nature protection areas.	3	NVV
`NNK_LIN`	Polyline	B	NO	Natural conservation boundaries.	3	NVV
`SNUS`	Polygon	B	NO	Protected natural areas.	3	NVV

Continued on the next page ...

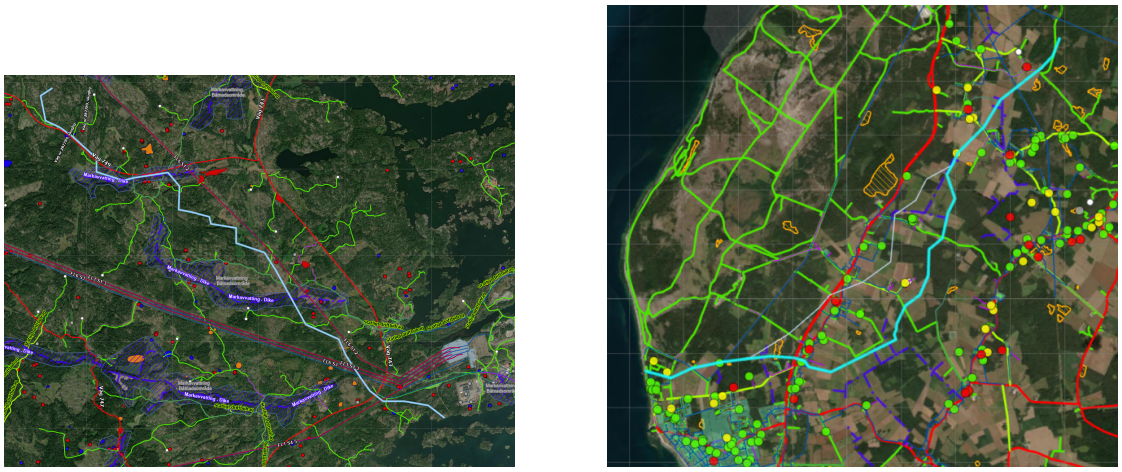
Data layer	Geometry Type	Data Type (B/NB)	Data Usage (O/NO)	Description	Mapping	Source
NR_polygon	Polygon	B	NO	Nature reserves.	3	NVV
main.grundlager	Polygon	NB	NO	Geological base layers.	6	SGU
Vattendragslinjer_nätverk_2016	Polyline	B	NO	Watercourse network representation.	5	SMHI
Vattenytor_2016	Polygon	B	O	Water surface representation.	2	SMHI
main_SkogoHistoriaPkt	Point	B	NO	Historical forest-related points.	3	SkS
main_RasoskreddKonsekvensomradebranterYta	Polygon	B	O	Landslide risk areas.	2	SkS
main_SkogoHistoriaLinje	Polyline	B	NO	Historical forestry boundaries.	3	SkS
main_SkogoHistoriaYta	Polygon	B	NO	Historical forest data.	3	SkS
main_NaturvardsavtalYta	Polygon	B	NO	Conservation agreement areas.	3	SkS
sksVegkvot.tif	Raster	NB	NO	Vegetation density index.	3, 6	SkS
main_NyckelbiotopYta	Polygon	B	NO	Key biotope areas.	3	SkS
sksSLUmfMarkfuktiget08.tif	Raster	NB	NO	Soil moisture data.	6	SkS
main_NaturvardeYta	Polygon	B	NO	Areas of ecological importance.	3	SkS
main_Storskogsbrukets Nyckelbiotop	Polygon	B	NO	Large-scale forestry biodiversity areas.	3	SkS
FMIS_RAÄ_Lämningar_Punkt	Point	B	O	Point-based archaeological remains.	2, 3	WSP
FMIS_RAÄ_Lämningar_Linje	Polyline	B	O	Line-based archaeological remains.	2, 3	WSP

Continued on the next page ...

Data layer	Geometry Type	Data Type (B/NB)	Data Usage (O/NO)	Description	Mapping	Source
Byggnadsminne	Point	B	O	Historically significant buildings.	2, 3	WSP
Väghållare	Polyline	NB	NO	Road ownership and responsibility.	5	WSP
FMIS_RAÄ_Lämningar_Yta	Polygon	B	O	Cultural heritage sites (area-based).	2, 3	WSP
Kyrkor	Point	B	O	Locations of churches, considered obstacles.	2	WSP
Vägbommar_och_vändplaner	Point	B	O	Indicates road barriers and turnaround points.	2	WSP
GEAB_6016_MSP_Punktobjekt	Point	B	O	Electrical infrastructure points.	2, 4	WSP
GEAB_6374_MSP_Punktobjekt	Point	B	O	Electrical infrastructure points.	2, 4	WSP
Markavvattning__Dike	Polyline	B	NO	Drainage ditches, affecting water flow.	5	WSP
RAÄ_RI_Kulturmiljövård_MB3kap6	Polygon	B	O	Areas under cultural heritage protection.	2, 3	WSP
Ytor_för_rättigheter_och_gemensamhetsanläggningar	Polygon	B	O	Areas of legal rights and shared facilities.	2	WSP
byggnad	Polygon	B	O	Buildings considered obstacles.	2	WSP
Markavvattning__Båtnadsområde	Polygon	B	NO	Areas benefiting from drainage systems.	3, 6	WSP
Kustlinje	Polyline	B	O	Coastal lines for mapping.	2	WSP

Continued on the next page ...

Data layer	Geometry Type	Data Type (B/NB)	Data Usage (O/NO)	Description	Mapping	Source
Kustlinje_Yta	Polygon	B	O	Coastal boundary representation.	2	WSP
Sjöar	Polygon	B	O	Lake areas, not suitable for routing.	2	WSP
Brunnar__Shape_	Point	B	O	Locations of wells.	2	WSP
Stolpnummer	Point	B	O	Identifies utility pole locations.	2, 4	WSP



(a) Mainland testing ground.

(b) Gotland testing ground.

Figure 4.2: Visualization of feature class data layers for the (a) mainland and the (b) Gotland testing grounds in ArcGIS Pro. The light blue lines are the manually created approximate planned cable path by SVK.

4.1.3 GIS Data to Raster Layers

To enable the models to traverse and evaluate the testing ground effectively, all geospatial data were processed into structured grids, referred to as raster layers. This transformation ensured that spatial information—such as elevation, land cover, water bodies, and road networks—was represented in a format suitable for computational path-finding and cost analysis.

The processing pipeline in ArcGIS Pro followed these steps:

(1) **Separation of feature classes:** All vector-based feature class layers were categorized into binary (obstacle/non-obstacle) and non-binary (continuous)

groups to facilitate structured rasterization.

(2) **Reference raster creation:** Externally downloaded raster datasets were processed to create a reference raster for alignment. This reference raster was used for snapping (cell alignment) and cell size matching when rasterizing feature class layers.

   (a) Height raster tiles, originally provided per Swedish county at 50×50 m resolution, were merged into a single raster and resampled to 10×10 m resolution to match the vegetation intensity index raster.

   (b) A slope raster was generated from the height raster using ArcGIS Pro geoprocessing tools.

   (c) Soil moisture rasters, already at 10×10 m resolution but provided per Swedish county, were merged into a single dataset.

   (d) All processed raster layers were clipped to the work area extent, snapped to the reference raster, and stored in a geodatabase (GDB).

(3) **Feature class rasterization:**

   (a) Binary feature classes were clipped to the work area and rasterized, ensuring alignment with the processed raster layers.

   (b) Non-binary feature classes were processed separately, as different attributes were used for rasterization.

(4) **Final raster storage:** All resulting rasters, whether processed from external data or rasterized from vector layers, were stored in the GDB.

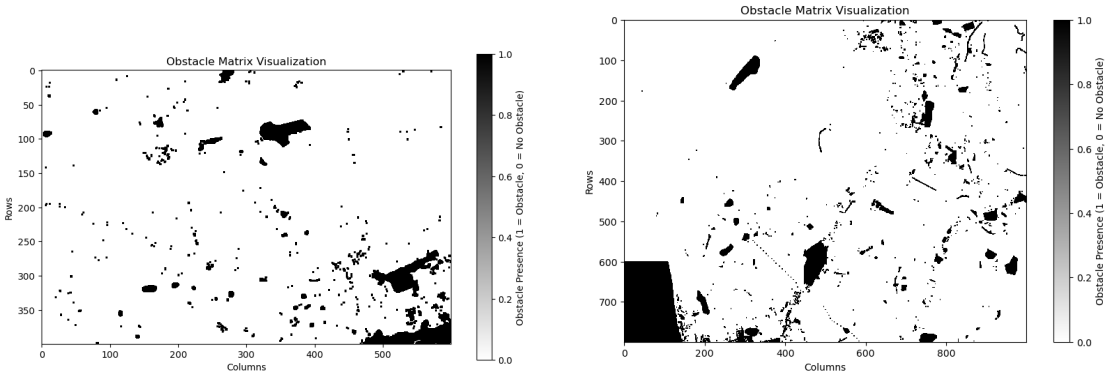
(5) **Loading into Python for further processing:** The processed rasters were loaded from the GDB using ArcPy. Each raster was:

   (a) Converted into a NumPy array.

   (b) Shape inconsistencies were checked and corrected.

   (c) The data arrays were stored in a dictionary for structured access.

(6) **Obstacle matrix generation:** The obstacle matrix was created by merging obstacle layers and applying a 10-meter buffer around obstacles, to ensure safe clearance and distance requirements. The resulting matrices are seen in Figure 4.3.



(a) Mainland testing ground.

(b) Mainland testing ground.

Figure 4.3: The obstacle matrices for (a) the mainland and (b) the Gotland testing grounds. The matrices were generated by layers considered to be absolute barriers, or obstacles, which the models were not allowed to traverse.

4.1.4 Raster Layers to Final Environment

After constructing the obstacle matrix, the next step was to generate the cost matrix using the AHP. To guide this process, similar projects were reviewed as reference, after which the AHP methodology was adapted to the specific requirements of this project. This procedure resulted in the creation of the final environment, represented as a cost matrix.

4.1.4.1 Background to AHP in Similar Projects

The hierarchical structure of the landscape followed the same general form as the example model in Figure 3.1, though expanded due to the larger number of parameters considered. The authors initially discussed applying the AHP using available datasets and assigning weights based on internal expertise at WSP. However, building such a complex model required either extensive domain knowledge or a formal series of expert interviews and validations. The authors lacked deep domain knowledge, and although the WSP team possessed considerable expertise, it was not readily transferable to the specific datasets and parameter structures introduced in this project. An interview-based approach also presented risks: the potential for inconsistent or overly averaged weightings due to varying expert perspectives, and significant time demands that could delay algorithm development. Given these limitations, the expert-driven weighting strategy was set aside. Instead, parameter weights were sourced from existing literature.

Building on the AHP foundation introduced in Section 3.1, a review of existing literature revealed that most GIS-based AHP studies focus on overhead transmission lines, with limited attention paid to underground routing. This pattern is evident in several frequently cited works [28, 21, 15, 20, 60, 16]. The term *transmission lines* is commonly associated with overhead systems, and many studies, including [60], explicitly examine overhead infrastructure while still being referenced for weight

selection in broader routing contexts. This raises concerns about the transferability and reliability of such weightings for underground cable planning, highlighting a gap in the current research.

In contrast, literature on underground pipeline routing—particularly for natural gas and oil—offers a more relevant framework, as these systems share terrain-related challenges with underground power cables. Unlike overhead transmission lines, which prioritize factors such as visibility and clearance, pipeline routing studies account for subsurface characteristics, soil composition, environmental constraints, and land-use impact, making them more applicable to this project.

Two studies by the same authors [61, 62] used the AHP to determine parameter weights for natural gas pipeline routing. Weights were derived through pairwise comparisons that reflected the relative influence of each factor, informed by Environmental Impact Assessments (EIAs), previous applications, scientific literature, and expert input from BOTAS (Turkey's Petroleum Pipeline Corporation). The resulting factors and sub-factors, along with their assigned weights, are presented in Table 4.2. These were used in this project.

4.1.4.2 AHP Application in This Project

Although the AHP weights used in this project are predefined, the general methodology for assessing the relative importance of factors influencing transmission line routing follows the AHP framework:

(1) Defining the set of criteria and sub-criteria relevant to the routing decision.

(2) Constructing a **pairwise comparison matrix** using weights extracted from previous studies on natural gas pipeline routing.

(3) Normalizing the matrix to a **1-9 scale** to ensure consistency in weight differentiation.

Following these principles and adapting the template model in Figure 3.1 to our specific environmental parameters, Figure 4.4 illustrates how the various criteria (factors) influence the model's decision-making process. These decisions correspond to the agent's possible movements to one of its eight neighboring cells in the Moore neighborhood. However, these principles serve as a conceptual framework rather than a direct implementation. Instead, a constructed cost matrix based on factor and sub-factor weights was used, implicitly integrating these principles in a more abstract and indirect manner, described below.

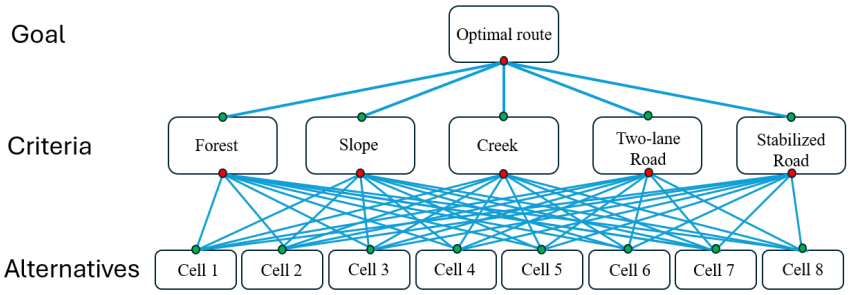


Figure 4.4: The AHP model principal structure for decision-making in our implementation, schematically showing the alternatives in the rasterized environment. The model evaluates and selects the most optimal surrounding cell for movement given the criteria and goal.

Table 4.2: Factor and sub-factor weights influencing natural gas pipeline routing [62], deemed applicable to underground cable routing in this thesis project.

Factors / Sub-Factors	**Weights**	**CR**
Land Cover	**0.263**	**0.0247**
Forest	0.096	
Cultivated Areas (Seasonal Agriculture)	0.043	
Agricultural Areas	0.063	
Wetland (absolute barrier)	0.134	∞
Rocky Areas	0.226	
Pasture Areas	0.028	
Settlement Areas (absolute barrier)	0.411	∞
Slope	**0.211**	**0.0108**
$<10°$	0.031	
10 - 20°	0.060	
20 - 30°	0.081	
30 - 40°	0.124	
40 - 50°	0.152	
50 - 60°	0.185	
$>60°$	0.367	
Geology	**0.162**	**0.0443**
Acid-Intermediate Intrusives	0.473	
Basic-Ultrabasic Rocks	0.288	
Metamorphic Rocks	0.149	
Volcanic Rocks	0.054	
Sedimentary Rocks	0.036	

Factors / Sub-Factors	Weights	CR
Soil	**0.130**	**0.0278**
I. Class soils – Excellent Agricultural	0.269	
II. Class soils	0.251	
III. Class soils	0.193	
IV. Class soils	0.104	
V. Class soils	0.081	
VI. Class soils	0.045	
VII. Class soils	0.037	
VIII. Class soils – Non Agricultural	0.020	
Landslide	**0.092**	**0.0334**
Active Landslide Areas (absolute barrier)	0.633	∞
Potential Landslide Areas	0.260	
Old Landslide Areas	0.106	
Stream	**0.040**	**0.0063**
River	0.444	
Stream	0.053	
Canal	0.262	
Brook	0.153	
Creek	0.089	
Road	**0.030**	**0.0238**
Highway	0.486	
Three-Lane Road	0.222	
Two-Lane Road	0.121	
Stabilized Road (two-lane)	0.090	
Stabilized Road (one-lane)	0.044	
Seasonal Road	0.037	
Protected Area	**0.049**	**0.0290**
Level I	0.407	
Level II	0.129	
Level III	0.079	
Urban Protected Areas	0.052	
Historical Protected Areas	0.333	
Recreation	**0.023**	**0.0167**
Upland	0.039	
Tourism Center	0.262	
Historical Monument	0.492	
Picnic Areas	0.069	
Promenade Areas	0.138	
Fault Line (absolute barrier)	∞	

4.1.4.3 Generating Cost Matrix

Given the background provided in the previous subsection, the process of constructing the cost matrix followed these steps:

(1) **Mapping data layers to AHP factors:** The data layers in Table 4.1 were mapped to AHP factors and sub-factors of Table 4.2. This resulted in Table 4.3. This ensured that each layer contributed appropriately to the cost calculation. As seen, the data layer names began with either `r_binary` or `r_non_binary` following the B/NB classification in Table 4.1. The mapping was implemented with a nested dictionary in Python. The **Indices** column refers either to:

   (a) The array element values $[i_0, i_1, \dots]$ that map the given layer to the sub-factor.

   (b) The array value thresholds $(t_{\text{lower}}, t_{\text{upper}})$ that map the given layer to the sub-factor.

   For example, forested areas (Forest, under Land Cover) were captured where either `r_non_binary_ma_baskarta = 110` or `r_non_binary_forest` $\in$ `(40,100)`.

   Utilizing the same AHP framework without modifications and directly mapping the data layers to the corresponding sub-factors ensured that the CR for each factor remained unchanged. This approach preserved the integrity of the analysis while significantly simplifying the workflow.

   The names/descriptions of the elements used in the **Indices** column in Table 4.3 are found in Table A.3 in Appendix.

(2) **Cost matrix computation:** The cost matrix was generated by:

   (a) Iterating through all AHP factors in the created nested dictionaty.

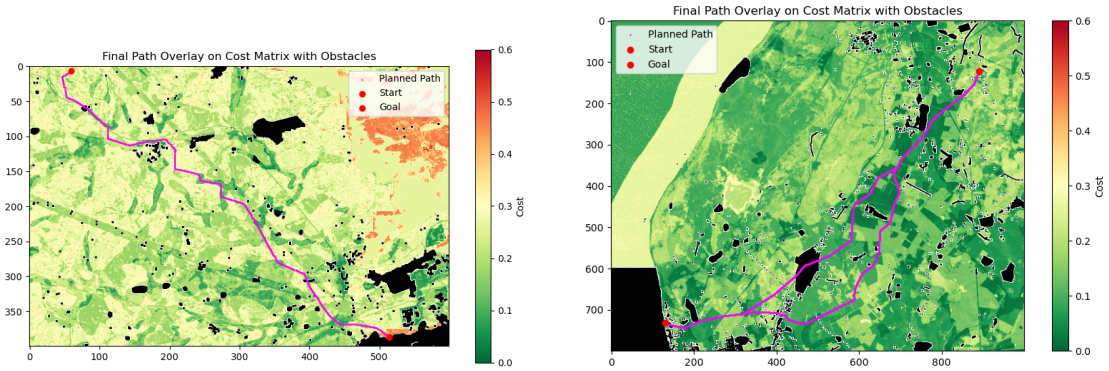
   (b) Extracting the relevant rasters and applying conditions (element or threshold values).

   (c) Multiplying each raster by its assigned AHP weight.

   (d) Summing all weighted layers to obtain the final cost matrix.

(3) **Saving Processed Data for Model Execution:** Finally, the obstacle and cost matrices, along with the planned path and the start-end coordinates, were saved as arrays. These files served as final environment for the agents, enabling efficient route optimization using the preprocessed GIS datasets.

The final cost matrix is visualized in Figure 4.5. Low-cost areas are shown in green, representing regions with low traversal cost, while high-cost areas appear in red. The magenta line overlaid on the map corresponds to the currently planned path by SVK.



(a) Mainland testing ground.

(b) Gotland testing ground.

Figure 4.5: Cost matrix visualization of (a) the mainland and (b) the Gotland testing ground of the Gotland link. The magenta lines are the manually created approximate planned cable path by SVK.

Table 4.3: Mapping of the data layers from Table 4.1 to the factors and sub-factors of the AHP model in Table 4.2 from [61]. The **Indices** column denotes either specific array element values $[i_0, i_1, \dots]$ or threshold ranges $(t_{\text{lower}}, t_{\text{upper}})$ that associate each layer with its corresponding sub-factor. While not all sub-factors are utilized in this project, they are included to maintain the CR values of the original model. Descriptions of the elements in the **Indices** column are found in Table A.3. Note that all factors and sub-factors are not in use.

Factors / Sub-Factors	Weight	Layer	Indices
Land Cover	**0.263**		
Forest	0.096	`r_non_binary_ma_baskarta`	[110]
		`r_non_binary_forest`	(40,100)
Cultivated Areas	0.043	`r_non_binary_ma_bete`	[6]
Agricultural Areas	0.063	`r_non_binary_ma_baskarta`	[3]
Wetland	0.134	`r_non_binary_ma_baskarta`	[2,23,120,128]
Rocky Areas	0.226	`r_non_binary_jordartlager`	[18]
Pasture Areas	0.028	`r_non_binary_ma_baskarta`	[41,42]
Settlement Areas	0.411	`r_non_binary_ma_baskarta`	[51,52]
Slope	**0.211**		
<10°	0.031	`r_non_binary_slope`	(0,10)
10–20°	0.060	`r_non_binary_slope`	(10,20)
20–30°	0.081	`r_non_binary_slope`	(20,30)
30–40°	0.124	`r_non_binary_slope`	(30,40)
40–50°	0.152	`r_non_binary_slope`	(40,50)
50–60°	0.185	`r_non_binary_slope`	(50,60)
>60°	0.367	`r_non_binary_slope`	(60,100)

Factors / Sub-Factors	Weight	Layer	Indices
Geology	**0.162**		
Acid-Intermediate Intrusives	0.473	r_non_binary_jordartlager	[18,6, 51]
		r_non_binary_markfukt	[101]
Basic-Ultrabasic Rocks	0.288	r_non_binary_jordartlager	[5,15, 30, 49, 50]
Metamorphic Rocks	0.149	r_non_binary_jordartlager	[4,11, 65]
Volcanic Rocks	0.054	r_non_binary_jordartlager	[44]
Sedimentary Rocks	0.036	r_non_binary_jordartlager	[16,3,32]
Soil	**0.130**		
I. Class soils – Excellent Agri.	0.269	r_non_binary_ma_baskarta	[3]
		r_non_binary_markfukt	(50,80)
II. Class soils	0.251	r_non_binary_ma_baskarta	[54]
		r_non_binary_ma_anlagda_områden	[4]
		r_non_binary_markfukt	(80,100)
III. Class soils	0.193	r_non_binary_ma_baskarta	[118]
IV. Class soils	0.104	r_non_binary_ma_baskarta	[42]
V. Class soils	0.081	r_non_binary_ma_baskarta	[41]
VI. Class soils	0.045	Not in use	
VII. Class soils	0.037	Not in use	
VIII. Class soils	0.020	Not in use	
Landslide	**0.092**		
River	0.633	Not in use	
Stream	0.260	Not in use	
Creek	0.106	Not in use	
Stream	**0.040**		
River	0.444	r_binary_Markavvattning--Båtnadsområde	[1]
Stream	0.053	r_binary_Vattendragslinjer_nätverk_2016	[1]
Canal	0.262	Not in use	
Brook	0.153	Not in use	
Creek	0.089	r_binary_Markavvattning--Dike	[1]
Road	**0.030**		
Highway	0.486	Not in use	
Three-Lane Road	0.222	Not in use	
Two-Lane Road	0.121	r_non_binary_väghållare	[1]
Stabilized Road (2-lane)	0.090	r_non_binary_väghållare	[2]
Stabilized Road (1-lane)	0.044	r_non_binary_väghållare	[3]
		r_non_binary_ma_baskarta	[53]
Seasonal Road	0.037	Not in use	
Protected Area	**0.049**		
Level I	0.407	r_binary_main_NyckelbiotopYta	[1]
		r_binary_NR_polygon	[1]
		r_binary_NNK_LIN	[1]
		r_binary_NNK_YTA	[1]
		r_binary_NNK_PKT	[1]
		r_binary_main_NaturvardsavtalYta	[1]
		r_binary_main_StorskogsbuketsNyckelbiotop	[1]
Level II	0.129	r_binary_SNUS	[1]
		r_binary_main_NaturvardeYta	[1]
Level III	0.079	Not in use	
Urban Protected Areas	0.052	Not in use	
Historical Protected Areas	0.333	r_binary_main_SkogoHistoriaLinje	[1]
		r_binary_main_SkogoHistoriaPkt	[1]
		r_binary_main_SkogoHistoriaYta	[1]
Recreation	**0.023**		
Upland	0.039	Not in use	
Tourism Center	0.262	Not in use	
Historical Monument	0.492	Not in use	
Picnic Areas	0.069	Not in use	
Promenade Areas	0.138	Not in use	

4.2 Model Application on Environments

4.2.1 Initial Testing

To evaluate the suitability of the Dijkstra, A*, and ACO algorithms for cable routing, an initial testing phase was conducted using a synthetic environment, as shown in Figure 1.2. This environment, similar in structure to the real-world testing grounds, was represented as a graph (Figure 4.6), where each edge carried two attributes: cost and distance. Diagonal movements were associated with an increased distance and cost, scaled by a factor of $\sqrt{2}$. This controlled setup enabled effective debugging, parameter tuning, and evaluation of each algorithm's ability to traverse the cost surface while avoiding obstacles.

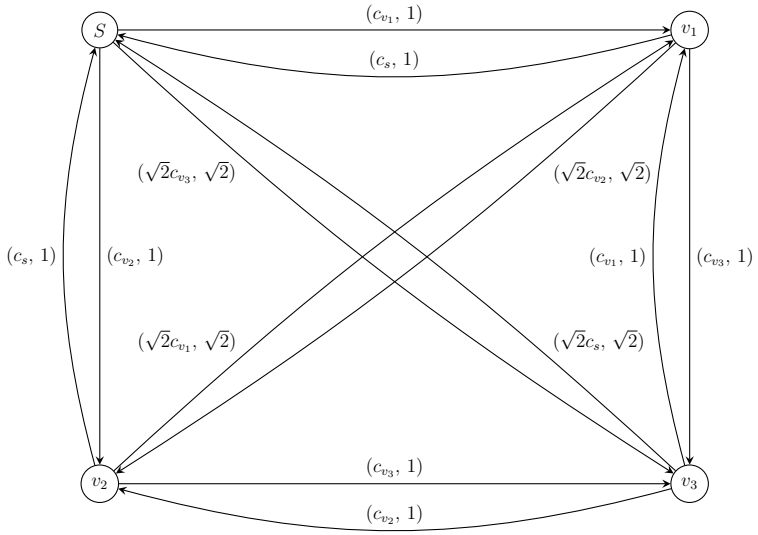


Figure 4.6: The graph structure of the rasterized environments. Each edge (u, v) from vertex u has two costs (d_1, d_2); d_1 is the traversal cost c_v of going to vertex v, and $d_2 \in \{1, \sqrt{2}\}$ is the distance to v.

Once the models demonstrated reliable performance on this environment, they were applied to the AHP model generated cost matrices in Figure 4.5.

4.2.2 Size-Varied Subgrids of the Final Environment

The performance of the routing models depended not only on the algorithm itself, but also on the spatial layout and cost heterogeneity of the environment. While the deterministic algorithms performed consistently across varying settings, ACO relied on several interdependent hyperparameters that strongly influenced both solution quality and runtime.

Since the parameter set performed differently across all grid sizes and terrain types, a gride size based tuning approach was adopted, where three differently sized sub-regions of the final mainland environment were selected: 100×100, 200×200, and

300×300 cells (1–9 km^2), shown in Figure 4.7. Three key ACO hyperparameters were analyzed: the base exploration rate q_0, and the pheromone update bounds a and b. To explore their effects, three representative configurations were defined:

(1) **Cost-focused (`cost_focus`):** prioritizes cost minimization by biasing most ants toward cost-sensitive behavior.

(2) **Distance-focused (`distance_only`):** favors short paths by ignoring cost in the heuristic.

(3) **Mixed (`mixed`):** interpolates between cost and distance, aiming for balanced exploration of the Pareto front.

The runs on all subgrids were conducted using the parameter settings listed in Table 4.4, with q_0 tested at three distinct values to assess its effect. 27 configurations were run three times, making it 81 runs distributed over the three subgrids.

In parallel, the sensitivity of the A* algorithm to the weight parameter w_{cost}, which balances the influence of cost versus distance, was also analyzed. This enabled a comparison of how both A* and ACO manage trade-offs across varying spatial scales, helping to identify which strategies deliver the most effective routing performance under different environmental conditions.

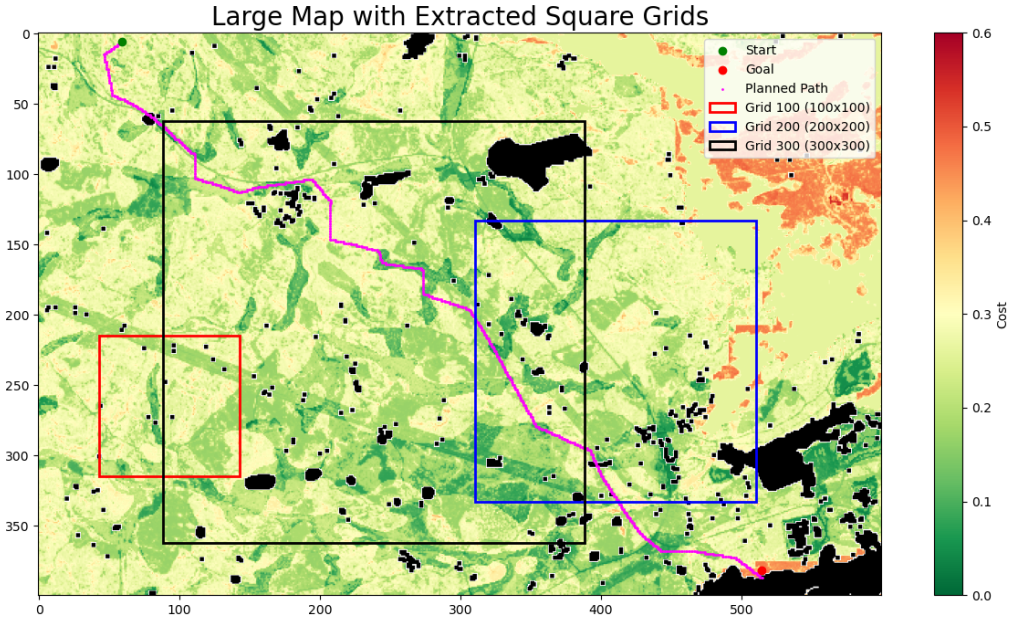


Figure 4.7: Grid areas of varying sizes used for hyperparameter tuning and model evaluation.

Table 4.4: Summary of Tuned and Fixed Hyperparameters for the ACO Study

Parameter	Value(s)	Description
α_1	1.0	Fixed for objective 1, influences the pheromone strength for cost-based decisions.
β_1	1.0	Fixed for objective 1, influences the heuristic (cost) component.
α_2	1.0	Fixed for objective 2. influences the pheromone strength for distance-based decisions.
β_2	1.0	Fixed for objective 2, influences the heuristic (distance) component.
λ_{penalty}	0.01	Penalty factor used in deadlock resolution.
N	100	Total number of ants used in the algorithm.
$I_{\max}$	100	Total number of iterations used in the algorithm.
q_0	0.9, 0.95, 0.99	Base rate of the transition ratio. Controls the balance between exploitation and exploration. Higher values favor exploitation.
Parameters a and b	Distance Only: $a = 100,\ b = 101$ Mixed: $a = -1,\ b = 101$ Cost focus: $a = -1,\ b = 50$	Determine the strategy mode by controlling whether ants use the distance heuristic (if $\lambda = 0$) or the cost-based heuristic (if $\lambda = 1$), or a mix in between.

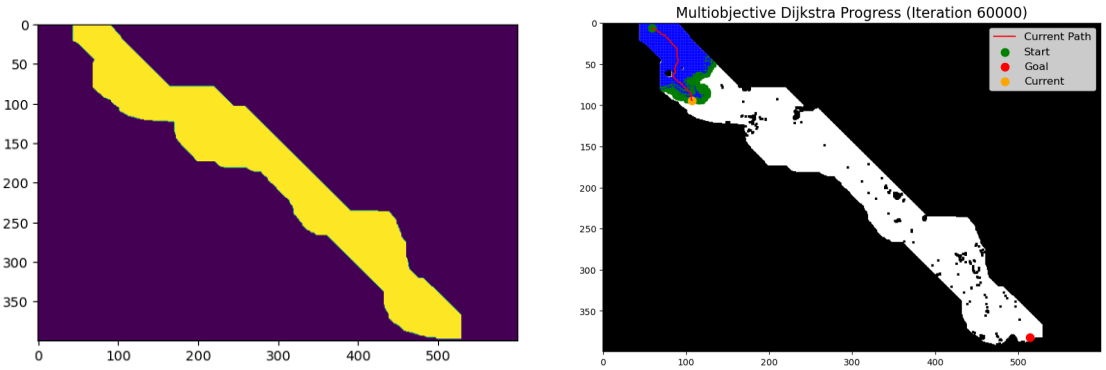
4.2.3 Final Environment

The final application of the models was conducted on the complete environments of the Gotland link, encompassing both the mainland and Gotland areas. The Dijkstra Classic and ACO algorithms were applied directly to both environments without modification. However, for A* and Dijkstra MSP, a reduced search space, or *corridor*, was generated based on the paths produced by the Dijkstra Classic algorithm. The process for constructing this reduced environment is described below.

4.2.3.1 Environment Reduction by Corridor Search

In this approach, two extreme single-objective paths were first computed using the Dijkstra Classic algorithm: one minimizing travel distance, $\mathbf{P_S}$, and another minimizing terrain cost, $\mathbf{P_C}$. These two paths naturally define a corridor, within which all Pareto-optimal biobjective solutions are expected to reside, as any efficient solution must have a length and cost bounded by those of $\mathbf{P_S}$ and $\mathbf{P_C}$. By constructing a mask that restricts the search space to this corridor — effectively eliminating regions

that cannot contain valuable alternatives — the number of cells to be explored is significantly reduced, without adversely impacting model performance. Figure 4.8 illustrates (a) the reduced search space mask for the mainland testing ground and (b) the Dijkstra MSP algorithm operating within the corridor.



(a) Mask for reducing the search space of the mainland testing ground.

(b) Dijkstra MSP algorithm running on the reduced search space.

Figure 4.8: (a) The reduced search space mask for the mainland testing ground of the Gotland link, and (b) the Dijkstra MSP algorithm running on the this reduced search space.

4.2.3.2 Planned Path Distance and Traversal Cost

The manually planned paths for the final environments served as baseline references against which the model outputs could be compared. Both the path length and the traversal cost of these manually defined routes were computed in the same manner as for the models. For the traversal cost, two methods were applied to handle segments that intersect with obstacles, areas that the models are not permitted to traverse:

(1) Obstacle cells were assigned a traversal cost of 0.

(2) Obstacle cells were assigned the mean cost of the entire cost matrix.

These two approaches are denoted with asterisks in Table 5.3: * for zero-cost treatment and ** for mean-cost substitution.

While the mainland environment included a single planned path, the Gotland environment featured two distinct planned routes, as illustrated in Figure 1.3(b). These were referred to as the *upper* and *lower* routes (UR and LR). Both shared the same start and end points but diverged in their intermediate segments, following different routes through the central part of the environment.

4.3 Normalization of Cost and Distance

To enable comparison of results both within and across environments and models, traversal cost and distance were normalized using the minimum-cost and shortest-distance paths, $\mathbf{P_C}$ and $\mathbf{P_S}$, respectively. As a result, all normalized values were ≥ 1, where a value of 1 corresponds to the optimal reference path $\mathbf{P_C}$ or $\mathbf{P_S}$. This normalization was applied only in the aggregated analysis of all model runs across environments, as visualized in Figure 5.6.

4.4 Pipeline summary

The following schematically summarizes the pipeline process from data colletion and preprocessing to generated outputs and evaluation.

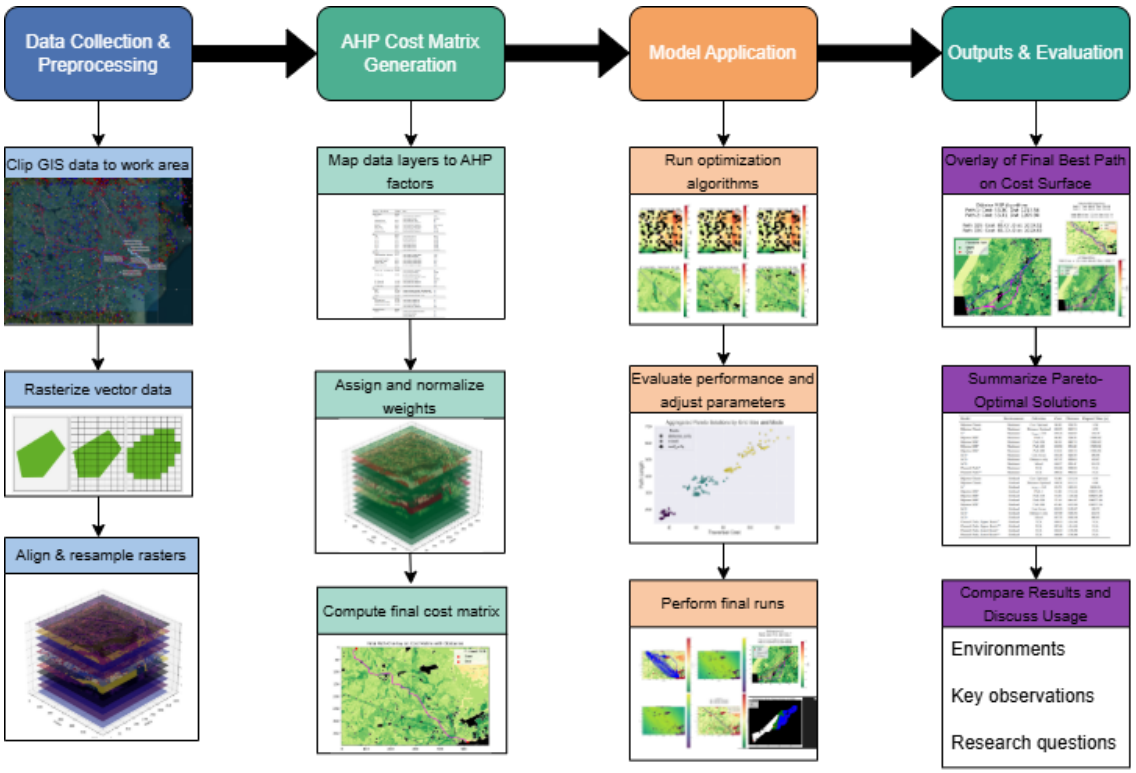


Figure 4.9: Schematic representation of the pipeline workflow of the project, from data collection and preprocessing to generated outputs and evaluation.

4.5 Computational Resources

All experiments were conducted on non-GPU laptops provided by WSP. While the runtimes reported in the results section are not representative of optimized execution environments, they are retained as indicative benchmarks. Even if minor relative to the overall lead times of cable routing projects, execution time remains

an important performance metric in AI-GIS path planning, serving as a baseline for future comparisons and scalability evaluations.

The reported **elapsed time** for each model is measured in a manner tailored to its operational logic. Though the models vary in complexity and search paradigm, the recorded times reflect representative computational demands. Details on how time was measured for each algorithm are provided below:

- **Dijkstra Classic:** Time is recorded from algorithm initiation to the identification of the shortest path under a single criterion (either cost or distance). This deterministic method yields a unique, optimal solution.
- **Dijkstra MSP:** Time reflects the duration required to generate the complete set of Pareto-optimal paths under bi-objective optimization. It scales significantly with environment size and is typically the most computationally intensive.
- **A*:** Time includes heuristic-driven exploration and path reconstruction. As a single-objective algorithm with pruning, A* often completes faster than Dijkstra MSP, though runtime depends on heuristic quality.
- **ACO:** As a metaheuristic involving a population of agents (ants), ACO iteratively refines solutions across generations. Time is measured from initialization to the end of all iterations, including solution aggregation into a Pareto front.

5

Results

The following sections present the results of the different models across the various environments. Tables summarizing the results for each environment are provided in the main text, while the paths of the individual models are shown in figures in the Appendix (Section A.1). The tables report the key metrics of interest on their respective environments - namely, the model name, selected criterion (e.g., cost or distance), total cost, total distance, and elapsed computation time.

It is important to note that the reported traversal cost and distance metrics do not represent real-world currency or physical length. Instead, they reflect accumulated cell costs and step lengths (including orthogonal and diagonal moves), enabling consistent comparison between paths within the grid-based framework. Normalization is only done on the aggregated results in Figure 5.6.

5.1 Initial Testing

As shown in Table 5.1, the shortest path of 29.21 cells and the lowest cost of 9.32 were both achieved by Dijkstra Classic, under different optimization criteria. Dijkstra MSP identified both of these extremal paths and also produced several intermediate trade-off solutions, effectively covering the Pareto front.

For the A* algorithm, varying the cost-heuristic weight w_{cost} resulted in little path variation, with most solutions concentrated near the shortest-path trajectory. Among the configurations, $w_{\text{cost}} = 1.0$ yielded the lowest traversal cost, though it still did not match the global cost minimum found by Dijkstra Classic or ACO.

The ACO model, run with 100 ants over 100 iterations, returned more diverse solutions. The cost-focused configuration achieved the lowest cost of all ACO runs, though it corresponded to the longest path. All three configurations converged to a common solution with a distance of 29.21 and cost of 11.90. The mixed configuration found intermediate values, while the distance-focused variant produced similarly short paths, with no major improvement in cost or efficiency.

In terms of runtime, Dijkstra and A* completed their executions in milliseconds, while ACO required several seconds per run due to its iterative, population-based nature.

Table 5.1: Summary of model results on the initial testing environment: cost, distance, and elapsed time. ACO was run using $q_0 = 0.99$, 100 ants for 100 iterations.

Model	Criterion	Cost	Distance	Elapsed Time (s)
Dijkstra Classic	Cost Optimal	9.32	39.90	0.0062
Dijkstra Classic	Distance Optimal	12.02	29.21	0.0033
A*	$w_{\text{cost}} = 0.0$	12.01	29.21	0.0496
A*	$w_{\text{cost}} = 0.111$	12.01	29.21	0.0496
A*	$w_{\text{cost}} = 0.222$	12.01	29.21	0.0496
A*	$w_{\text{cost}} = 0.333$	12.19	30.04	0.0496
A*	$w_{\text{cost}} = 0.444$	12.15	30.04	0.0496
A*	$w_{\text{cost}} = 0.555$	12.15	30.04	0.0496
A*	$w_{\text{cost}} = 0.666$	12.15	30.04	0.0496
A*	$w_{\text{cost}} = 0.777$	12.15	30.04	0.0496
A*	$w_{\text{cost}} = 0.888$	12.15	30.04	0.0496
A*	$w_{\text{cost}} = 1.0$	11.55	32.87	0.0496
Dijkstra MSP	Path 1	9.32	39.90	0.0109
Dijkstra MSP	Path 2	9.34	39.31	0.0109
Dijkstra MSP	Path 9	10.54	32.73	0.0109
Dijkstra MSP	Path 10	10.57	30.38	0.0109
ACO	Cost Focus	9.41	38.73	8.1614
ACO	Distance only	11.90	29.21	6.0993
ACO	Mixed	10.57	30.38	7.5297

5.2 Size-Varied Subgrid Analysis

The subgrid results showed that Dijkstra Classic consistently found the shortest and the cheapest paths across all grid sizes. The cost-optimal and distance-optimal paths differed noticeably, confirming that the two criteria led to distinct routing behaviors.

Dijkstra MSP identified the minimum-cost path in each subgrid but did not exactly recover the shortest path, although the distance difference remained small. The computed Pareto fronts were bounded between the cost-optimal and distance-optimal paths, with many paths overlapping or nearly identical—particularly in the 300×300 grid.

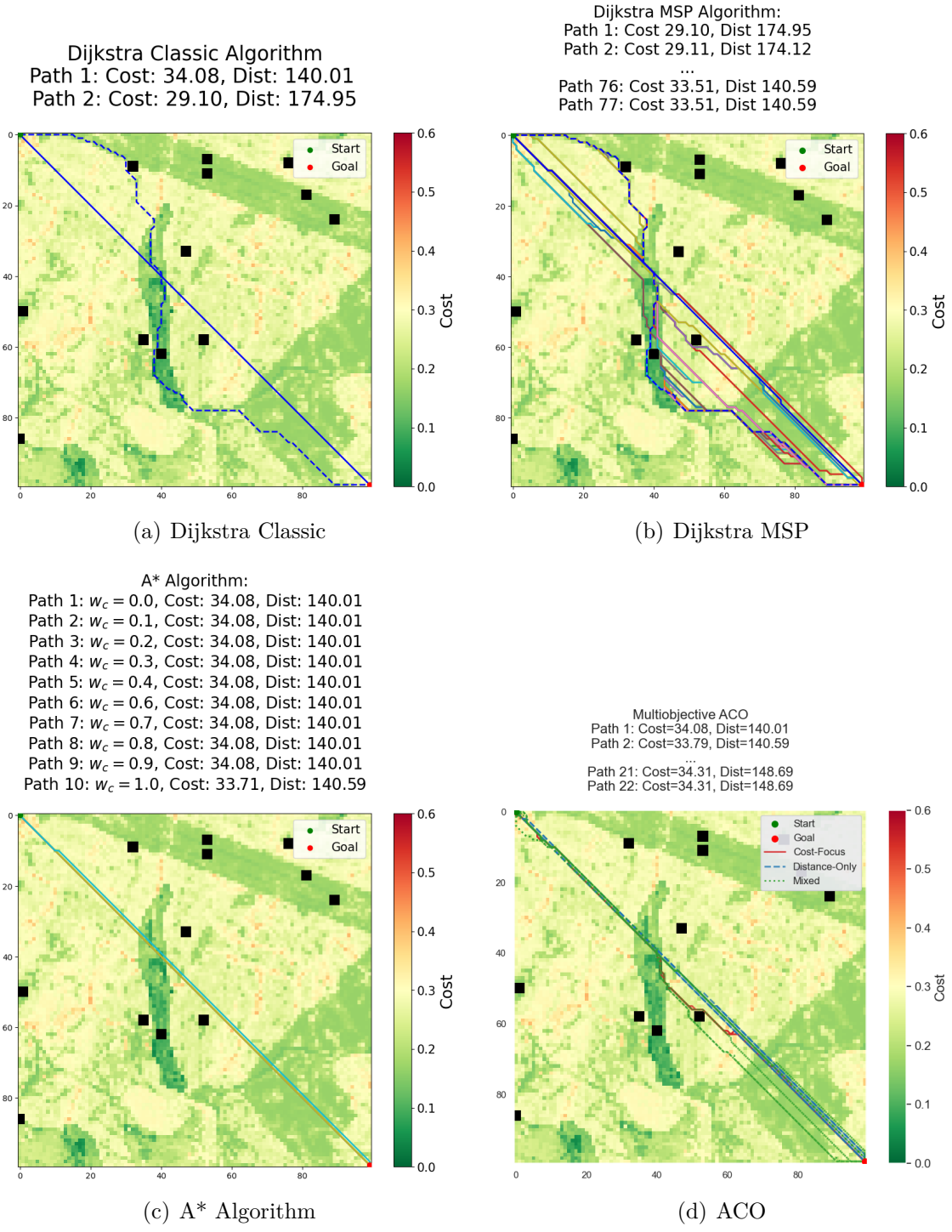
The A* algorithm exhibited minimal variation across the ten tested values of w_{cost}. Most resulting paths were similar and closely followed the shortest path. Among all tested values, $w_{\text{cost}} = 1.0$ achieved the lowest mean traversal cost, while $w_{\text{cost}} = 0.7$ produced one of the lowest distances and the second-best total cost.

ACO, configured with $q_0 = 0.99$, returned the lowest cost/distance coverage of Pareto-optimal paths, as shown in Figure A.9. The cost-focused setup produced the lowest cost overall, while the distance-focused mode yielded the shortest paths. The mixed configuration, as expected, found paths with intermediate cost and distance. Decision was made to further examine cost variation, when varying the β_1 parameter from 1 to 5, the influence of the cost heuristic increased, resulting in broader exploration patterns. The corresponding paths aligned closely with those generated by Dijkstra MSP.

Table 5.2: Summary of model results for different grid sizes: cost, distance, and elapsed time.

Model	Grid Size	Criterion	Cost	Distance	Elapsed Time (s)
Dijkstra Classic	100	Cost Optimal	29.10	174.95	0.1122
Dijkstra Classic	100	Distance Optimal	34.08	140.01	0.0781
A*	100	$w_{\text{cost}} = 0.0$	34.08	140.01	1.3491
A*	100	$w_{\text{cost}} = 0.111$	34.08	140.01	1.3491
A*	100	$w_{\text{cost}} = 0.222$	34.08	140.01	1.3491
A*	100	$w_{\text{cost}} = 0.333$	34.08	140.01	1.3491
A*	100	$w_{\text{cost}} = 0.444$	34.08	140.01	1.3491
A*	100	$w_{\text{cost}} = 0.555$	34.08	140.01	1.3491
A*	100	$w_{\text{cost}} = 0.666$	34.08	140.01	1.3491
A*	100	$w_{\text{cost}} = 0.777$	34.08	140.01	1.3491
A*	100	$w_{\text{cost}} = 0.888$	34.08	140.01	1.3491
A*	100	$w_{\text{cost}} = 1.0$	33.71	140.59	1.3491
Dijkstra MSP	100	Path 1	29.10	174.95	22.88
Dijkstra MSP	100	Path 26	29.81	157.58	22.88
Dijkstra MSP	100	Path 52	31.52	151.38	22.88
Dijkstra MSP	100	Path 77	33.51	140.59	22.88
ACO	100	Cost Focus	33.93	147.87	31.56
ACO	100	Distance Only	34.08	140.01	31.65
ACO	100	Mixed	33.26	149.04	41.44
Dijkstra Classic	200	Cost Optimal	41.17	324.72	0.39
Dijkstra Classic	200	Distance Optimal	61.49	284.94	0.25
A*	200	$w_{\text{cost}} = 0.0$	59.86	284.94	9.25
A*	200	$w_{\text{cost}} = 0.111$	59.86	284.94	9.25
A*	200	$w_{\text{cost}} = 0.222$	59.86	284.94	9.25
A*	200	$w_{\text{cost}} = 0.333$	59.86	285.77	9.25
A*	200	$w_{\text{cost}} = 0.444$	59.29	284.94	9.25
A*	200	$w_{\text{cost}} = 0.555$	59.26	284.94	9.25
A*	200	$w_{\text{cost}} = 0.666$	58.87	284.94	9.25
A*	200	$w_{\text{cost}} = 0.777$	58.85	284.94	9.25

Model	Grid Size	Criterion	Cost	Distance	Elapsed Time (s)
A*	200	$w_{\text{cost}} = 0.888$	58.81	284.94	9.25
A*	200	$w_{\text{cost}} = 1.0$	58.75	284.94	9.25
Dijkstra MSP	200	Path 1	41.17	324.72	124.17
Dijkstra MSP	200	Path 23	41.52	317.65	124.17
Dijkstra MSP	200	Path 47	42.06	302.52	124.17
Dijkstra MSP	200	Path 70	45.45	291.97	124.17
ACO	200	Cost Focus	55.11	313.14	27.88
ACO	200	Distance Only	60.13	284.94	52.41
ACO	200	Mixed	60.21	291.04	33.79
Dijkstra Classic	300	Cost Optimal	65.53	505.33	0.96
Dijkstra Classic	300	Distance Optimal	81.67	426.95	0.52
A*	300	$w_{\text{cost}} = 0.0$	80.61	427.78	71.02
A*	300	$w_{\text{cost}} = 0.111$	80.45	427.78	71.02
A*	300	$w_{\text{cost}} = 0.222$	80.53	427.78	71.02
A*	300	$w_{\text{cost}} = 0.333$	80.92	428.36	71.02
A*	300	$w_{\text{cost}} = 0.444$	81.03	428.36	71.02
A*	300	$w_{\text{cost}} = 0.555$	81.04	428.36	71.02
A*	300	$w_{\text{cost}} = 0.666$	81.07	428.36	71.02
A*	300	$w_{\text{cost}} = 0.777$	81.29	430.95	71.02
A*	300	$w_{\text{cost}} = 0.888$	81.28	430.95	71.02
A*	300	$w_{\text{cost}} = 1.0$	81.21	430.70	71.02
Dijkstra MSP	300	Path 1	65.53	505.33	3450.89
Dijkstra MSP	300	Path 178	65.94	486.98	3450.89
Dijkstra MSP	300	Path 356	66.32	476.06	3450.89
Dijkstra MSP	300	Path 534	74.14	432.81	3450.89
ACO	300	Cost Focus	89.64	529.80	63.21
ACO	300	Distance Only	86.40	426.95	87.55
ACO	300	Mixed	81.83	472.67	55.72



(a) Dijkstra Classic

(b) Dijkstra MSP

(c) A* Algorithm

(d) ACO

Figure 5.1: Results of the four different models on the 100×100 subgrid. The rest of the results are found in Section A.1.2.

5.3 Final Environments

The final environments, mainland and Gotland, represent the culmination of the study, allowing each model to be evaluated in a real-world, large-scale routing scenario. Table 5.3 presents the resulting traversal cost, distance, and elapsed time across all models, including planned paths. The associated figures showed the routes found by each algorithm overlaid on the cost surfaces.

Across both environments, Dijkstra Classic once again identified the shortest and cheapest paths, confirming its role as a baseline for performance evaluation. For the mainland environment, the shortest path had a length of 610.74 cells, and the cheapest path had a cost of 96.92. In the Gotland environment, the shortest and cheapest paths measured 1014.11 and 53.30, respectively.

The Dijkstra MSP model found both extremes and mapped out a wide range of trade-offs between them. Its results formed a continuous Pareto front, bounded by the shortest and cheapest paths identified by Dijkstra Classic.

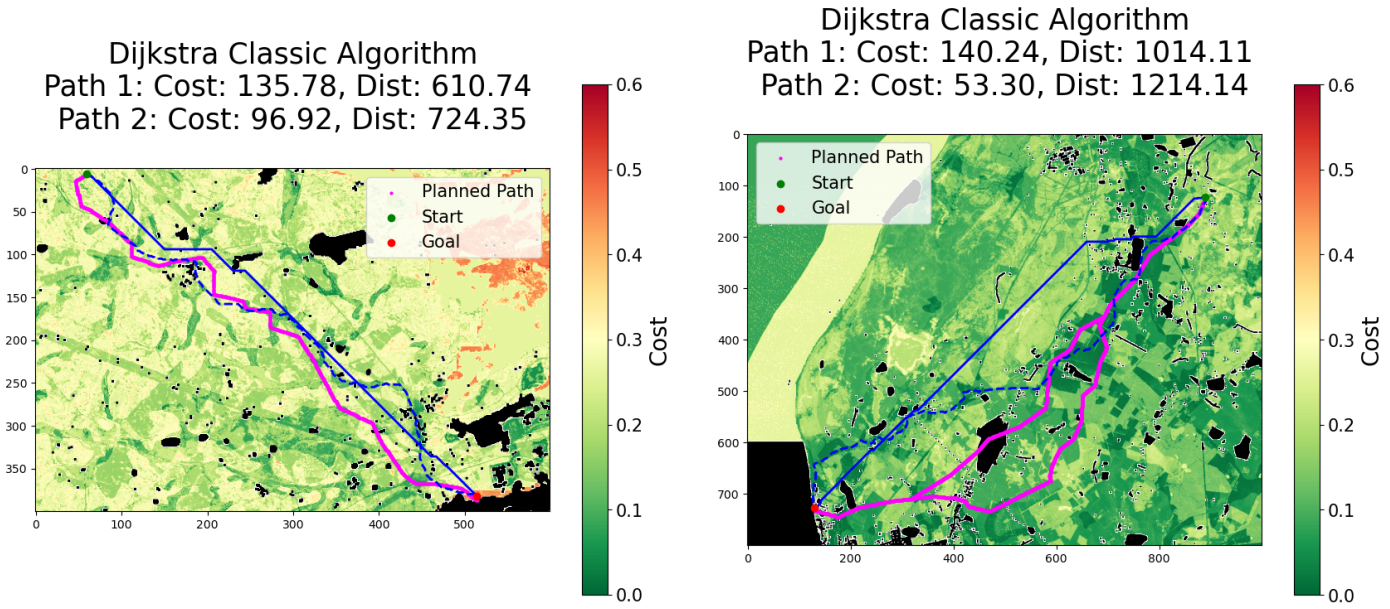
A* consistently identified near-optimal trade-off paths between distance and cost. For both environments, the A* solution was positioned close to the Pareto knee, indicating a balanced compromise. In terms of performance, it matched or surpassed ACO in cost and distance, although it required more computation time due to its exhaustive heuristic-based search.

ACO showed competitive performance with greater variation across configurations. In the mainland scenario, the mixed and distance-only configurations produced better cost-distance trade-offs than the cost-focused mode. In Gotland, the distance-only configuration again performed best among the ACO variants, achieving results comparable to A*.

The planned paths for both environments were significantly longer and more expensive than any of the model-generated paths. This reinforces the potential of the AI-based approach to reduce both material use and construction effort in practical applications.

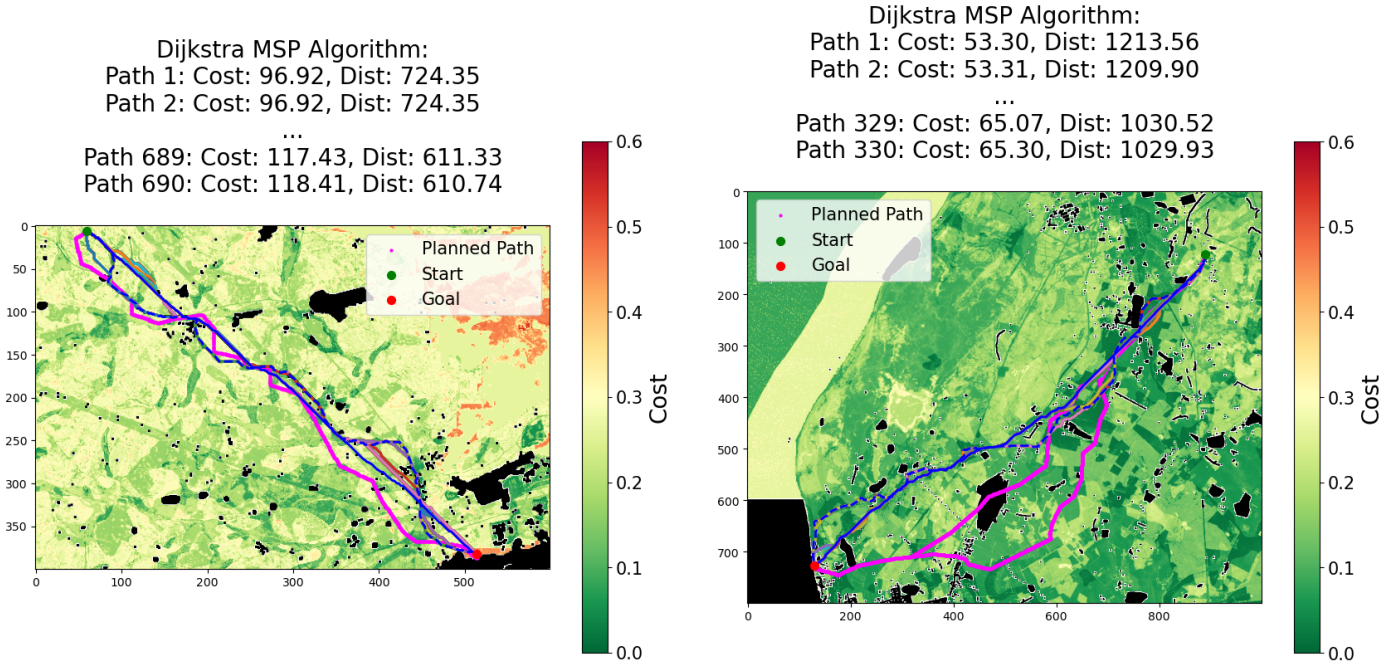
Table 5.3: Summary of model results for the mainland and Gotland environments: cost, distance, and elapsed time. For ACO, the configuration used was $q_0 = 0.99$ with 100 ants and 100 iterations. Planned path traversal costs are reported using two methods for obstacle handling: * assigns zero cost to obstacle cells, and ** assigns the mean cost of the matrix. UR denotes "Upper Route" and LR "Lower Route" for the Gotland planned path.

Model	Environment	Criterion	Cost	Distance	Elapsed Time (s)
Dijkstra Classic	Mainland	Cost Optimal	96.92	724.35	5.26
Dijkstra Classic	Mainland	Distance Optimal	135.78	610.74	3.79
A*	Mainland	$w_{\text{cost}} = 1$	129.00	613.82	184.79
Dijkstra MSP	Mainland	Path 1	96.92	724.35	17695.92
Dijkstra MSP	Mainland	Path 230	98.24	692.74	17695.92
Dijkstra MSP	Mainland	Path 460	100.50	671.22	17695.92
Dijkstra MSP	Mainland	Path 690	118.41	610.74	17695.92
ACO	Mainland	Cost focus	135.36	624.50	386.06
ACO	Mainland	Distance only	137.37	622.64	163.07
ACO	Mainland	Mixed	133.77	625.47	231.25
Planned Path*	Mainland	N/A	185.66	882.83	N/A
Planned Path**	Mainland	N/A	190.52	882.83	N/A
Dijkstra Classic	Gotland	Cost Optimal	53.30	1214.14	9.00
Dijkstra Classic	Gotland	Distance Optimal	140.24	1014.11	6.50
A*	Gotland	$w_{\text{cost}} = 1$	86.60	1061.27	1009.07
Dijkstra MSP	Gotland	Path 1	53.30	1213.56	196277.20
Dijkstra MSP	Gotland	Path 110	54.81	1128.22	196277.20
Dijkstra MSP	Gotland	Path 220	57.10	1084.07	196277.20
Dijkstra MSP	Gotland	Path 330	65.30	1029.93	196277.20
ACO	Gotland	Cost focus	104.76	1049.67	440.75
ACO	Gotland	Distance only	137.89	1020.84	224.75
ACO	Gotland	Mixed	107.78	1039.39	390.85
Planned Path, UR*	Gotland	N/A	109.13	1454.48	N/A
Planned Path, UR**	Gotland	N/A	127.44	1454.48	N/A
Planned Path, LR*	Gotland	N/A	133.10	1576.96	N/A
Planned Path, LR**	Gotland	N/A	140.00	1576.96	N/A



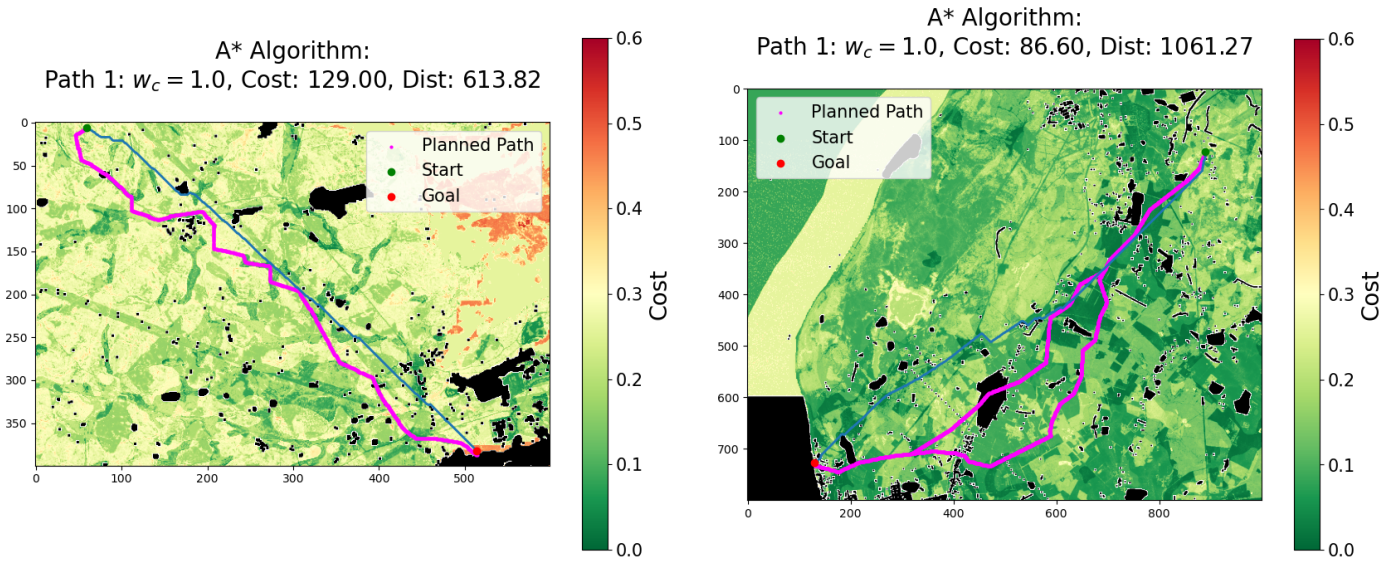
(a) Mainland (b) Gotland

Figure 5.2: Results of the Dijkstra Classic algorithm on the two final environments.



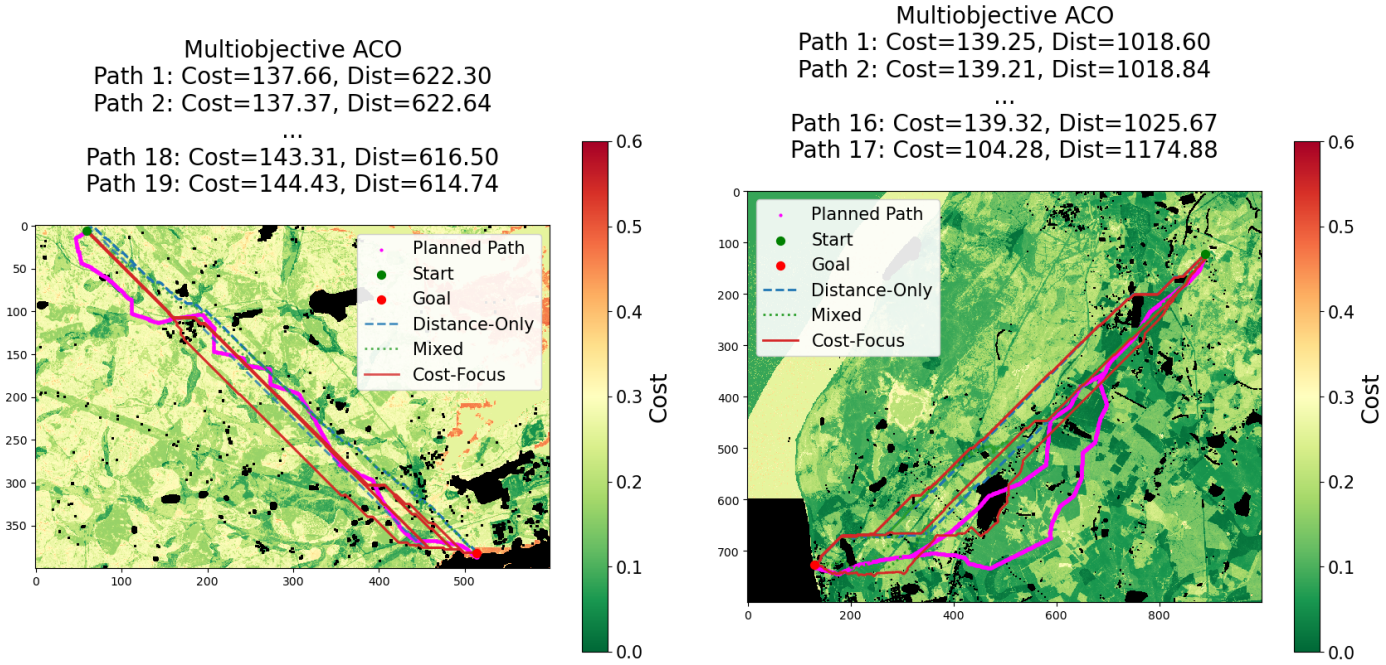
(a) Mainland (b) Gotland

Figure 5.3: Results of the Dijkstra MSP algorithm on the two final environments.



(a) Mainland (b) Gotland

Figure 5.4: Results of the A* algorithm on the two final environments.



(a) Mainland (b) Gotland

Figure 5.5: Results of the ACO algorithm on the two final environments.

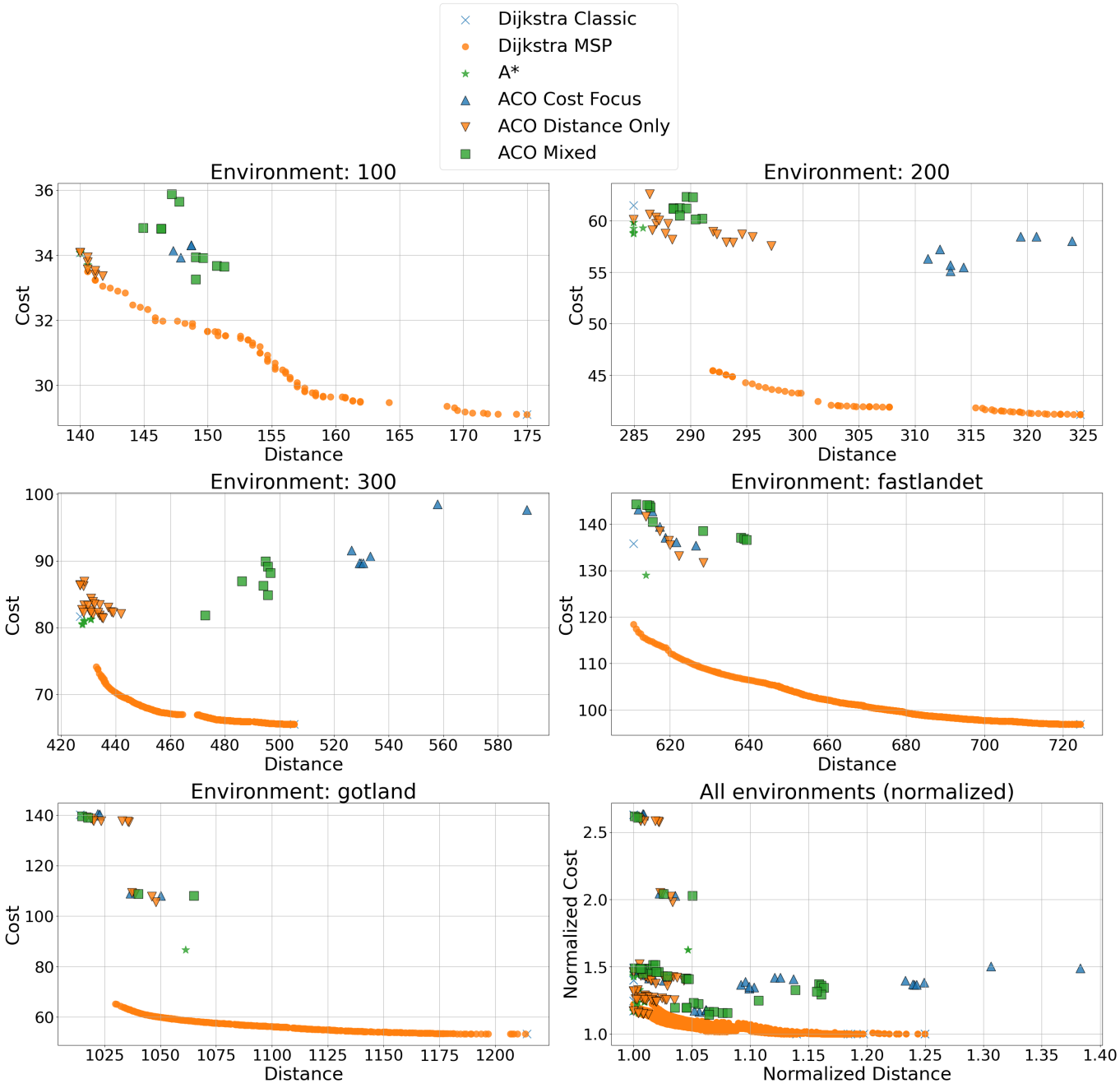


Figure 5.6: Performance comparison of all six model configurations across all test environments. Panels 1–5 plot raw Distance vs. Cost for each algorithm (Dijkstra Classic, Dijkstra MSP, A*, and three ACO strategies) on the five environments (100, 200, 300, Fastlandet [Mainland], Gotland). The bottom-right panel aggregates every run with both axes normalized to the Dijkstra-Classic baseline, highlighting each method's relative trade-off between path length and cost.

6

Discussion

6.1 Initial Testing

The results reinforce the strength of Dijkstra Classic in identifying both extremal solutions efficiently and validating the separation between cost and distance as distinct routing objectives.

Dijkstra MSP's ability to recover both ends of the cost-distance spectrum, along with a well-distributed set of intermediate paths, highlights its suitability for mapping Pareto-optimal solutions. Its behavior supports the conceptual framing of the routing task as a multi-objective problem.

A* shows limited flexibility in navigating the cost-distance trade-off, as even significant changes in w_{cost} do not result in meaningful path differences. While $w_{\text{cost}} = 1.0$ yields the best cost among A* configurations, its lack of global awareness restricts its exploration compared to ACO.

ACO demonstrates a clear advantage in trade-off exploration. Its ability to locate a shared high-quality compromise path across all modes (cost = 11.90, distance = 29.21) and to find the global cost minimum (cost = 9.41) illustrates its strength in broader search space coverage. Although slower, this flexibility is valuable for more complex routing environments where diverse objectives and constraints must be balanced.

6.2 Size-Varied Subgrid Analysis

The bounded nature of the Pareto fronts generated by Dijkstra MSP (consistently falling between the Dijkstra Classic baselines) validates the use of corridor search as a viable strategy for environment reduction. The high frequency of overlapping or similar Pareto paths indicates that optimal solutions are concentrated near these baselines, particularly in large grids, strengthening the case for search-space minimization.

While A* is efficient and easy to implement, its inability to significantly vary path selection based on heuristic weight highlights its limitation in handling trade-offs.

The performance analysis shows that $w_{\text{cost}} = 1.0$ provides the best overall trade-off, followed closely by $w_{\text{cost}} = 0.7$, but both remain distance-biased.

ACO demonstrates greater flexibility in exploring the cost-distance trade-off. Its ability to generate structurally distinct alternatives, particularly under varying β_1, shows potential for applications requiring nuanced balancing of routing criteria. This is especially relevant in complex or sensitive terrain, where cost-optimal paths may require deviation from distance-minimizing routes.

6.3 Final Environments

All models consistently outperform the planned paths in both traversal cost and distance. Despite their differing architectures, model outputs show little spatial correlation with the planned routes, suggesting a fundamental shift in methodology due to AHP-based cost surface generation. This divergence supports the project's hypothesis: automating the routing process with quantified constraints can yield better-performing solutions than expert-based manual planning.

Interestingly, certain planned path segments do align with model outputs - particularly in terrain with low traversal cost - indicating that the AHP-derived environment reasonably effectively captures relevant spatial characteristics. These overlaps occur across all models, further supporting the conclusion that environment setup, rather than model architecture, exerts a stronger influence on routing outcomes.

The Dijkstra MSP model, while computationally intensive, produces a dense Pareto front and offers unmatched coverage of the solution space. The increasing number of Pareto-optimal paths with environment size confirms that its use is more critical in larger or more complex terrain, while its outputs remain bounded by Dijkstra Classic baselines.

ACO continues to perform competitively, particularly in the *Distance Only* and *Mixed* configurations. The *Cost Focus* mode, however, often falls into local optima; an effect exacerbated by the environment's increasing scale. This behavior is explained by the growing influence of the distance-based heuristic as the goal becomes farther away. The *Distance Only* configuration proves to be the most robust, offering consistent and well-balanced trade-offs across environments.

A* demonstrates solid performance comparable to ACO Distance Only. It also benefits from deterministic path generation but suffers from higher runtime. A* excels in finding a consistent best compromise solution, even outperforming ACO in some larger environments. However, its single-path output limits diversity, which ACO provides through stochastic sampling.

6.4 Key Observations

6.4.1 ACO Behavior and Improvements

ACO's distance-focused and mixed modes yield better solutions than cost-focused in larger environments. This is due to the second heuristic's global nature, which minimizes unnecessary detours. In contrast, the first heuristic, based on local cost, is prone to local optima and inefficiency. The mixed mode offers a balance but does not always outperform distance-focused due to potential interference from suboptimal cost-focused agents. Improvements could include adaptive weighting, stronger global pheromone mechanisms, or real-time reinforcement learning for heuristic adaptation.

6.4.2 AI Efficiency Compared to Manual Routing

Model outputs consistently outperform planned routes, often by a significant margin. The structured use of AHP allows the models to factor in all major terrain constraints highlighted in literature and by WSP experts. Given that manual path design is time- and labor-intensive, and model generation is automated and faster, this confirms the scalability and practical advantage of the AI-GIS-AHP framework. However, it is important to note that the AHP-based data layers may not fully capture all the specific considerations applied by SVK in their planned routing decisions.

6.4.3 Environment Setup vs. Model Architecture

The alignment of model outputs in key terrain regions - regardless of model type and use of corridors - demonstrates that environment setup exerts greater influence than algorithmic architecture. The careful design of the cost surface via AHP, including subjective weightings and layer combinations, is thus a primary determinant of routing outcomes. This underscores the critical role of preprocessing and constraint modeling in optimization workflows.

6.4.4 Dijkstra MSP and the Role of Exhaustive Search

Dijkstra MSP demonstrates a clear advantage in producing complete Pareto fronts, especially in larger environments. However, its computational cost grows rapidly with grid size, limiting its practical use to small or corridor-restricted areas. The Classic Dijkstra model, by identifying extremal paths efficiently, provides a crucial baseline and corridor for bounding Pareto regions. Its inclusion in any routing workflow seems essential for speed and completeness.

6.4.5 ACO Heuristic Sensitivity and Exploration

ACO's performance is increasingly shaped by the problem scale. As the environment size increases, the second heuristic (goal proximity) dominates, promoting distance-optimized paths. This helps avoid costly local traps common in the cost-focused

configuration. The Distance Only setup consistently yields the most effective exploration and performance. This confirms the importance of balancing heuristic inputs and supports adaptive approaches in future research.

6.4.6 A* vs ACO: Determinism vs Diversity

A* proves reliable and consistent, especially in identifying best-compromise paths near the Pareto knee. However, it lacks the diversity of ACO, which, through repeated sampling, explores the solution space more broadly. Despite this, A* often outperforms ACO in individual cases, particularly in large-scale environments. This affirms that while ACO's stochastic nature allows broader exploration, it does not guarantee optimality without extensive sampling or parameter tuning.

6.5 Answering Research Questions

- **How can AI-driven optimization methods improve the efficiency of underground cable routing compared to traditional approaches?**

  By automating the conversion of complex geospatial data into cost surfaces and applying algorithms to evaluate possible paths efficiently. Compared to manual planning, AI reduces both time and labor requirements while expanding the solution space and improving result quality.

- **Which key factors/constraints can be used in route optimization?**

  Constraints such as route length, slope, environmental protection zones, land use, soil conditions, buffer zones, and crossing penalties were encoded into cost and obstacle matrices. These were then evaluated via the AHP framework to produce a scalar surface.

- **Which machine learning or optimization algorithms are best suited for optimizing cable routing while considering multiple constraints?**

  The results indicate that Dijkstra Classic and MSP are best for optimality and corridor definition, A* is suitable for deterministic trade-offs, and ACO offers flexible, approximate solutions with lower computational cost. Each is suited to different phases of a multi-stage pipeline.

- **How can one quantify the influence of model architecture versus environment setup on the final routing outcome, and to what extent does each component contribute to overall performance?**

  While difficult to quantify exactly, our findings show environment setup (via AHP) has a dominant impact. All models, regardless of architecture, often converged on similar routes when using the same cost surface, suggesting that terrain encoding outweighs architectural differences in determining outcomes.

6.5.1 Future Research

Future work can expand and refine the methodology presented in this thesis in several key directions:

- **Improved cost surface generation:** The AHP framework used in this study relies on expert-defined weights, which may be subjective. Future work could explore more systematic alternatives, such as supervised learning based on historical routing data or participatory methods involving multiple stakeholders. Additionally, a MCDA tool used at WSP, *GoldSET*, was briefly investigated. This tool may help streamline data preprocessing and could serve as a practical baseline for evaluating model performance, as it allows for manual route planning based on weighted criteria. It is expected to be adopted by the WSP's swedish cable routing team in the near future.
- **Adaptive heuristics in ACO:** The ACO algorithm's sensitivity to heuristic weighting, especially in large environments, suggests a need for dynamic or learning-based heuristic adjustments. Hybrid methods that incorporate reinforcement learning or metaheuristic tuning could improve performance. In particular, further investigation into the design of the second heuristic matrix and the adaptation of pheromone deposition strength based on performance trade-offs between distance and cost.
- **Exploration of alternative A* heuristics:** The A* heuristic used in this project relied primarily on Euclidean distance to the goal, limiting its ability to differentiate between cost and distance objectives. Future research should investigate composite heuristics that better capture both distance and local traversal cost in a weighted manner, enabling more diverse and responsive behavior across different weight settings.
- **Efficient Pareto front approximation:** While Dijkstra MSP is exhaustive and precise, its scalability is limited. Parallelization or approximate multi-objective search techniques could enable efficient Pareto front generation in larger domains.
- **Exploration of additional Multiobjective Shortest Path (MSP) algorithms:** While this project primarily investigated cost and distance as routing objectives, real-world planning often requires the simultaneous optimization of multiple, potentially conflicting factors such as regulatory zones, accessibility, and construction timelines. Future research should explore a broader set of MSP algorithms - both exact and heuristic - to better address such complexity. This includes exploring label-setting algorithms, metaheuristics, or revisiting multi-objective reinforcement learning approaches, which could enhance both the practical relevance and the diversity of generated solutions.
- **Modeling uncertainty and robustness:** Future studies should address terrain data uncertainty, changing land use, and incomplete information. Integrating stochastic modeling or scenario analysis can help produce more robust

and future-proof routing plans.

7

Conclusion

This thesis has demonstrated the feasibility and potential of integrating AI-driven optimization algorithms with GIS-based terrain modeling for underground cable routing. By translating geospatial constraints - such as slope, land cover, protected areas, and technical construction limitations - into a unified cost surface via the Analytic Hierarchy Process (AHP), we created an environment suitable for algorithmic pathfinding. This systematic approach enabled the application and evaluation of several pathfinding methods, namely Dijkstra Classic, Dijkstra MSP, A*, and Ant Colony Optimization (ACO), across both synthetic and real-world environments.

The results indicate that each algorithm fulfills a specific role. Dijkstra Classic efficiently finds optimal single-objective paths and provides reference baselines. Dijkstra MSP extends this to multi-objective optimization, producing exhaustive Pareto fronts but at a high computational cost. A* offers a deterministic middle ground, particularly useful when weighted trade-offs are preferred. Meanwhile, ACO - though stochastic and approximate - provides broad search capabilities and flexibility in balancing multiple objectives.

Across all evaluated environments, AI models consistently outperformed manually planned paths in terms of both traversal cost and distance. The results also revealed that the environment setup, especially the structure of the cost matrix, had a greater impact on routing outcomes than model architecture. This underscores the importance of accurate preprocessing, constraint weighting, and data integration during early project phases.

While this study has focused on three optimization techniques, the framework is adaptable and can be extended to include more advanced methods, such as hybrid algorithms. Future work may also include refining cost matrix generation using expert-in-the-loop or data-driven learning approaches, improving heuristic functions in ACO, or scaling to national or transnational routing problems.

In conclusion, the integration of AI and GIS offers a promising path forward for automated, scalable, and constraint-aware underground cable routing. With proper data preprocessing and model tuning, this approach has the potential to significantly reduce planning time and cost, while improving the quality and sustainability of infrastructure development.

References

[1] Brian Mackenzie. *How to optimise subsea power cable routing and design*. June 2021. URL: https://www.fugro.com/news/long-reads/2021/optimise-subsea-power-cable-routing-design-fugro-longread.

[2] Yuancun Qin et al. "Automatic optimization model of transmission line based on GIS and genetic algorithm". In: *Array* 17 (Mar. 2023), p. 100266. ISSN: 25900056. DOI: 10.1016/j.array.2022.100266. URL: https://www.sciencedirect.com/science/article/pii/S2590005622000996.

[3] Guojun Nan et al. "Smart line planning method for power transmission based on D3QN-PER algorithm". In: *IET Control Theory and Applications* (Nov. 2024). ISSN: 17518652. DOI: 10.1049/cth2.12689. URL: https://ietresearch.onlinelibrary.wiley.com/doi/epdf/10.1049/cth2.12689.

[4] Song Guanfu. *What is AI GIS (Artificial Intelligence GIS) ? - SuperMap*. Feb. 2020. URL: https://www.supermap.com/en-us/news/?82_2701.html.

[5] Tomasz Smolarczyk. *GIS and artificial intelligence: what is GeoAI? - Spyrosoft*. Mar. 2024. URL: https://spyro-soft.com/blog/geospatial/gis-and-artificial-intelligence-what-is-geoai.

[6] IGNESA. *How AI is Revolutionizing Geospatial Analysis - IGNESA - The GeoAI Company*. July 2024. URL: https://ignesa.com/insights/how-ai-is-revolutionizing-geospatial-analysis/.

[7] Svenska Kraftnät. *Gotlandsförbindelsen*. May 2023. URL: https://www.svk.se/utveckling-av-kraftsystemet/transmissionsnatet/transmissionsnatsprojekt/gotland/.

[8] Svenska Kraftnät. *Digital samrådskarta Gotlandsförbindelsen v2*. 2024. URL: %https://experience.arcgis.com/experience/c4157a5084334f5d88b9191ca7f3fe75.

[9] Brent Daniel Mittelstadt et al. "The ethics of algorithms: Mapping the debate". In: *Big Data & Society* 3.2 (2016). DOI: 10.1177/2053951716679679. URL: https://doi.org/10.1177/2053951716679679.

[10] Anna Jobin, Marcello Ienca, and Effy Vayena. "The global landscape of AI ethics guidelines". In: *Nature Machine Intelligence* 1.9 (2019), pp. 389–399. ISSN: 2522-5839. DOI: 10.1038/s42256-019-0088-2. URL: https://doi.org/10.1038/s42256-019-0088-2.

[11] Luciano Floridi et al. “AI4People—An Ethical Framework for a Good AI Society: Opportunities, Risks, Principles, and Recommendations”. In: *Minds and Machines* 28.4 (2018), pp. 689–707. ISSN: 1572-8641. DOI: 10.1007/s11023-018-9482-5. URL: https://doi.org/10.1007/s11023-018-9482-5.

[12] *World Energy Outlook Special Report Energy and AI*. Tech. rep. International Energy Agency. URL: www.iea.org/terms.

[13] *Sustainable Infrastructure for the SDGs - Nexus Dialogue*. Feb. 2019. URL: https://unemg.org/sustainable-infrastructure-for-the-sdgs/.

[14] David Rolnick et al. “Tackling Climate Change with Machine Learning”. In: *ACM Computing Surveys* 55.2 (June 2019). ISSN: 15577341. DOI: 10.1145/3485128. URL: https://arxiv.org/pdf/1906.05433.

[15] Bin Wang et al. “Research on transmission line path planning model based on TFN-AHP and ACO”. In: *IET Generation, Transmission & Distribution* 18.13 (July 2024), pp. 2373–2381. ISSN: 1751-8687. DOI: 10.1049/gtd2.13208. URL: https://ietresearch.onlinelibrary.wiley.com/doi/full/10.1049/gtd2.13208.

[16] Li Wang et al. “Transmission line route planning based on AHP-ACO algorithm”. In: *Journal of Physics: Conference Series* 2310.1 (Oct. 2022), p. 012014. ISSN: 1742-6588. DOI: 10.1088/1742-6596/2310/1/012014. URL: https://www.researchgate.net/publication/364482688_Transmission_line_route_planning_based_on_AHP-ACO_algorithm.

[17] Dong Yang et al. “Transmission Line Planning Based on Artificial Intelligence in Smart Cities”. In: *Mobile Information Systems* 2022 (Mar. 2022), pp. 1–8. ISSN: 1875-905X. DOI: 10.1155/2022/2010189. URL: https://onlinelibrary.wiley.com/doi/10.1155/2022/2010189?msockid=34678e232b1a654110c39b502a956459.

[18] Rodolfo Mendes de Lima et al. “Least-cost path analysis and multi-criteria assessment for routing electricity transmission lines”. In: *IET Generation, Transmission & Distribution* 10.16 (Dec. 2016), pp. 4222–4230. ISSN: 1751-8687. DOI: 10.1049/iet-gtd.2016.1119. URL: https://ietresearch.onlinelibrary.wiley.com/doi/10.1049/iet-gtd.2016.1119.

[19] Kelin Long. “Optimization of high voltage transmission line path based on AI”. In: *2022 IEEE 2nd International Conference on Power, Electronics and Computer Applications (ICPECA)*. IEEE, Jan. 2022, pp. 1138–1141. ISBN: 978-1-6654-4276-3. DOI: 10.1109/ICPECA53709.2022.9718951. URL: https://ieeexplore.ieee.org/document/9718951.

[20] Nawa Raj Chapagain et al. “TRANSMISSION LINE ROUTING USING GIS TOOLS OF SPATIAL SCIENCES”. In: Dec. 2020. DOI: 10.13033/isahp.y2020.067. URL: https://www.researchgate.net/publication/347991409_TRANSMISSION_LINE_ROUTING_USING_GIS_TOOLS_OF_SPATIAL_SCIENCES.

[21] Faizah Husain et al. “A study on TNB transmission line route sustainability and suitability using GIS-AHP”. In: *2012 IEEE Control and System Graduate Research Colloquium*. IEEE, July 2012, pp. 364–369. ISBN: 978-1-4673-2036-8. DOI: 10.1109/ICSGRC.2012.6287193. URL: https://ieeexplore.ieee.org/document/6287193.

[22] Kimlin Saing et al. “Revolutionizing energy infrastructure: Automated route planning for underground transmission lines in Phnom Penh”. In: *e-Prime - Advances in Electrical Engineering, Electronics and Energy* 9 (Sept. 2024). ISSN: 27726711. DOI: 10.1016/j.prime.2024.100633. URL: https://www.sciencedirect.com/science/article/pii/S2772671124002134.

[23] C. D. Viswarani et al. “Optimization on shortest path finding for underground cable transmission lines routing using GIS”. In: *Journal of Theoretical and Applied Information Technology* 65 (Jan. 2014), pp. 639–643. URL: https://www.researchgate.net/publication/287362427_Optimization_on_shortest_path_finding_for_underground_cable_transmission_lines_routing_using_GIS.

[24] Zhuo Liu et al. “A Transmission Line Planning Method Based on Variable Size Grid and Improved Ant Colony Optimization Algorithm”. In: *2023 38th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*. IEEE, Aug. 2023, pp. 838–843. ISBN: 979-8-3503-0363-6. DOI: 10.1109/YAC59482.2023.10401803. URL: https://ieeexplore.ieee.org/abstract/document/10401803.

[25] Yazhou Fan. “The Research of Path Planning for Transmission Network Based on Improved Ant Colony Algorithms”. In: *Journal of Physics: Conference Series* 1345.5 (Nov. 2019), p. 052074. ISSN: 1742-6588. DOI: 10.1088/1742-6596/1345/5/052074. URL: https://www.researchgate.net/publication/337601132_The_Research_of_Path_Planning_for_Transmission_Network_Based_on_Improved_Ant_Colony_Algorithms.

[26] Yun Cao et al. “A Comparation Study of Transmission Line Routing Based on A* and RRT Algorithms”. In: *Journal of Computer Science Research* 6.3 (July 2024), pp. 10–16. ISSN: 2630-5151. DOI: 10.30564/jcsr.v6i3.6714. URL: https://journals.bilpubgroup.com/index.php/jcsr/article/view/6714/5360.

[27] Hasan Eroğlu and Musa Aydin. “Solving power transmission line routing problem using improved genetic and artificial bee colony algorithms”. In: *Electrical Engineering* 100.3 (Sept. 2018), pp. 2103–2116. ISSN: 0948-7921. DOI: 10.1007/s00202-018-0688-6. URL: https://link.springer.com/content/pdf/10.1007/s00202-018-0688-6.pdf.

[28] Hasan EROĞLU and Musa AYDIN. “Optimization of electrical power transmission lines’ routing using AHP, fuzzy AHP, and GIS”. In: *TURKISH JOURNAL OF ELECTRICAL ENGINEERING & COMPUTER SCIENCES* 23 (2015), pp. 1418–1430. ISSN: 13000632. DOI: 10.3906/elk-1211-59. URL: https://journals.tubitak.gov.tr/elektrik/vol23/iss5/17/.

[29] Monica Bhavani M. and A Valarmathi. "Optimal Traffic Route Finder System". In: *Reliability and Risk Assessment in Engineering*. Ed. by V. Gupta et al. Singapore: Springer Singapore, 2020, pp. 39–47. ISBN: 978-981-15-3746-2. URL: https://link.springer.com/chapter/10.1007/978-981-15-3746-2_4#citeas.

[30] Antonio Sedeño-noda and Marcos Colebrook. "A biobjective Dijkstra algorithm". In: *European Journal of Operational Research* 276.1 (July 2019), pp. 106–118. ISSN: 0377-2217. DOI: 10.1016/J.EJOR.2019.01.007.

[31] Pedro Maristany de las Casas, Antonio Sedeño-Noda, and Ralf Borndörfer. "An Improved Multiobjective Shortest Path Algorithm". In: *Computers & Operations Research* 135 (Nov. 2021), p. 105424. ISSN: 0305-0548. DOI: 10.1016/J.COR.2021.105424.

[32] T. Karlsson et al. "Automatic Cable Harness Layout Routing in a Customizable 3D Environment". In: *CAD Computer Aided Design* 169 (Apr. 2024). ISSN: 00104485. DOI: 10.1016/j.cad.2023.103671.

[33] Sadhana H. Barkund, Ashutosh Sharma, and H. R Bhapkar. "Survey of Shortest Path Algorithms". In: *International Journal of Renewable Energy Exchange* 10.11 (Nov. 2022), pp. 46–57. DOI: 10.58443/IJREX.10.11.2022.46-57.

[34] Robert W. Floyd. "Algorithm 97: Shortest path". In: *Communications of the ACM* 5.6 (June 1962), p. 345. ISSN: 15577317. DOI: 10.1145/367766.368168.

[35] Massimo Meneghello and Andrea Tommasi. *Wind Farm Cables Routing Optimization*. Mar. 2019. URL: https://github.com/caerbannogwhite/cables-routing-optimization/blob/master/report/report.pdf.

[36] Martina Fischetti and David Pisinger. "Optimizing wind farm cable routing considering power losses". In: *European Journal of Operational Research* 270.3 (Nov. 2018), pp. 917–930. ISSN: 03772217. DOI: 10.1016/j.ejor.2017.07.061.

[37] A B Conru. "A genetic approach to the cable harness routing problem". In: *Proceedings of the First IEEE Conference on Evolutionary Computation. IEEE World Congress on Computational Intelligence*. 1994, pp. 200–205. DOI: 10.1109/ICEC.1994.350016.

[38] Yingfeng Zhao et al. "Multi-Branch Cable Harness Layout Design Based on Genetic Algorithm with Probabilistic Roadmap Method". In: *Chinese Journal of Mechanical Engineering (English Edition)* 34.1 (Dec. 2021). ISSN: 21928258. DOI: 10.1186/s10033-021-00544-9.

[39] Dan Zhang et al. "Multi-branch Cable Harness Layout Optimization Based on Pattern Search Improved Particle Swarm Optimization Algorithm". In: *Mechanisms and Machine Science*. Vol. 111. Springer Science and Business Media B.V., 2022, pp. 1255–1273. ISBN: 9789811673801. DOI: 10.1007/978-981-16-7381-8{_}78.

[40] Dan Zhang et al. “Multi-objective layout optimization of aircraft multi-branch cable harness based on MOPSO/D”. In: *2021 12th International Conference on Mechanical and Aerospace Engineering, ICMAE 2021*. Institute of Electrical and Electronics Engineers Inc., July 2021, pp. 139–144. ISBN: 9781665433211. DOI: 10.1109/ICMAE52228.2021.9522568.

[41] Qiang Liu and Chengen Wang. “Multi-terminal pipe routing by Steiner minimal tree and particle swarm optimisation”. In: *Enterprise Information Systems* 6.3 (Aug. 2012), pp. 315–327. ISSN: 17517575. DOI: 10.1080/17517575.2011.594910.

[42] 1000minds. *What is the Analytic Hierarchy Process (AHP)?* URL: https://www.1000minds.com/decision-making/analytic-hierarchy-process-ahp#analytic-hierarchy-process-ahp-a-five-step-overview.

[43] Qiang Luo et al. “Research on path planning of mobile robot based on improved ant colony algorithm”. In: *Neural Computing and Applications* 32.6 (Mar. 2020), pp. 1555–1566. ISSN: 14333058. DOI: 10.1007/s00521-019-04172-2.

[44] Zhuqing Jiao et al. “A path planning method using adaptive polymorphic ant colony algorithm for smart wheelchairs”. In: *Journal of Computational Science* 25 (Mar. 2018), pp. 50–57. ISSN: 18777503. DOI: 10.1016/j.jocs.2018.02.004.

[45] Zhikang T. Wang and Masahito Ueda. “Convergent and Efficient Deep Q Network Algorithm”. In: (June 2021). URL: http://arxiv.org/abs/2106.15419.

[46] R.W. Saaty. “The analytic hierarchy process—what it is and how it is used”. In: *Mathematical Modelling* 9.3-5 (1987), pp. 161–176. ISSN: 02700255. DOI: 10.1016/0270-0255(87)90473-8. URL: https://www.sciencedirect.com/science/article/pii/0270025587904738.

[47] Jeremy Y. L. Yap, Chiung Ching Ho, and Choo-Yee Ting. “Analytic hierarchy process (AHP) for business site selection”. In: 2018, p. 020151. DOI: 10.1063/1.5055553. URL: https://www.researchgate.net/publication/316736276_Analytic_Hierarchy_Process_AHP_for_business_site_selection.

[48] Thomas H. Cormen et al. *Introduction To Algorithms Third Edition (2009)*. Third Edition. 2009, pp. 657–660. URL: https://archive.org/details/introduction-to-algorithms-third-edition-2009/page/657/mode/2up.

[49] Muhammad Adeel Javaid. “Understanding Dijkstra Algorithm”. In: *SSRN Electronic Journal* (2013). ISSN: 1556-5068. DOI: 10.2139/ssrn.2340905. URL: https://www.researchgate.net/publication/273264449_Understanding_Dijkstra_Algorithm.

[50] Yuandong Chen et al. “Research on the A* Algorithm for Automatic Guided Vehicles in Large-Scale Maps”. In: *Applied Sciences (Switzerland)* 14.22 (Nov. 2024). ISSN: 20763417. DOI: 10.3390/app142210097.

[51] Mattias Wahde. *Biologically Inspired Optimization Methods: An Introduction*. 1st edition. WIT Press, 2008, pp. 100–114. ISBN: 9781845641481.

[52] Keivan Ghoseiri and Behnam Nadjari. “An ant colony optimization algorithm for the bi-objective shortest path problem”. In: *Applied Soft Computing* 10.4 (Sept. 2010), pp. 1237–1246. ISSN: 15684946. DOI: 10.1016/j.asoc.2009.09.014. URL: https://www.sciencedirect.com/science/article/pii/S1568494609001963.

[53] ThePro3DStudio. *7 Different Types of 3D Rendering Techniques Used in 3D Art*. June 2023. URL: https://professional3dservices.com/blog/3d-rendering-techniques.html.

[54] Lantmäteriet. *Markhöjdmodell Nedladdning, grid 50+*. May 2022. URL: https://www.lantmateriet.se/sv/geodata/vara-produkter/produktlista/markhojdmodell-nedladdning-grid-50/.

[55] SMHI. *Utforskaren - Öppna data*. June 2020. URL: https://www.smhi.se/data/utforskaren-oppna-data.

[56] Skogsstyrelsen. *Ladda ner geodata*. Jan. 2025. URL: https://www.skogsstyrelsen.se/sjalvservice/karttjanster/geodatatjanster/nerladdning-av-geodata.

[57] SGU. *Jordartsdata*. May 2024. URL: https://www.sgu.se/produkter-och-tjanster/geologiska-data/jordarter--geologisk-data/jordartsdata/#1-1.

[58] Naturvårdsverket. *Geodatakatalogen*. URL: https://geodatakatalogen.naturvardsverket.se/geonetwork/srv/swe/catalog.search#/search.

[59] Post- och telestyrelsen (PTS). *LEDNINGSKOLLEN*. Swedish. URL: https://www.ledningskollen.se/Det-har-ar-Ledningskollen.

[60] Andrew J Schmidt. “Implementing a GIS Methodology for Siting High Voltage Electric Transmission Lines”. In: *Saint Mary’s University of Minnesota University Central Services Press* 11 (2009). URL: https://gis.smumn.edu/GradProjects/SchmidtA.pdf.

[61] V. Yildirim et al. “Natural Gas Transmission Pipeline Route Selection Using GIS and AHP”. In: (2013). URL: https://www.researchgate.net/publication/312289377_Natural_Gas_Transmission_Pipeline_Route_Selection_Using_GIS_and_AHP.

[62] V. Yildirim, T. Yomralioglu, and R. Nisanci. “A Raster Based Geospatial Model for Natural Gas Transmission Line Routing”. In: *Natural Gas - Extraction to End Use*. InTech, Oct. 2012. DOI: 10.5772/45814. URL: https://www.researchgate.net/publication/273452747_A_Raster_Based_Geospatial_Model_for_Natural_Gas_Transmission_Line_Routing.

[63] PyTorch. *Reinforcement Learning (DQN) Tutorial*. Mar. 2017. URL: https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html.

[64] GeeksforGeeks. *Q-Learning in Reinforcement Learning*. Feb. 2025. URL: https://www.geeksforgeeks.org/q-learning-in-python/.

[65] GeeksforGeeks. *Deep Q-Learning in Reinforcement Learning*. Jan. 2023. URL: https://www.geeksforgeeks.org/deep-q-learning/.

[66] Tom Schaul et al. "Prioritized Experience Replay". In: *ICLR* (2016). URL: https://arxiv.org/abs/1511.05952.

[67] Haojie Wang et al. "Landslide identification using machine learning". In: *Geoscience Frontiers* 12.1 (Jan. 2021), pp. 351–364. ISSN: 16749871. DOI: 10.1016/j.gsf.2020.02.012.

[68] Kounghoon Nam and Fawu Wang. "The performance of using an autoencoder for prediction and susceptibility assessment of landslides: A case study on landslides triggered by the 2018 Hokkaido Eastern Iburi earthquake in Japan". In: *Geoenvironmental Disasters* 6.1 (2019), p. 19. ISSN: 2197-8670. DOI: 10.1186/s40677-019-0137-5. URL: https://doi.org/10.1186/s40677-019-0137-5.

[69] Yange Li et al. "Review on the artificial intelligence-based methods in landslide detection and susceptibility assessment: Current progress and future directions". In: *Intelligent Geoengineering* 1.1 (Dec. 2024), pp. 1–18. ISSN: 30506190. DOI: 10.1016/j.ige.2024.10.003. URL: https://linkinghub.elsevier.com/retrieve/pii/S305061902400003X.

[70] Faming Huang et al. "A deep learning algorithm using a fully connected sparse autoencoder neural network for landslide susceptibility prediction". In: *Landslides* 17.1 (2020), pp. 217–229. ISSN: 1612-5118. DOI: 10.1007/s10346-019-01274-9. URL: https://doi.org/10.1007/s10346-019-01274-9.

[71] Chitta Ranjan. *Build the right Autoencoder-Tune and Optimize using PCA principles. Part I*. July 2019. URL: https://medium.com/data-science/build-the-right-autoencoder-tune-and-optimize-using-pca-principles-part-i-1f01f821999b.

[72] Chitta Rajan. *Build the right Autoencoder-Tune and Optimize using PCA principles. Part II*. July 2019. URL: https://medium.com/@cran2367/build-the-right-autoencoder-tune-and-optimize-using-pca-principles-part-ii-24b9cca69bd6.

[73] *Phi coefficient*. URL: https://en.wikipedia.org/wiki/Phi_coefficient.

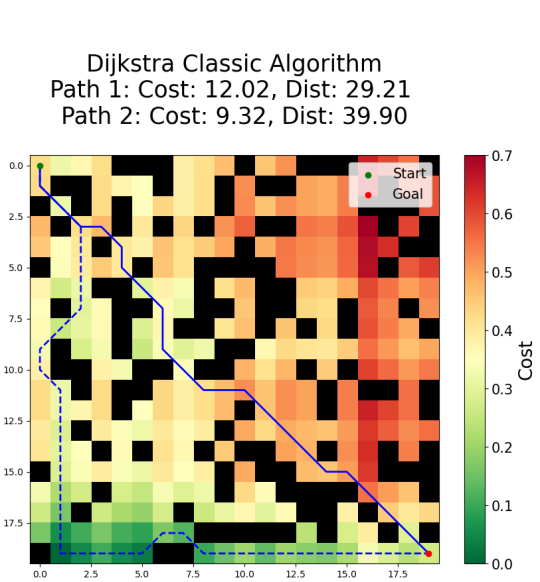
[74] *Metrics: Matthew's correlation coefficient*. URL: https://thedatascientist.com/metrics-matthews-correlation-coefficient/.

A

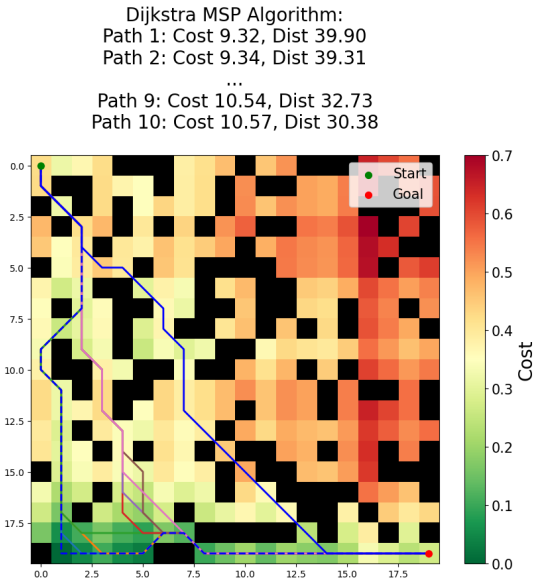
Appendix

A.1 Plots from Results

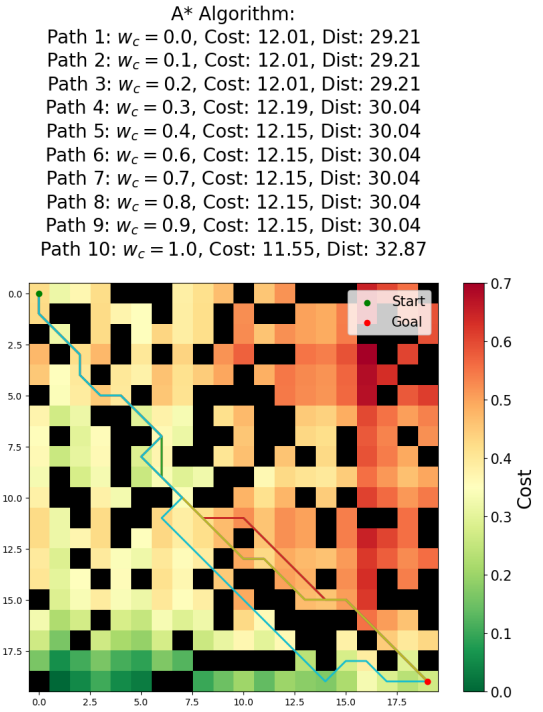
A.1.1 Initial Testing



(a) Final path of Dijkstra Classic algorithm.

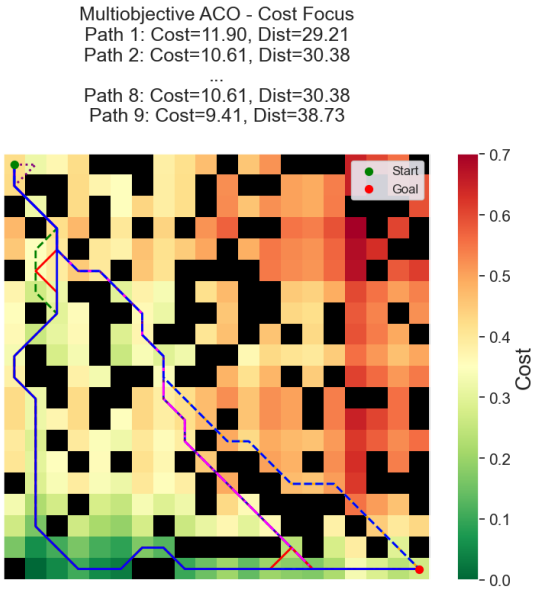


(b) Final path of Dijkstra MSP algorithm.

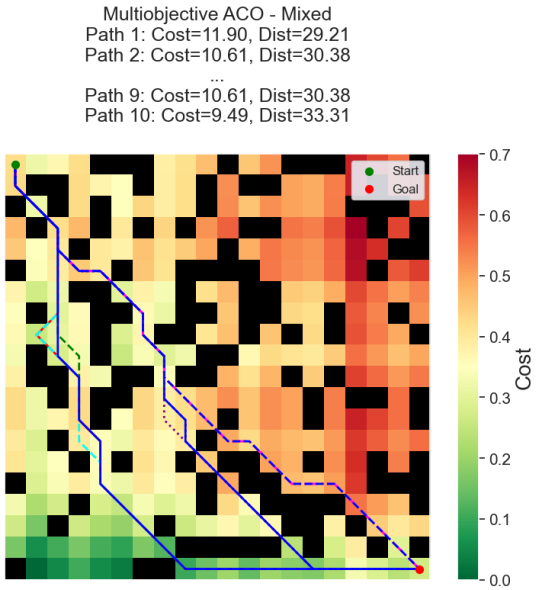


(c) Final path of A* algorithm.

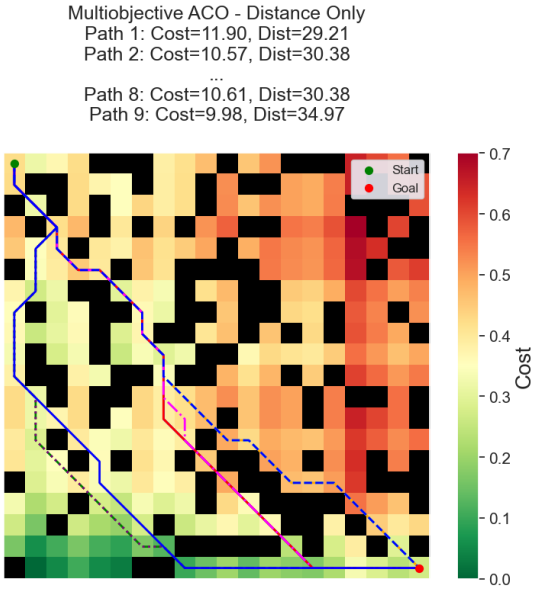
Figure A.1: Results of the Dijkstra and A* algorithms on the initial testing environment, of size 20×20 cells.



(a) Cost-Focus Mode ($a = -1$, $b = \frac{N}{2}$): for $h < b$, $\lambda < 1$ gives a mixed effect, but once $h \geq b$, $\lambda = 1$ so ants focus entirely on minimizing cost.



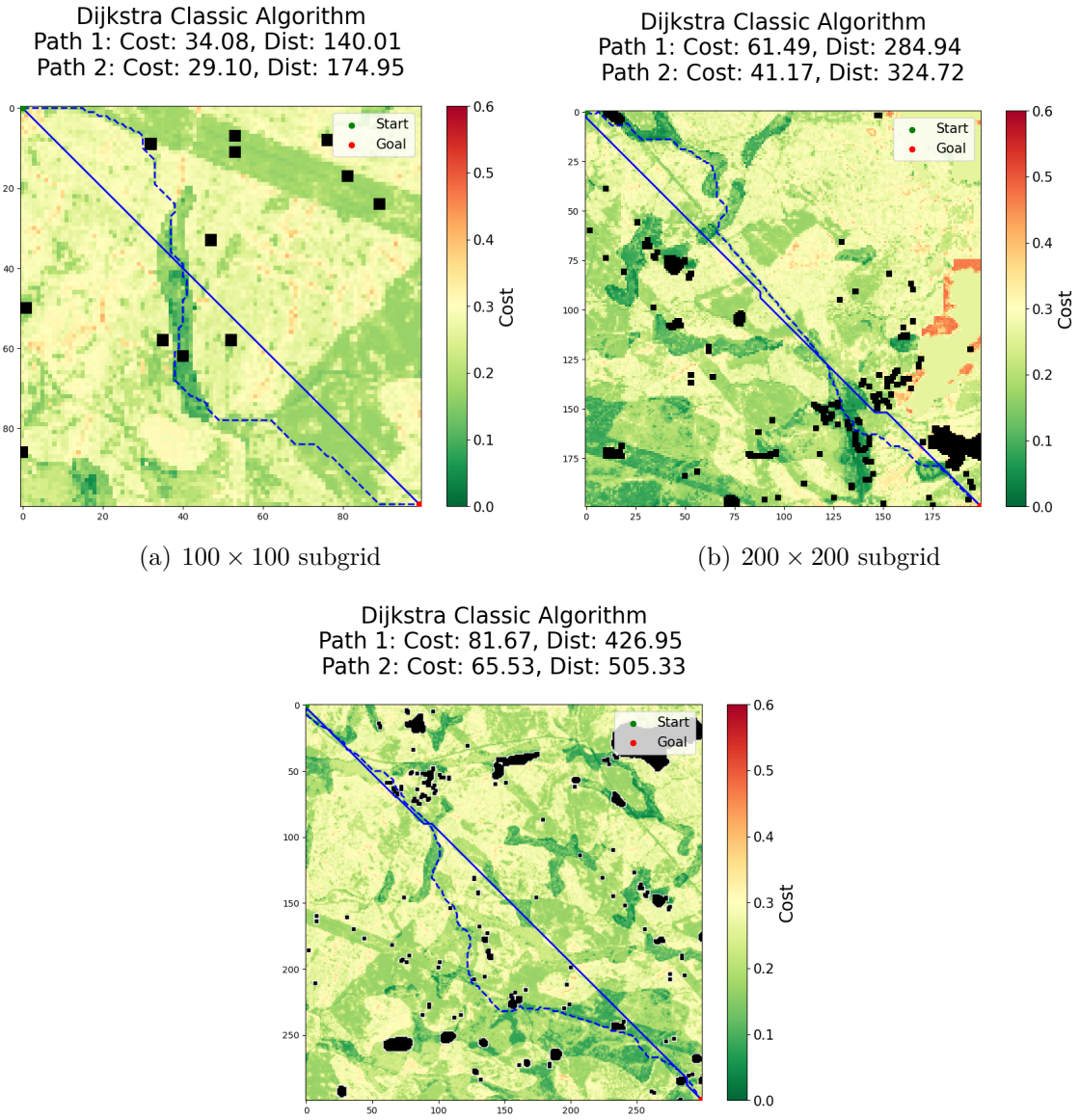
(b) Mixed Mode ($a = -1$, $b = N + 1$): since $a < h < b$ for all ants, λ varies smoothly between 0 and 1, balancing cost vs. distance.



(c) Distance-Only Mode ($a = N$, $b = N + 1$): here $h \leq a$ for every ant, so $\lambda = 0$ and the algorithm optimizes solely for shortest-path distance.

Figure A.2: Results of the ACO algorithm on the initial testing environment, of size 20×20 cells.

A.1.2 Size-Varied Subgrid Analysis

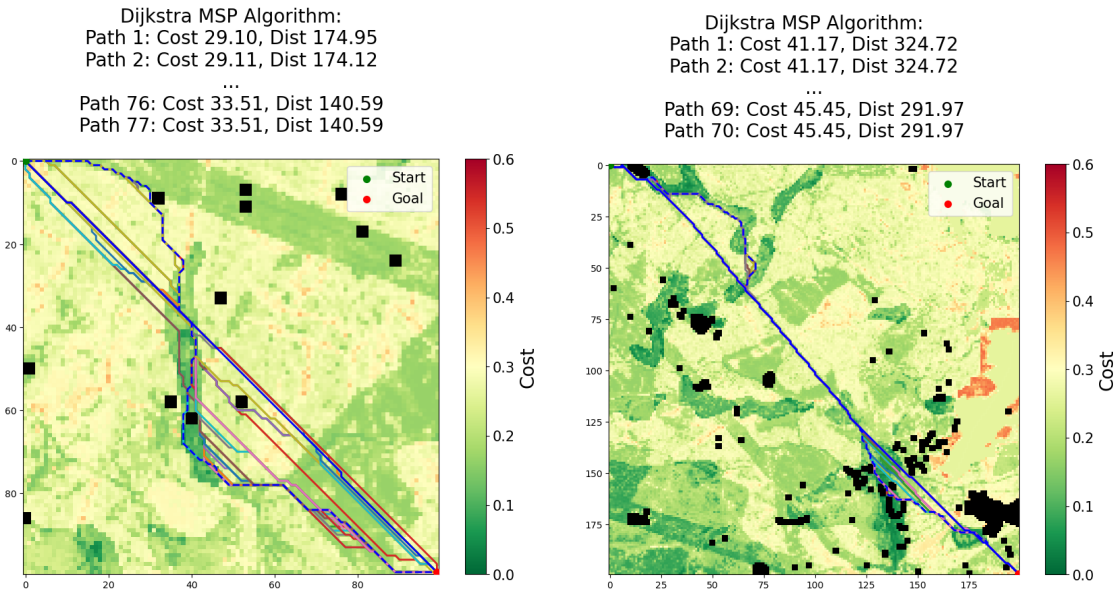


(a) 100×100 subgrid

(b) 200×200 subgrid

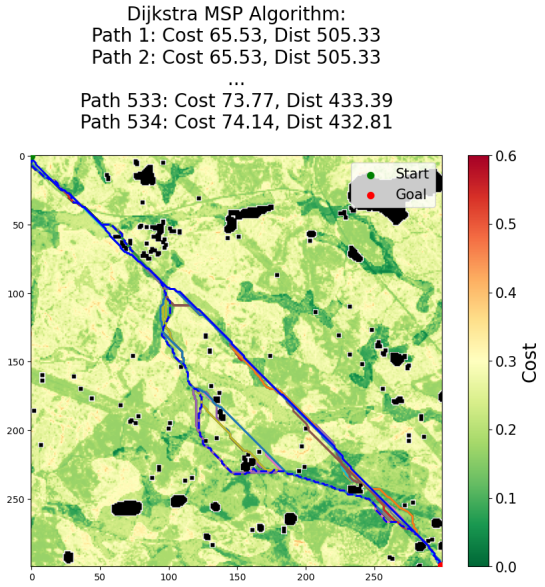
(c) 300×300 subgrid

Figure A.3: Results of the Dijkstra Classic algorithm on the three different subgrids.



(a) 100×100 subgrid

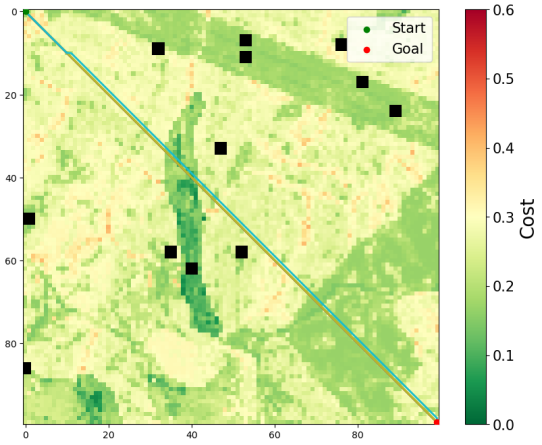
(b) 200×200 subgrid



(c) 300×300 subgrid

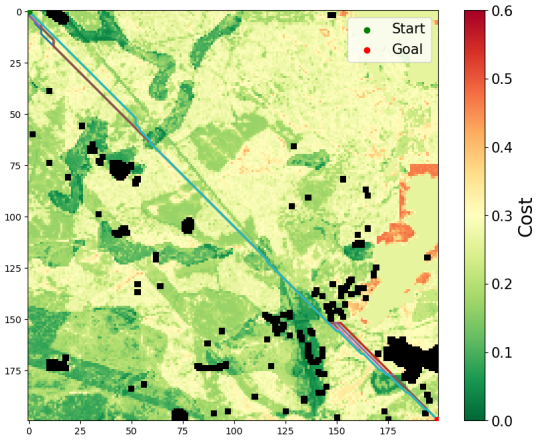
Figure A.4: Results of the Dijkstra MSP algorithm on the three different subgrids.

A* Algorithm:
Path 1: $w_c = 0.0$, Cost: 34.08, Dist: 140.01
Path 2: $w_c = 0.1$, Cost: 34.08, Dist: 140.01
Path 3: $w_c = 0.2$, Cost: 34.08, Dist: 140.01
Path 4: $w_c = 0.3$, Cost: 34.08, Dist: 140.01
Path 5: $w_c = 0.4$, Cost: 34.08, Dist: 140.01
Path 6: $w_c = 0.6$, Cost: 34.08, Dist: 140.01
Path 7: $w_c = 0.7$, Cost: 34.08, Dist: 140.01
Path 8: $w_c = 0.8$, Cost: 34.08, Dist: 140.01
Path 9: $w_c = 0.9$, Cost: 34.08, Dist: 140.01
Path 10: $w_c = 1.0$, Cost: 33.71, Dist: 140.59



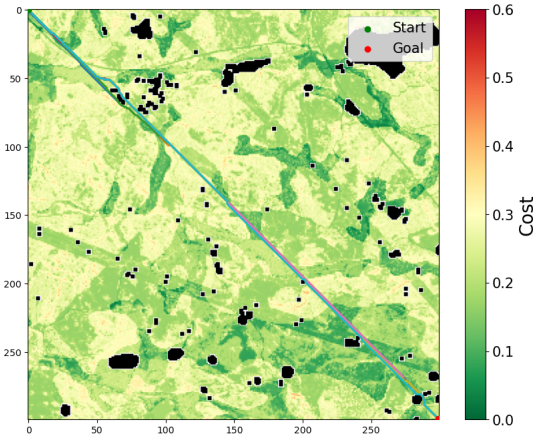
(a) 100×100 subgrid

A* Algorithm:
Path 1: $w_c = 0.0$, Cost: 59.87, Dist: 284.94
Path 2: $w_c = 0.1$, Cost: 59.86, Dist: 284.94
Path 3: $w_c = 0.2$, Cost: 59.86, Dist: 284.94
Path 4: $w_c = 0.3$, Cost: 59.86, Dist: 284.94
Path 5: $w_c = 0.4$, Cost: 59.29, Dist: 285.77
Path 6: $w_c = 0.6$, Cost: 59.26, Dist: 284.94
Path 7: $w_c = 0.7$, Cost: 58.87, Dist: 284.94
Path 8: $w_c = 0.8$, Cost: 58.85, Dist: 284.94
Path 9: $w_c = 0.9$, Cost: 58.81, Dist: 284.94
Path 10: $w_c = 1.0$, Cost: 58.75, Dist: 284.94



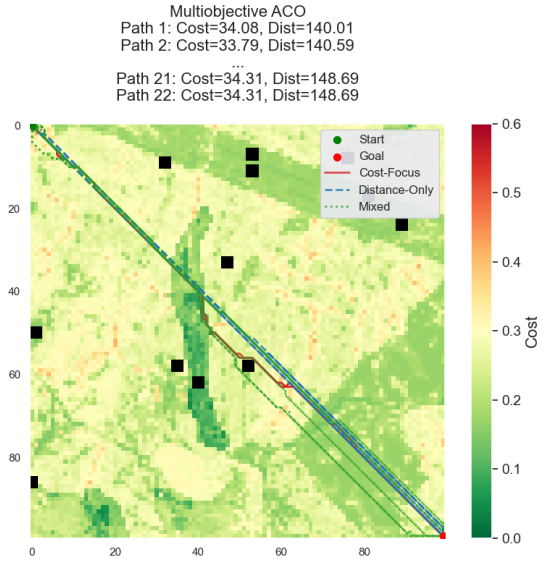
(b) 200×200 subgrid

A* Algorithm:
Path 1: $w_c = 0.0$, Cost: 80.61, Dist: 427.78
Path 2: $w_c = 0.1$, Cost: 80.45, Dist: 427.78
Path 3: $w_c = 0.2$, Cost: 80.53, Dist: 427.78
Path 4: $w_c = 0.3$, Cost: 80.92, Dist: 428.36
Path 5: $w_c = 0.4$, Cost: 81.03, Dist: 428.36
Path 6: $w_c = 0.6$, Cost: 81.05, Dist: 428.36
Path 7: $w_c = 0.7$, Cost: 81.07, Dist: 428.36
Path 8: $w_c = 0.8$, Cost: 81.29, Dist: 430.95
Path 9: $w_c = 0.9$, Cost: 81.28, Dist: 430.95
Path 10: $w_c = 1.0$, Cost: 81.21, Dist: 430.71

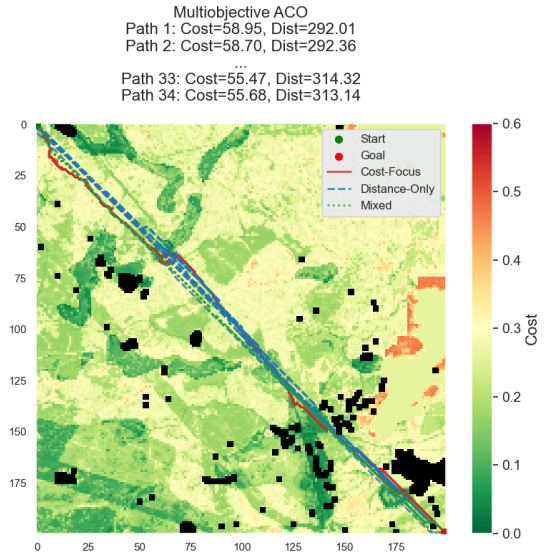


(c) 300×300 subgrid

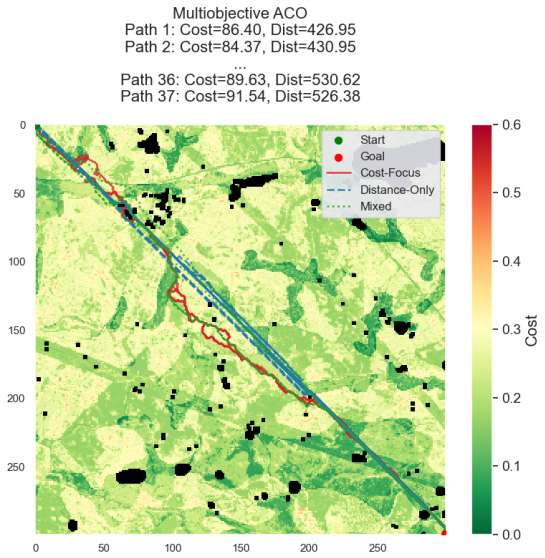
Figure A.5: Results of A* on the three different subgrids.



(a) 100×100 subgrid

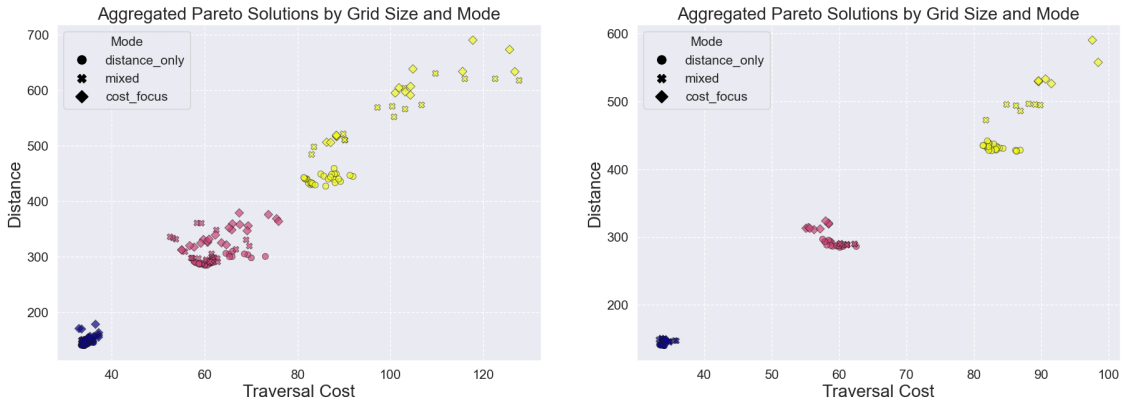


(b) 200×200 subgrid



(c) 300×300 subgrid

Figure A.6: Results of ACO on the three different subgrids used for the hyperparameter tuning.



(a) $q_0 = \{0.9, 0.95, 0.99\}$ (b) $q_0 = 0.99$

Figure A.7: Pareto solutions for the three ACO configurations for different values of q_0 and grid sizes; 100 (blue), 200 (red), and 300 (yellow).

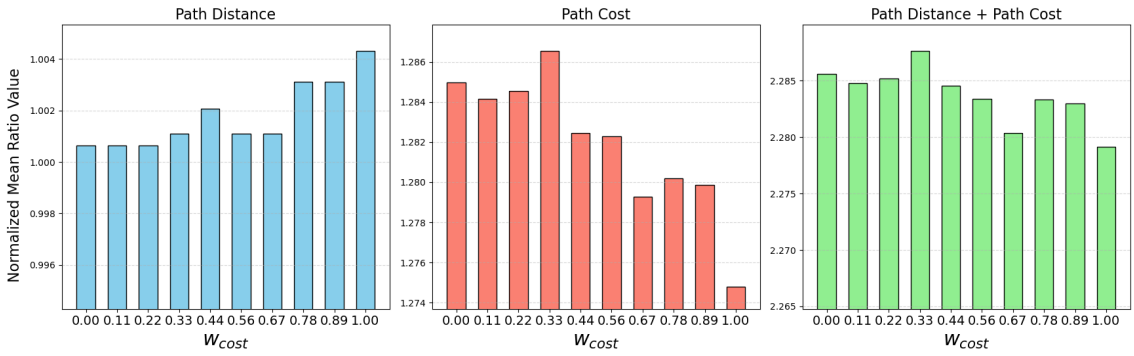
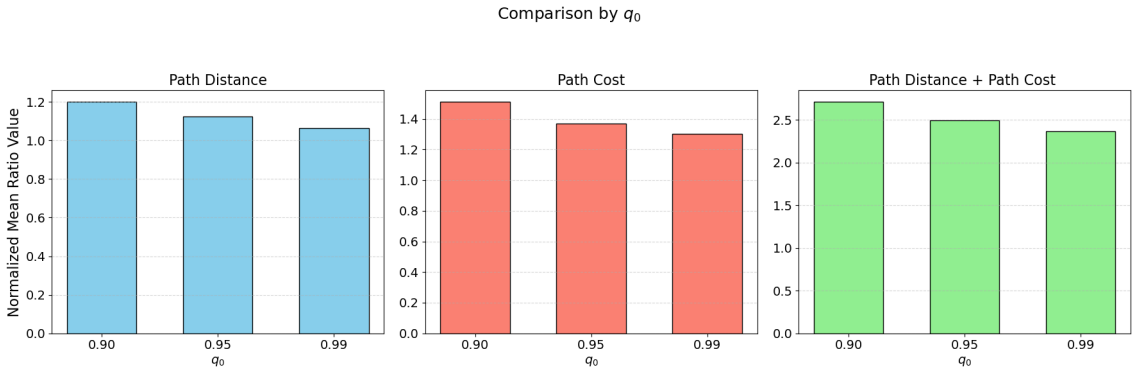
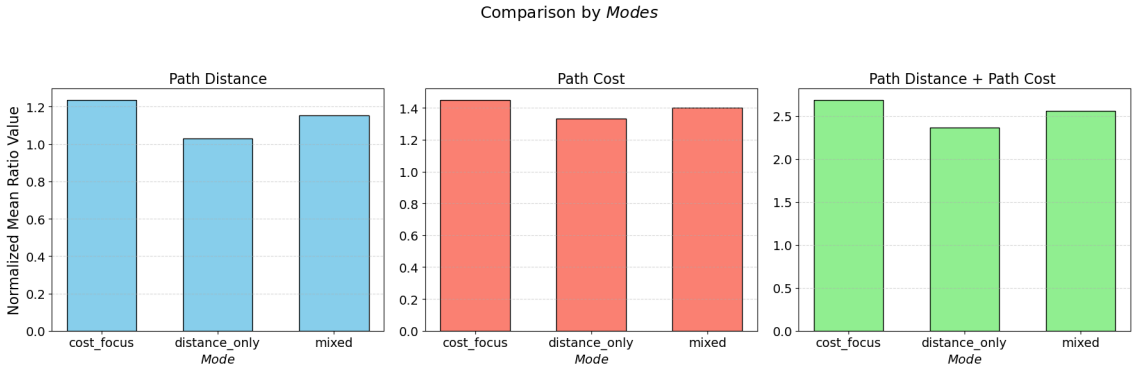


Figure A.8: Bar plots showing the A* algorithm's performance averaged across all subgrid environments, expressed as ratios relative to the shortest ($\mathbf{P_S}$) and cheapest ($\mathbf{P_C}$) reference paths from Dijkstra Classic. The plots display: (left) normalized mean distance ratios, (center) normalized mean cost ratios, and (right) their combined sum. A value of 1 indicates performance equal to the corresponding Dijkstra Classic baseline. Each bar represents one of ten tested values of the cost-heuristic weight w_c.



(a) Evaluation by $q_0 = \{0.9, 0.95, 0.99\}$



(b) Evaluation by mode

Figure A.9: Bar plots showing the performance of the ACO algorithm averaged across all subgrid environments, expressed as normalized mean ratios relative to the shortest ($\mathbf{P_S}$) and cheapest ($\mathbf{P_C}$) reference paths from Dijkstra Classic. Each subplot displays: (left) normalized mean path distance, (center) normalized mean path cost, and (right) their combined sum. A value of 1 indicates parity with the respective Dijkstra baseline. Subfigure (a) compares performance across three ACO greediness parameters $q_0 = \{0.90, 0.95, 0.99\}$. Subfigure (b) compares performance across three optimization modes: `cost_focus`, `distance_only`, and `mixed`.

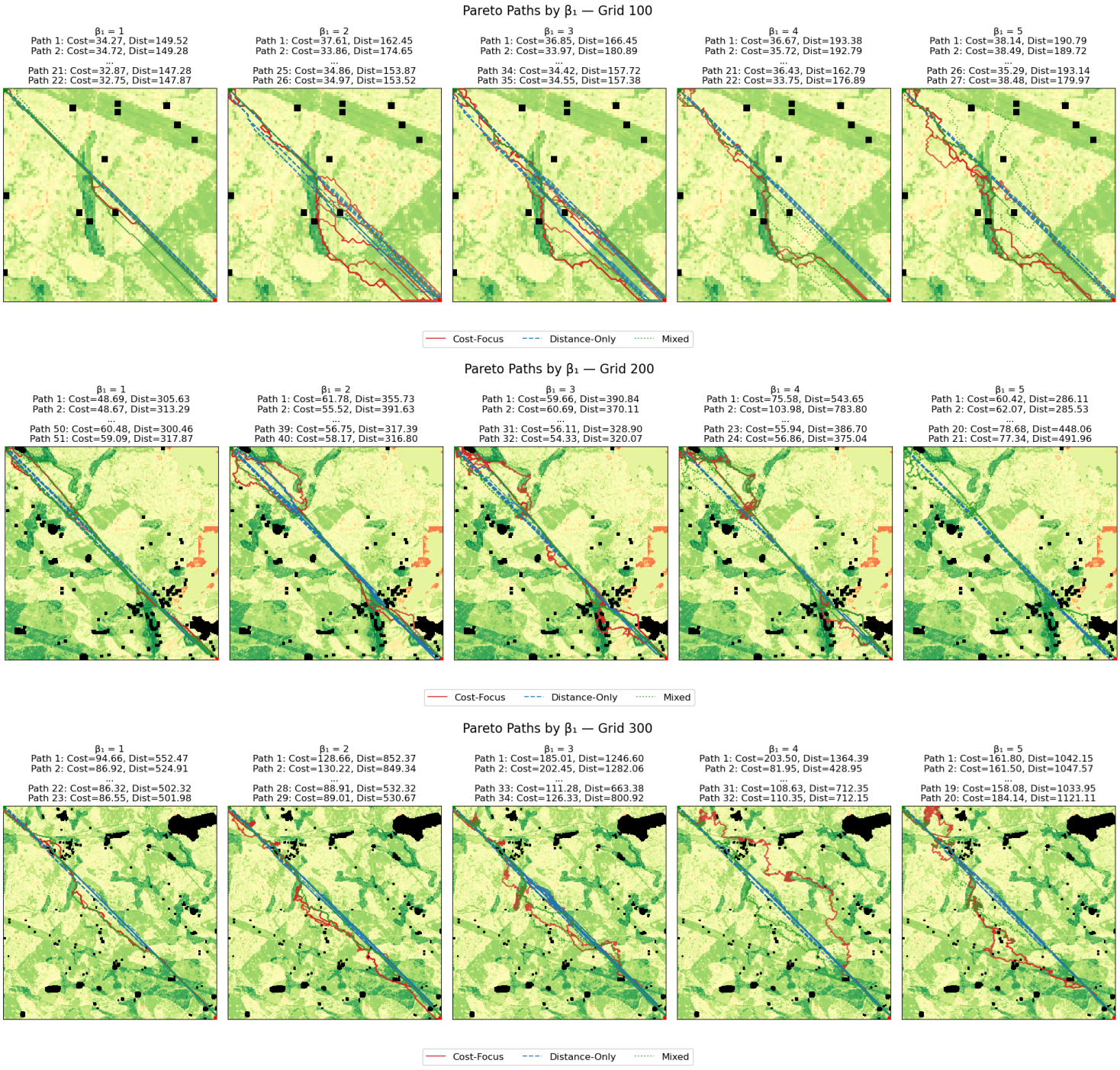


Figure A.10: Pareto paths generated using increasing values of β_1 (1 to 5). Higher β_1 values strengthen the influence of the cost-based heuristic, highlighting regions of strategic interest despite slightly higher objective costs.

A.2 Exploration of Reinforcement Learning for Routing

This appendix details the Convergent Dueling Double Deep Q-Network (C-D3QN) approach, a reinforcement learning model, that was examined during the project for its potential in optimizing transmission line routing and identifying Pareto-optimal paths. As the DQN is a model-free reinforcement learning algorithm, its appeal lay in the ability to modify the reward system based on movements within the environment, offering flexibility to incorporate multiple constraints and preferred routing options. The objective was to investigate the possibilities of training neural networks for different routing strategies based on the given environment. The model draws inspiration from the integration of prioritized experience replay and the Dueling Double DQN structure in prior work on overhead transmission line planning using D3QN-PER [3], while also incorporating the convergence-guaranteeing loss formulation introduced in C-DQN [45].

However, due to several factors, the decision was made not to fully develop and integrate this model into the main body of the project work presented. The primary reasons include the novelty of the specific combined approach (C-D3QN for this exact problem), its significant computational expense, the reliance on findings from limited prior work [3], and the substantial research effort required for hyperparameter tuning and reward system design. Consequently, to ensure timely project completion, the authors decided to sideline the C-D3QN exploration. The following sections document the theoretical background, components, and trials associated with this model.

A.2.1 Explanation of the C-D3QN Framework

Q-learning is a widely used reinforcement learning algorithm that learns an optimal action-selection policy for a Markov Decision Process (MDP) without requiring a model of the environment. Q-learning estimates the action-value function $Q(s, a)$, which represents the expected cumulative reward obtained by taking action a in state s and following an optimal policy thereafter. This estimation is iteratively updated using the Bellman equation [63, 64]:

$$Q(s,a) = Q(s,a) + \alpha \left[r + \gamma \max_{a'} Q(s',a') - Q(s,a) \right]$$

- α is the learning rate, controlling the step size of updates.
- γ is the discount factor, balancing immediate and future rewards.
- r is the immediate reward received after taking action a.
- s' is the next state reached after action a.

Q-learning updates a table storing $Q(s, a)$ values for all state-action pairs. While effective in small environments, this method becomes infeasible as the state space grows, due to the exponential increase in the number of possible state-action combinations [3]. The storage and update of such a Q-table quickly become computationally impractical. To address this, the Deep Q-Network (DQN) is introduced which is a reinforcement learning algorithm designed to solve sequential decision-making problems in high-dimensional state spaces. It extends traditional Q-learning by approximating the Q-value function using a deep neural network, allowing it to generalize across large state spaces [3]. The architectural distinction between Q-learning and Deep Q-learning is illustrated in Figure A.11. While Q-learning relies on a discrete lookup table for storing Q-values, DQN employs a neural network to predict Q-values directly from input states. This mapping from states to Q-value estimates for all possible actions allows DQN to operate efficiently in environments where traditional methods would fail due to scale.

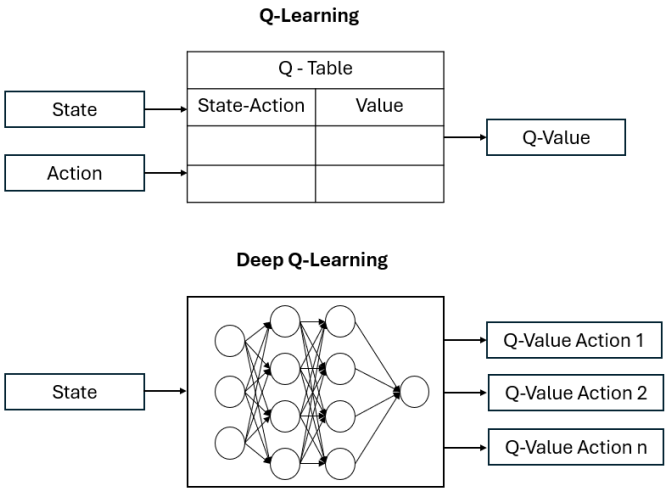


Figure A.11: A schematic illustration highlighting the distinction between Q-learning and Deep Q-learning. Q-learning directly stores Q-values for each state-action pair, while Deep Q-learning uses a neural network to approximate these Q-values. Inspiration of the figure is taken from [65].

A.2.1.1 State Representation

In this implementation of the C-D3QN agent, the state is designed to give the agent both a local and global understanding of its environment.

The main local feature is the **local patch** - a $N \times N$ area taken from the cost matrix, centered on the agent's current position. This patch provides detailed information about the nearby terrain and obstacles. When the agent is near the edge of the grid, padding is added using the highest cost value in the matrix to keep the patch size consistent. This patch is then passed through a Convolutional Neural Network (CNN), which extracts features from the spatial structure of the terrain. The CNN output is flattened into a vector.

In addition to this local view, the state also includes the agent's current coordinates and a normalized vector pointing from its position to the goal. This gives the agent a sense of direction and overall progress. This setup differs from previous work such as [3], where the previous position was used instead of a goal vector—mainly to handle direction changes for overhead lines and construction of towers.

Finally, the CNN features and the position/direction information are combined into a single vector. This full state representation is fed into the rest of the network, helping the agent make decisions based on both its immediate surroundings and where it's trying to go.

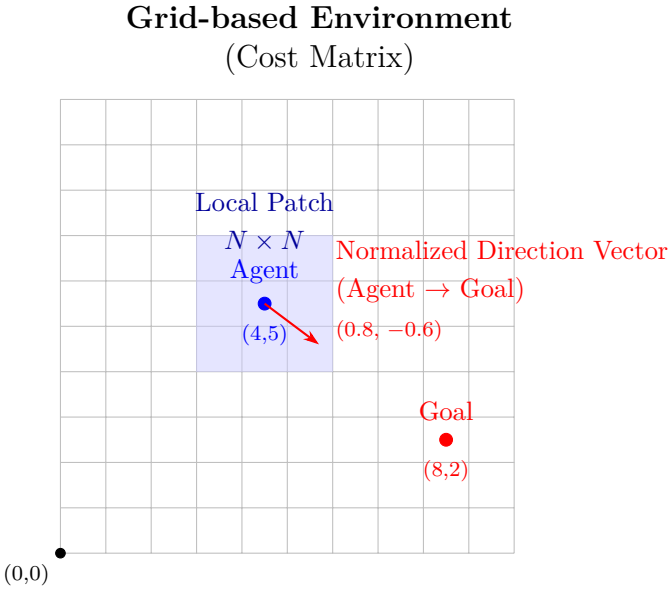


Figure A.12: State representation in C-D3QN: origin $(0, 0)$, local cost patch (blue), agent and goal positions, and normalized direction vector.

A.2.1.2 Standard DQN Algorithm

The goal of DQN is to learn an optimal policy that maximizes the cumulative reward in a MDP. It uses a neural network to approximate the Q-value function. Training is performed by minimizing the temporal difference (TD) error:

$$\delta = r + \gamma \max_{a'} Q(s', a') - Q(s, a)$$

This is achieved through the following steps:

(1) **Initialize** a deep neural network $Q_\theta(s, a)$ with network parameters θ (weights).

(2) **Observe** the initial state s_0.

(3) **Select an action** using an ϵ-greedy strategy, where ϵ is the probability of taking a random action for exploration.

(4) **Execute the action**, transition to the next state s', and receive a reward r.

(5) **Store the experience** (s, a, r, s') in a replay buffer.

(6) **Sample a batch** of past experiences from the buffer.

(7) **Compute the TD target:**

$$y = r + \gamma \max_{a'} Q_\theta(s', a')$$

(8) **Update network parameters** by minimizing the mean squared error loss function:

$$\mathcal{L}(\theta) = \mathbb{E}\left[(y - Q_\theta(s, a))^2\right]$$

The expectation operator $\mathbb{E}$ appears because the loss is optimized over the entire distribution of experiences.

(9) Repeat steps (3)–(8) until convergence.

While DQN has been successful in a variety of applications, it suffers from instability and slow convergence due to issues such as correlated training samples and overestimation of Q-values. To address these problems, several improvements have been introduced [45, 3], described below.

A.2.1.3 Double DQN

One of the key challenges with standard DQN is its tendency to overestimate Q-values. This issue arises because DQN employs a max operation over the next-state Q-values when computing the target, which often leads to systematic overestimation of action values. This bias is problematic because it can cause the learning process to favor suboptimal policies based on inflated Q-value estimates.

To mitigate this issue, Double DQN (DDQN) introduces a fundamental modification: it decouples action selection and action evaluation by leveraging two separate networks - a *policy network* and a *target network*. The improved target calculation in DDQN is formulated as:

$$y = r + \gamma Q_{\theta^-}(s', \arg\max_{a'} Q_\theta(s', a'))$$

Here, $Q_\theta(s', a')$ refers to the policy network, which is used to predict the Q-values for the next state-action pairs. The term $\arg\max_{a'} Q_\theta(s', a')$ selects the action with the highest predicted Q-value according to this policy network. Meanwhile, $Q_{\theta^-}(s', a')$ denotes the target network, which is used to evaluate the Q-value of the selected action. This separation helps mitigate overestimation by preventing the same network from both selecting and evaluating actions.

Moreover, the target network is updated only every few episodes, rather than after every step. This periodic synchronization has been shown to yield more stable and performant learning in practice [3].

As a result, DDQN reduces value bias and enhances the stability of the training process, leading to more reliable and accurate policy learning [3].

A.2.1.4 Dueling DQN

The standard DQN computes a single Q-value per state-action pair. This can, however, lead to inefficient learning, as the value of some states depends more on the state itself rather than the specific action taken to get into that state. To mitigate this, Dueling DQN decomposes the Q-value function into two separate components [45]:

- **State Value Function** $V(s)$, representing the general desirability of a state.
- **Advantage Function** $A(s, a)$, capturing the relative value of each action in that state.

The Q-value function is then computed as:

$$Q(s, a; \theta) = V(s; \theta) + \left(A(s, a; \theta) - \frac{1}{|A|} \sum_{a'} A(s, a'; \theta) \right)$$

The advantage function is normalized to ensure that the overall Q-value remains well-calibrated.

This decomposition allows the network to better distinguish between states where action choice is critical and states where any action leads to a similar outcome, thereby improving training efficiency [3].

A.2.1.5 Prioritized Experience Replay

As the DQN is non-linear and can have many local minima one can further mitigate this issue using a technique known as *Experience Replay* (ER). ER involves the agent saving a selection of its experiences (including state, action, reward, and next state) in a memory buffer. The agent then samples from this buffer to update the Q-function, which helps to reduce data correlation and stabilize the learning process [65]. However, standard ER in DQN samples past experiences uniformly, which can lead to inefficient learning if important transitions are rarely revisited. To improve learning efficiency, Prioritized Experience Replay (PER) is introduced [3].

PER assigns a priority score to each experience i based on the magnitude of its TD error:

$$p_i = |\delta_i| + \epsilon, \quad 0 < \epsilon \ll 1$$

During training, experiences are sampled with probability:

$$P(i) = \frac{p_i^\sigma}{\sum_j p_j^\sigma}$$

Here, σ controls how strongly prioritization is enforced ($\sigma = 0$ corresponds to uniform distribution). This ensures that transitions with higher TD errors, which are more informative for learning, are replayed more frequently.

To prevent overfitting to high-priority samples, a β factor is introduced, controlling the correction factor to ensure an unbiased estimate of the expected Q-values. This is done by applying importance weights sampling [66]:

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)} \right)^\beta$$

N is the buffer size and $\beta \in [0, 1]$ adjusts the strength of the correction. As β approaches 1, the update becomes fully unbiased. These weights are then multiplied with the loss function, ensuring that high-priority transitions influence the update while still preserving a more unbiased update mechanism.

A.2.1.6 Convergent Deep Q-Network Loss

Despite previous implementations, the model continued to suffer from high computational costs during pathfinding. To address this, and with the aim of further improving convergence stability and reducing training time, the Convergent Deep Q-Network (C-DQN) was explored as a potential solution.

C-DQN enhances standard DQN by integrating both the conventional DQN loss (mean squared error loss) and the Mean Squared Bellman Error (MSBE) loss. Instead of relying on a single objective, it dynamically selects the larger of the two loss terms at each update step. This conservative approach ensures that the learning process prioritizes stability - particularly when target network updates risk increasing the loss - thus promoting more reliable convergence behavior.

The standard DQN loss of current state-action pair can be seen as the error between the policy network $Q_\theta(s, a)$ and the target network Q_{θ^-}:

$$L_{\text{DQN}} = \mathbb{E}\left[\left(Q_\theta(s, a) - (r + \gamma Q_{\theta^-}(s', a')) \right)^2 \right], \tag{A.1}$$

Meanwhile, the MSBE loss differs by utilizing only the current policy network:

$$L_{\text{MSBE}} = \mathbb{E}\left[(Q_\theta(s,a) - (r + \gamma Q_\theta(s',a')))^2\right], \tag{A.2}$$

C-DQN mitigates instability by adopting an adaptive loss function:

$$L_{\text{C-DQN}} = \max(L_{\text{DQN}}, L_{\text{MSBE}}). \tag{A.3}$$

This formulation ensures that the learning process is guided by the worst-case scenario, preventing optimistic updates that could lead to divergence.

In the explored implementation, the C-DQN loss was computed at each training step as follows:

(1) Compute $Q_\theta(s,a)$ from the current policy network.

(2) Compute the DQN target using the target network:

$$y_{\text{DQN}} = r + \gamma Q_{\theta^-}(s', \arg\max_{a'} Q_\theta(s',a')).$$

(3) Compute the MSBE target using the policy network:

$$y_{\text{MSBE}} = r + \gamma Q_\theta(s', \arg\max_{a'} Q_\theta(s',a')).$$

(4) Compute the individual losses:

$$L_{\text{DQN}} = (Q_\theta(s,a) - y_{\text{DQN}})^2, \quad L_{\text{MSBE}} = (Q_\theta(s,a) - y_{\text{MSBE}})^2.$$

(5) Use the maximum loss:

$$L_{\text{C-DQN}} = \max(L_{\text{DQN}}, L_{\text{MSBE}}).$$

A.2.1.7 Dynamic Exploration and Learning/Discount Rate Decay

To balance exploration and exploitation, an ϵ-greedy strategy was used where ϵ decays exponentially over time:

$$\epsilon_t = \max(\epsilon_{\min}, \epsilon_{\text{start}} \cdot \left(\frac{\epsilon_{\min}}{\epsilon_{\text{start}}}\right)^{\frac{t}{T}})$$

- ϵ_{start} is the initial exploration rate.
- $\epsilon_{\min}$ is the minimum exploration rate.
- t is the current episode.

- T is the total number of episodes.

This approach ensures that exploration decreases smoothly over time while preventing premature convergence.

Similarly, the **discount factor** γ was updated dynamically using an exponential increase from the initial discount factorγ_{start} to the maximum discount factor γ_{max}:

$$\gamma_t = \min(\gamma_{\text{max}}, \gamma_{\text{start}} \left(\frac{\gamma_{\text{max}}}{\gamma_{\text{start}}} \right)^{\frac{t}{T}})$$

This ensures that early in training, the model prioritizes short-term rewards, while later, it gradually shifts towards optimizing long-term rewards.

Furthermore, the **learning rate** α_t at time step t followed an exponential decay controlled by a scheduler:

$$\alpha_t = \alpha_{\text{start}} \cdot \lambda^t$$

With $\lambda \in (0, 1)$, α_t gradually reduces the step size of weight updates, improving stability and convergence.

By dynamically adapting exploration rate, discount factor, and learning rate, the model can efficiently transition from an exploratory phase to a stable and optimized policy over training.

A.2.1.8 Concluding Remarks on C-D3QN Exploration

The combination of Dueling DQN, Double DQN, PER, and the adaptive C-DQN loss (incorporating MSBE) represents a sophisticated approach to reinforcement learning. These enhancements collectively aim to improve stability, convergence speed, and overall performance compared to standard DQN. While not pursued to final integration in this project, the theoretical framework detailed here demonstrates a potentially powerful method for tackling complex sequential decision-making problems like optimal cable routing in challenging environments [3, 45]. Further research, particularly in hyperparameter optimization and reward shaping, would be necessary to fully evaluate its practical efficacy for underground transmission line routing.

Below is the architecture of the explored C-D3QN algorithm. The two inputs (local cost patch and position + goal vector) follow separate processing branches, are concatenated, passed through shared fully-connected layers, and finally split into *Value* and *Advantage* streams. These streams then recombine, to produce Q-values for all actions, according to:

$$Q(s, a) = V(s) + \left(A(s, a) - \frac{1}{|A|} \sum_{a'} A(s, a') \right)$$

Arrows indicate the forward data flow during inference and training.

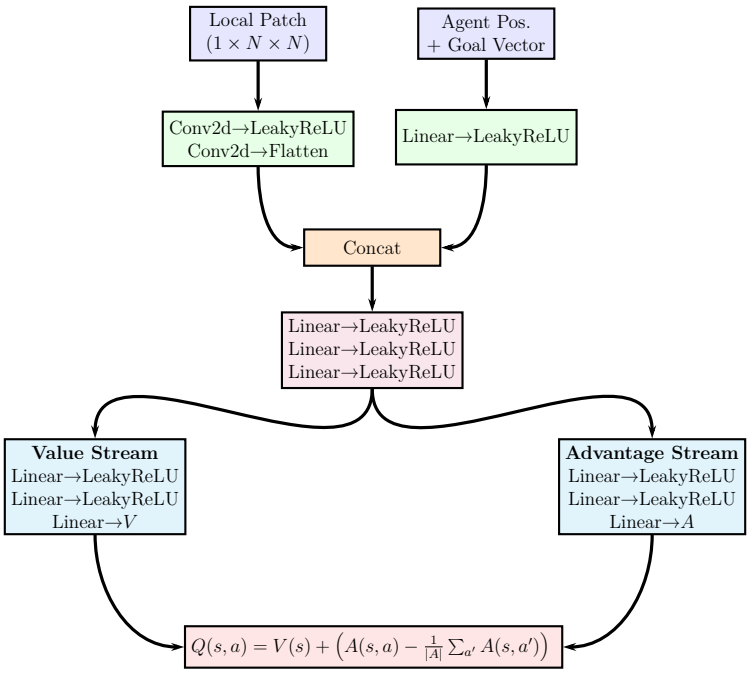


Figure A.13: Network architecture of the C-D3QN model. The local patch and agent-goal vector are processed through separate branches before being fused and passed through shared fully connected layers. The output splits into value and advantage streams, which are combined to compute the final Q-values.

A.3 Landslide Risk Classification Models

During the project, various landslide classification models were explored as a side investigation alongside the main work. The goal was to assess whether machine learning models could accurately classify landslide-prone areas based on available geospatial data layers. However, after extensive evaluation, this aspect was deemed outside the project's primary scope and too time-consuming to integrate effectively. The results and methodology followed are documented here for completeness.

A.3.1 Input Data

The data used for classification included five input features representing different terrain and environmental attributes:

(1) Soil type

(2) Land use

(3) Forest vegetation fraction

(4) Slope angle

(5) Elevation

These features were extracted using different regions for training, validation and testing, as seen in Figure A.14. A northern region in Sweden was used in order to facilitate more landslide risk areas, hence reducing the data imbalances. The sizes were 3500, 900 and 1500 km^2 respectively. After consulting with the geology department at WSP, the minimum size of an landslide susceptible area was deemed difficult to pinpoint. Anyhow, the data was resampled from cell size 10×10 meters to 50×50, in order to facilitate a more reasonable minimum area of landslide suitability. This also decreased the data sizes and speed up training of the models. All sets were normalized to ensure consistent data scales and more efficient feature extraction [67]. A binary label was used to indicate whether a grid cell represented a landslide risk area, as seen in Figure A.15(a).

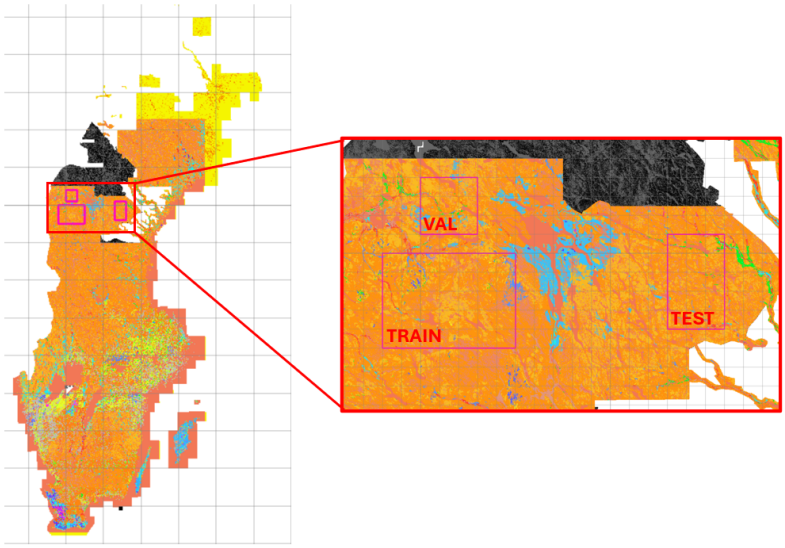


Figure A.14: The different regions for extracting the train, validation and test sets for the landslide models. The background shows some of the different data layers.

A.3.2 Models

A.3.2.1 Deep Neural Network

The first approach tested was a fully connected deep neural network (DNN). Once extracted, the data was oversampled in favor of the minority class (landslide risk = 1) to mitigate class imbalance, inspired from [68]. The network had the layout of seven layers (five hidden), with $[5, 16, 32, 64, 32, 16, 1]$ neurons respectively. LeakyReLU was used as activation function and the AdamW as optimizer, which is an implementation of the Adam algorithm with weight decay, a regularization technique to prevent overfitting. A batch size of 256 was used.

A.3.2.2 Autoencoder

Since convolutional neural networks (CNNs) have been found to be highly effective for landslide prediction [69], a CNN approach was initially tested. However, due to the lack of spatial imagery input, this approach was not a suitable fit for this dataset. Instead, the model was revised into an autoencoder (AE) approach, something previously used within field [68, 70]. Autoencoders are particularly useful for imbalanced data, as they allow the classifier to work with a latent space representation, reducing the impact of the majority class.

A standard AE consists of three parts:

- An **encoder** that compresses input into a lower-dimensional latent space.
- A **decoder** that reconstructs the input from the latent space.

- A **classifier** that uses latent space representations to classify landslide risk.

The encoder hade the shape $[5, 32, 16, 2]$, whereas the decoder had the shape $[2, 32, 16, 5]$. Hence, the latent space was two-dimensional, making sure to compress the relationship between the input parameters and the landslide susceptibility risk. Then, the classifier hade the shape $[2, 32, 16, 1]$, in order for binary classification.

Beside this standard AE setup, a PCA-Inspired (Principal Component Analysis) AE was also implemented based on [71, 72], introducing tied weights, orthogonal constraints, and uncorrelated features. This is mostly done in order to improve latent space representations. The PCA-Inspired AE follows a similar structure but introduces key modifications:

- **Tied Weights**: The decoder weights are constrained to be the transpose of the encoder weights, reducing the number of trainable parameters.
- **Orthogonal Weights**: The encoder layers are constrained to have orthogonal weight matrices, ensuring better feature independence.
- **Uncorrelated Features**: The model penalizes correlated latent representations to encourage diverse features.
- **Unit Norm Constraint**: Ensures that each layer's weights have a unit norm, preventing excessively large weights that could lead to unstable training.

The encoder had the same shape as the previous setup, whereas the decoder here uses tied weights from the encoder layers for reconstruction from the latent space. The shape is $[2, 16, 5]$.

Since the AEs are using unsupervised training, and the classifier supervised training, the data was split evenly between the AE and the classifier. A batch size of 256 was used.

A.3.2.2.1 Evaluation Metrics Initially, accuracy was used as the evaluation metric. However, due to the severe class imbalance and the nature of the problem (where a FN is worse than a FP), accuracy alone proved insufficient. Instead, additional metrics were introduced, considering the ratios in Table A.1, also used in previous studies [67, 70]:

- **Accuracy**: $\frac{TP+TN}{TP+TN+FP+FN}$
- **Precision**: $\frac{TP}{TP+FP}$
- **Recall (Sensitivity)**: $\frac{TP}{TP+FN}$
- **F1-Score**: $2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$

- **Matthews Correlation Coefficient (MCC)**:

$$MCC = \frac{TP \times TN - FP \times FN}{\sqrt{(TP + FP)(TP + FN)(TN + FP)(TN + FN)}}$$

- **Overall Score**: Accuracy + F1 + MCC

Table A.1: Confusion matrix of true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN) in the case of landslide predictions.

	Actual Negative (No landslide)	**Actual Positive (Landslide)**
Predicted Negative (No landslide)	TN	FN
Predicted Positive (Landslide)	FP	TP

A.3.3 Results and Discussion

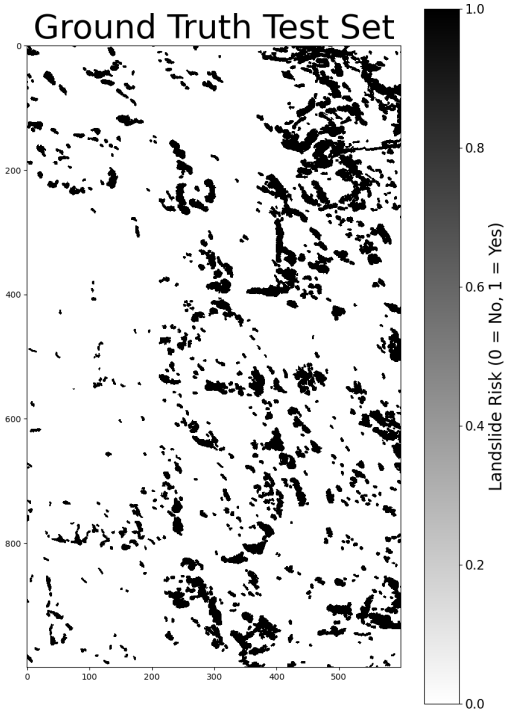
Training both the DNN and the AEs (both autoencoder and classifier components) for 30 epochs each, they achieved the results seen in Figure A.15 and Table A.2 on the test set.

Table A.2: Performance comparison of the DNN, Standard AE, and PCA-Inspired AE on the test set after 30 training epochs on the training set. Scores marked in red and green indicate the worst and the best performing model for that given score, respectively. For reference, 1.8861 and 2.4975 are the lowest and highest overall score respectively acheived in [67].

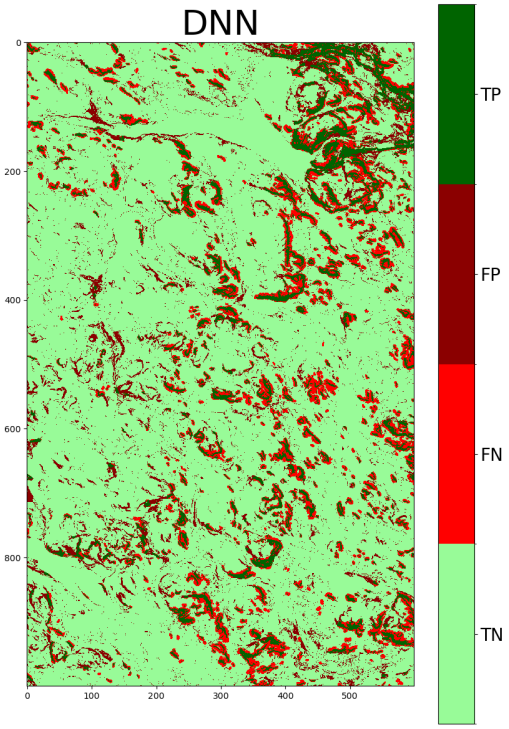
Model	**Accuracy**	**Precision**	**Recall**	**F1-Score**	**MCC**	**Overall Score**
DNN	0.8550	0.4799	0.4566	0.4679	0.3842	1.7071
Standard AE	0.8903	0.7691	0.3065	0.4383	0.4408	1.7694
PCA-Inspired AE	0.8816	0.8459	0.1867	0.3059	0.3633	1.5508

As seen in Figure A.15(b), the DNN classifies landslide risk areas (true positives) more effectively than the AEs. Consequently, it has a low number of false negatives, contributing to its high recall score. However, since the DNN is more inclined to classify areas as positive (risk areas, 1), it overshoots and produces a large number of false positives. This leads to a significantly lower precision score, which is reflected in the increased ”noise” in the colormap. The high false positive rate also negatively impacts the accuracy score, as accuracy simply measures the proportion of correctly classified samples.

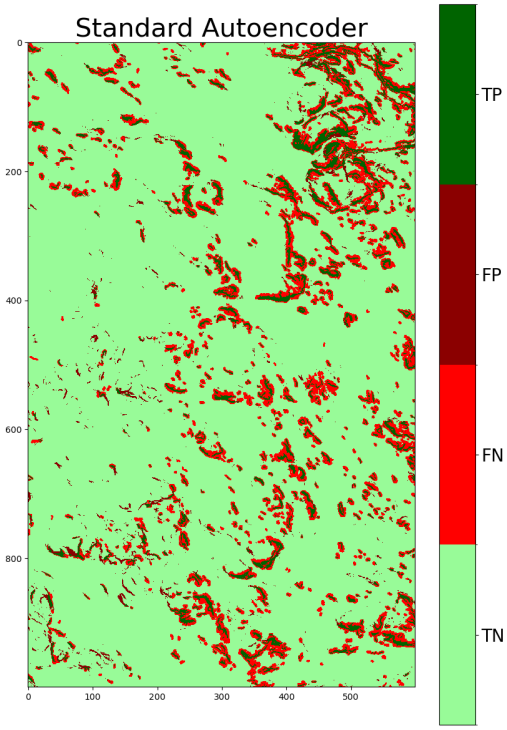
For precision, the PCA-Inspired AE performs the best, as it appears more conservative in labeling areas as positive. However, this conservatism results in a higher



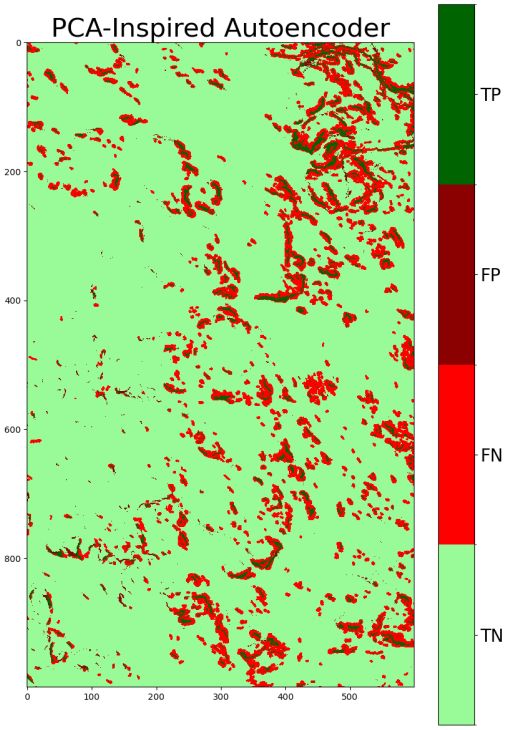
(a) Ground Truth Test Set



(b) DNN Predictions



(c) Standard AE Predictions



(d) PCA-Inspired AE Predictions

Figure A.15: Comparison of landslide risk classifications from different models. (a) Ground truth, (b) DNN performance, (c) Standard AE predictions, and (d) PCA-Inspired AE predictions.

number of false negatives, leading to a lower recall score. While the Standard AE nearly doubles the recall of the PCA-Inspired AE, the DNN still achieves the highest recall.

The F1-score, which balances precision and recall through a harmonic mean, also favors the DNN. However, the Matthews Correlation Coefficient (MCC), considered a more reliable metric due to its invariance to class renaming and ability to account for class imbalances across all four confusion matrix categories [73, 74], indicates that the Standard AE performs best. This suggests that it is the overall most balanced model for classifying landslide risk areas. It also achieves the highest overall score. This is evident in Figure A.15(c), where the Standard AE produces fewer false positives than the DNN while also detecting more true positives than the PCA-Inspired AE. However, in this application, false negatives are more critical than false positives, as misclassifying landslide-prone areas as no-risk regions could have severe consequences. From this perspective, the PCA-Inspired AE performs the worst, followed by the Standard AE, with the DNN being the most effective at minimizing false negatives.

Thus, despite the Standard AE outperforming the DNN in some metrics, the DNN remains the more reliable option for avoiding the misclassifications of landslide risk areas as no-risk zones.

A.3.4 Conclusion

The AE-based approach demonstrated slightly better metric performance than the DNN, but less reliability for correctly classifying landslide risk areas. Hence, from this perspective, the DNN is the best model of the three tested. The results indicate that the dimensionality reduction in the AEs can be an effective technique for landslide susceptibility mapping, particularly in handling class imbalance by leveraging latent space representations. However, further work is needed to refine the models and their respective performance. For reference, the lowest and highest overall scores reported in [67] were 1.8861 and 2.4975, respectively, suggesting that substantial improvements are possible. These include:

- **Extended Training**: Increasing the number of training epochs to allow the models to converge more effectively.
- **Hyperparameter Tuning**: Optimizing key parameters such as learning rate, dropout rate, and layer size to improve generalization.
- **Feature Engineering**: Exploring additional input features that could enhance the model's ability to distinguish between landslide-prone and stable areas.
- **Alternative Architectures**: Testing other architectures, such as variational AEs (VAEs) or attention-based models

- **Post-Processing Techniques**: Implementing threshold optimization or calibrated classification thresholds to further reduce false positives and false negatives.

Due to time constraints, the investigation was not extended further. Given the complexity and scope of the problem, this area could alone serve as a dedicated master's thesis project, focusing on the development and optimization of deep learning models for landslide susceptibility mapping. Despite being a side study, this work demonstrates the potential of machine learning in landslide risk assessment and highlights avenues for further research.

A.4 Data Layer Element Descriptions

Table A.3: Elements names in the non-binary data layers.

Layer / Element Number	Element Name (Swedish)	Element Name (English)
r_non_binary_jordartlager		
1	Urberg	Bedrock
2	Mossetorv	Moss peat
3	Postglacial sand	Postglacial sand
4	Isälvssediment	Glaciofluvial sediment
5	Torv	Peat
6	Vatten	Water
7	Glacial golvsilt–finsand	Glacial floor silt–fine sand
8	Isälvssediment, sand	Glaciofluvial sediment, sand
9	Älvsediment, sand	River sediment, sand
10	Postglacial grovlera	Postglacial coarse clay
11	Fyllning	Fill material
12	Svämsediment, grovsilt–finsand	Flood sediment, coarse silt–fine sand
13	Lera	Clay
14	Sandig morän	Sandy moraine
15	Morän	Moraine
16	Svallsediment, grus	Wave-washed sediment, gravel
17	Glacial grovlera	Glacial coarse clay
18	Berg	Rock
19	Postglacial finsand	Postglacial fine sand
20	Oklassat område	Unclassified area
21	Älvsediment sten–block	River sediment, stone–boulder
22	Glasial silt	Glacial silt
23	Morän omväxlande med sorterade sediment	Moraine alternating with sorted sediments
24	Blockmark	Boulder terrain
25	Torv, tidvis under vatten	Peat, periodically submerged
26	Älvsediment, grus	River sediment, gravel
27	Svämsediment, grus	Flood sediment, gravel
28	Svämsediment, ler–silt	Flood sediment, clay–silt
29	Postglacial lera	Postglacial clay
30	Kärrtorv	Fen peat
31	Gyttja	Gytta (organic-rich mud)
32	Flygsand	Aeolian sand
33	Älvsediment	River sediment
34	Silt	Silt
35	Talus (rasmassor)	Talus (rockfall debris)
36	Gyttjelera (eller lergyttja)	Muddy clay (or clay gytta)
37	Grusig morän	Gravelly moraine
38	Postglacial silt	Postglacial silt
39	Svämsediment, sand	Flood sediment, sand
40	Sandig-siltig morän	Sandy-silty moraine
41	Moränlera	Moraine clay
42	Glacial lera	Glacial clay

Layer / Element Number	Element Name (Swedish)	Element Name (English)
43	Älvsediment, grovsilt–finsand	River sediment, coarse silt–fine sand
44	Lera–silt	Clay–silt
45	Lerig morän	Clayey moraine
46	Skålla av sedimentärt berg	Eroded sedimentary rock
47	Slamströmssediment, ler–block	Debris flow sediment, clay–boulder
48	Isälvssediment, sten–block	Glaciofluvial sediment, stone–boulder
49	Moränlera eller lerig morän	Moraine clay or clayey moraine
50	Klapper	Shingle (coarse gravel)
51	Sedimentärt berg	Sedimentary rock
52	Älvsediment, ler–silt	River sediment, clay–silt
53	Isälvssediment, grus	Glaciofluvial sediment, gravel
54	Postglacial finlera	Postglacial fine clay
55	Glacial finlera	Glacial fine clay
56	Rösberg	Rocky debris
57	Svämsediment	Flood sediment
58	Morängrovlera	Moraine coarse clay
59	Vittringsjord	Weathered soil
60	Postglacial grovsilt-finsand	Postglacial coarse silt–fine sand
61	Fyllning, rödfyr	Fill material, red shale
62	Sten–block	Stone–boulder
63	Vittringsjord, sand–grus	Weathered soil, sand–gravel
64	Vittringsjord, ler–silt	Weathered soil, clay–silt
65	Bleke och kalkgyttja	Marl and calcareous gyttja
66	Moränfinlera	Moraine fine clay
67	Fanerozoisk diabas	Phanerozoic diabase
68	Skaljord	Shell-bearing soil
69	Oklassat område, tidvis under vatten	Unclassified area, periodically submerged
70	Kalktuff	Calcareous tuff
71	Lera–silt, tidvis under vatten	Clay–silt, periodically submerged
72	Flytjord eller skredjord	Flow soil or landslide soil
73	Morän, sten–block	Moraine, stone–boulder
74	Sand	Sand
75	Glaciär	Glacier
76	Skålla av sandsten	Eroded sandstone
`r_non_binary_ma_baskarta`		
2	Öppen våtmark	Open wetland
3	Åkermark	Farmland
23	Låg fjällskog på våtmark	Low mountain forest on wetland
41	Öppen fastmark utan vegetation	Open upland without vegetation
42	Öppen fastmark med vegetation	Open upland with vegetation
43	Låg fjällskog på fastmark	Low mountain forest on upland
51	Byggnad	Building
52	Anlagd mark, ej byggnad eller väg/järnväg	Constructed land, excluding buildings or roads/railways

Layer / Element Number	Element Name (Swedish)	Element Name (English)
53	Väg eller järnväg	Road or railway
54	Torvtäkt	Peat extraction site
61	Inlandsvatten	Inland water
62	Hav	Sea
110	Skog på fastmark	Forest on upland
118	Temporärt ej skog på fastmark	Temporarily non-forest on upland
120	Skog på våtmark	Forest on wetland
128	Temporärt ej skog på våtmark	Temporarily non-forest on wetland
`r_non_binary_ma_bete`		
2	Bete	Pasture
`r_non_binary_ma_baskarta`		
0	Flygplatsområde	Airport area
1	Kyrkogårdar/begravningsplatser	Cemeteries/Burial grounds
2	Grus-, berg-, mineraltäkt	Gravel, rock, mineral extraction site
3	Torvtäkt	Peat extraction site
4	Gruvområde	Mining area
5	Koloniområde	Allotment area
6	Campingplats	Campsite
7	Golfbana	Golf course
8	Skidbacke	Ski slope
9	Motorbana	Motor racing track
10	Övrig sport- och idrottsanläggning	Other sports and athletic facility
11	Avfalls-, återvinningsanläggning	Waste and recycling facility

DEPARTMENT OF PHYSICS
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY